# Contents

<table>
<thead>
<tr>
<th>Zabbix Manual</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Copyright notice</td>
<td>5</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>5</td>
</tr>
<tr>
<td>1 Manual structure</td>
<td>5</td>
</tr>
<tr>
<td>2 What is Zabbix</td>
<td>6</td>
</tr>
<tr>
<td>3 Zabbix features</td>
<td>6</td>
</tr>
<tr>
<td>4 Zabbix overview</td>
<td>7</td>
</tr>
<tr>
<td>5 What’s new in Zabbix 6.2.0</td>
<td>8</td>
</tr>
<tr>
<td>6 What’s new in Zabbix 6.2.1</td>
<td>13</td>
</tr>
<tr>
<td>7 What’s new in Zabbix 6.2.2</td>
<td>13</td>
</tr>
<tr>
<td>2. Definitions</td>
<td>13</td>
</tr>
<tr>
<td>3. Zabbix processes</td>
<td>15</td>
</tr>
<tr>
<td>1 Server</td>
<td>15</td>
</tr>
<tr>
<td>2 Agent</td>
<td>22</td>
</tr>
<tr>
<td>3 Agent 2</td>
<td>25</td>
</tr>
<tr>
<td>4 Proxy</td>
<td>27</td>
</tr>
<tr>
<td>5 Java gateway</td>
<td>29</td>
</tr>
<tr>
<td>6 Sender</td>
<td>33</td>
</tr>
<tr>
<td>7 Get</td>
<td>34</td>
</tr>
<tr>
<td>8</td>
<td>JS</td>
</tr>
<tr>
<td>8 Web service</td>
<td>35</td>
</tr>
<tr>
<td>4. Installation</td>
<td>35</td>
</tr>
<tr>
<td>1 Getting Zabbix</td>
<td>35</td>
</tr>
<tr>
<td>2 Requirements</td>
<td>36</td>
</tr>
<tr>
<td>3 Installation from sources</td>
<td>48</td>
</tr>
<tr>
<td>4 Installation from packages</td>
<td>57</td>
</tr>
<tr>
<td>5 Installation from containers</td>
<td>72</td>
</tr>
<tr>
<td>6 Web interface installation</td>
<td>79</td>
</tr>
<tr>
<td>7 Upgrade procedure</td>
<td>84</td>
</tr>
<tr>
<td>8 Known issues</td>
<td>96</td>
</tr>
<tr>
<td>9 Template changes</td>
<td>99</td>
</tr>
<tr>
<td>10 Upgrade notes for 6.2.0</td>
<td>100</td>
</tr>
<tr>
<td>11 Upgrade notes for 6.2.1</td>
<td>101</td>
</tr>
<tr>
<td>5. Quickstart</td>
<td>101</td>
</tr>
<tr>
<td>1 Login and configuring user</td>
<td>102</td>
</tr>
<tr>
<td>2 New host</td>
<td>105</td>
</tr>
<tr>
<td>3 New item</td>
<td>106</td>
</tr>
<tr>
<td>4 New trigger</td>
<td>108</td>
</tr>
<tr>
<td>5 Receiving problem notification</td>
<td>110</td>
</tr>
<tr>
<td>6 New template</td>
<td>114</td>
</tr>
<tr>
<td>6. Zabbix appliance</td>
<td>116</td>
</tr>
<tr>
<td>7. Configuration</td>
<td>118</td>
</tr>
<tr>
<td>1 Hosts and host groups</td>
<td>127</td>
</tr>
<tr>
<td>2 Items</td>
<td>137</td>
</tr>
<tr>
<td>3 Triggers</td>
<td>327</td>
</tr>
<tr>
<td>4 Events</td>
<td>344</td>
</tr>
<tr>
<td>5 Event correlation</td>
<td>347</td>
</tr>
<tr>
<td>6 Tagging</td>
<td>353</td>
</tr>
<tr>
<td>7 Visualization</td>
<td>356</td>
</tr>
<tr>
<td>8 Templates and template groups</td>
<td>383</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>8. Service monitoring</td>
<td>477</td>
</tr>
<tr>
<td>1. Service tree</td>
<td>478</td>
</tr>
<tr>
<td>2. Service actions</td>
<td>482</td>
</tr>
<tr>
<td>3. SLA</td>
<td>482</td>
</tr>
<tr>
<td>4. Setup example</td>
<td>484</td>
</tr>
<tr>
<td>9. Web monitoring</td>
<td>489</td>
</tr>
<tr>
<td>1. Web monitoring items</td>
<td>496</td>
</tr>
<tr>
<td>2. Real life scenario</td>
<td>498</td>
</tr>
<tr>
<td>10. Virtual machine monitoring</td>
<td>506</td>
</tr>
<tr>
<td>VMware monitoring item keys</td>
<td>510</td>
</tr>
<tr>
<td>Virtual machine discovery key fields</td>
<td>524</td>
</tr>
<tr>
<td>JSON examples for VMware items</td>
<td>526</td>
</tr>
<tr>
<td>11. Maintenance</td>
<td>530</td>
</tr>
<tr>
<td>12. Regular expressions</td>
<td>535</td>
</tr>
<tr>
<td>13. Problem acknowledgment</td>
<td>540</td>
</tr>
<tr>
<td>1. Problem suppression</td>
<td>542</td>
</tr>
<tr>
<td>14. Configuration export/import</td>
<td>543</td>
</tr>
<tr>
<td>1. Template groups</td>
<td>545</td>
</tr>
<tr>
<td>2. Host groups</td>
<td>545</td>
</tr>
<tr>
<td>3. Templates</td>
<td>545</td>
</tr>
<tr>
<td>4. Hosts</td>
<td>566</td>
</tr>
<tr>
<td>5. Network maps</td>
<td>584</td>
</tr>
<tr>
<td>6. Media types</td>
<td>591</td>
</tr>
<tr>
<td>15. Discovery</td>
<td>598</td>
</tr>
<tr>
<td>1. Network discovery</td>
<td>598</td>
</tr>
<tr>
<td>2. Active agent autoregistration</td>
<td>606</td>
</tr>
<tr>
<td>3. Low-level discovery</td>
<td>609</td>
</tr>
<tr>
<td>16. Distributed monitoring</td>
<td>661</td>
</tr>
<tr>
<td>1. Proxies</td>
<td>661</td>
</tr>
<tr>
<td>17. Encryption</td>
<td>664</td>
</tr>
<tr>
<td>1. Using certificates</td>
<td>671</td>
</tr>
<tr>
<td>2. Using pre-shared keys</td>
<td>677</td>
</tr>
<tr>
<td>3. Troubleshooting</td>
<td>679</td>
</tr>
<tr>
<td>18. Web interface</td>
<td>682</td>
</tr>
<tr>
<td>1. Menu</td>
<td>682</td>
</tr>
<tr>
<td>2. Frontend sections</td>
<td>684</td>
</tr>
<tr>
<td>3. User settings</td>
<td>830</td>
</tr>
<tr>
<td>4. Global search</td>
<td>834</td>
</tr>
<tr>
<td>5. Frontend maintenance mode</td>
<td>836</td>
</tr>
<tr>
<td>6. Page parameters</td>
<td>837</td>
</tr>
<tr>
<td>7. Definitions</td>
<td>838</td>
</tr>
<tr>
<td>8. Creating your own theme</td>
<td>838</td>
</tr>
<tr>
<td>9. Debug mode</td>
<td>839</td>
</tr>
<tr>
<td>10. Cookies used by Zabbix</td>
<td>839</td>
</tr>
<tr>
<td>11. Time zones</td>
<td>840</td>
</tr>
<tr>
<td>12. Rebranding</td>
<td>841</td>
</tr>
<tr>
<td>19. API</td>
<td>842</td>
</tr>
<tr>
<td>Method reference</td>
<td>847</td>
</tr>
<tr>
<td>Appendix 1. Reference commentary</td>
<td>1342</td>
</tr>
<tr>
<td>Appendix 2. Changes from 6.0 to 6.2</td>
<td>1348</td>
</tr>
<tr>
<td>Zabbix API changes in 6.2</td>
<td>1351</td>
</tr>
<tr>
<td>20. Modules</td>
<td>1351</td>
</tr>
<tr>
<td>21. Appendixes</td>
<td>1357</td>
</tr>
<tr>
<td>1. Frequently asked questions / Troubleshooting</td>
<td>1357</td>
</tr>
<tr>
<td>2. Installation and setup</td>
<td>1358</td>
</tr>
<tr>
<td>3. Process configuration</td>
<td>1395</td>
</tr>
<tr>
<td>4. Protocols</td>
<td>1444</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>Zabbix manpages</td>
<td>1559</td>
</tr>
<tr>
<td>zabbix_agent2</td>
<td>1559</td>
</tr>
<tr>
<td>NAME</td>
<td>1559</td>
</tr>
<tr>
<td>SYNOPSIS</td>
<td>1559</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>1559</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>1560</td>
</tr>
<tr>
<td>FILES</td>
<td>1560</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>1560</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>1561</td>
</tr>
<tr>
<td>Index</td>
<td>1561</td>
</tr>
<tr>
<td>zabbix_agentd</td>
<td>1561</td>
</tr>
<tr>
<td>NAME</td>
<td>1561</td>
</tr>
<tr>
<td>SYNOPSIS</td>
<td>1561</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>1561</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>1561</td>
</tr>
<tr>
<td>FILES</td>
<td>1562</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>1563</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>1563</td>
</tr>
<tr>
<td>Index</td>
<td>1563</td>
</tr>
<tr>
<td>zabbix_get</td>
<td>1563</td>
</tr>
<tr>
<td>NAME</td>
<td>1563</td>
</tr>
<tr>
<td>SYNOPSIS</td>
<td>1563</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>1563</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>1564</td>
</tr>
<tr>
<td>EXAMPLES</td>
<td>1565</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>1565</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>1565</td>
</tr>
<tr>
<td>Index</td>
<td>1565</td>
</tr>
<tr>
<td>zabbix_js</td>
<td>1565</td>
</tr>
<tr>
<td>NAME</td>
<td>1565</td>
</tr>
<tr>
<td>SYNOPSIS</td>
<td>1565</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>1565</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>1566</td>
</tr>
<tr>
<td>EXAMPLES</td>
<td>1566</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>1566</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>1566</td>
</tr>
<tr>
<td>Index</td>
<td>1566</td>
</tr>
<tr>
<td>zabbix_proxy</td>
<td>1566</td>
</tr>
<tr>
<td>NAME</td>
<td>1566</td>
</tr>
<tr>
<td>SYNOPSIS</td>
<td>1567</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>1567</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>1567</td>
</tr>
<tr>
<td>FILES</td>
<td>1568</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>1568</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>1568</td>
</tr>
<tr>
<td>Index</td>
<td>1568</td>
</tr>
<tr>
<td>zabbix_sender</td>
<td>1568</td>
</tr>
<tr>
<td>NAME</td>
<td>1568</td>
</tr>
<tr>
<td>SYNOPSIS</td>
<td>1568</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>1569</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>1569</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
</tr>
<tr>
<td>EXIT STATUS</td>
<td>1571</td>
</tr>
<tr>
<td>EXAMPLES</td>
<td>1571</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>1571</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>1572</td>
</tr>
<tr>
<td>Index</td>
<td>1572</td>
</tr>
<tr>
<td>zabbix_server</td>
<td>1572</td>
</tr>
<tr>
<td>NAME</td>
<td>1572</td>
</tr>
<tr>
<td>SYNOPSIS</td>
<td>1573</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>1573</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>1573</td>
</tr>
<tr>
<td>FILES</td>
<td>1574</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>1574</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>1574</td>
</tr>
<tr>
<td>Index</td>
<td>1574</td>
</tr>
<tr>
<td>zabbix_web_service</td>
<td>1574</td>
</tr>
<tr>
<td>NAME</td>
<td>1574</td>
</tr>
<tr>
<td>SYNOPSIS</td>
<td>1574</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>1574</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>1575</td>
</tr>
<tr>
<td>FILES</td>
<td>1575</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>1575</td>
</tr>
<tr>
<td>AUTHOR</td>
<td>1575</td>
</tr>
<tr>
<td>Index</td>
<td>1575</td>
</tr>
</tbody>
</table>
Zabbix Manual

Welcome to the user manual for Zabbix software. These pages are created to help users successfully manage their monitoring tasks with Zabbix, from the simple to the more complex ones.

Copyright notice

Zabbix documentation is NOT distributed under a GPL license. Use of Zabbix documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Zabbix disseminates it (that is, electronically for download on a Zabbix web site) or on a USB or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Zabbix. Zabbix reserves any and all rights to this documentation not expressly granted above.

1. Introduction

Please use the sidebar to access content in the Introduction section.

1 Manual structure

Structure

The content of this manual is divided into sections and subsections to provide easy access to particular subjects of interest.

When you navigate to respective sections, make sure that you expand section folders to reveal full content of what is included in subsections and individual pages.

Cross-linking between pages of related content is provided as much as possible to make sure that relevant information is not missed by the users.

Sections

Introduction provides general information about current Zabbix software. Reading this section should equip you with some good reasons to choose Zabbix.

Zabbix concepts explain the terminology used in Zabbix and provides details on Zabbix components.

Installation and Quickstart sections should help you to get started with Zabbix. Zabbix appliance is an alternative for getting a quick taster of what it is like to use Zabbix.

Configuration is one of the largest and more important sections in this manual. It contains loads of essential advice about how to set up Zabbix to monitor your environment, from setting up hosts to getting essential data to viewing data to configuring notifications and remote commands to be executed in case of problems.

IT services section details how to use Zabbix for a high-level overview of your monitoring environment.

Web monitoring should help you learn how to monitor the availability of web sites.

Virtual machine monitoring presents a how-to for configuring VMware environment monitoring.

Maintenance, Regular expressions, Event acknowledgment and XML export/import are further sections that reveal how to use these various aspects of Zabbix software.

Discovery contains instructions for setting up automatic discovery of network devices, active agents, file systems, network interfaces, etc.

Distributed monitoring deals with the possibilities of using Zabbix in larger and more complex environments.

Encryption helps explaining the possibilities of encrypting communications between Zabbix components.

Web interface contains information specific for using the web interface of Zabbix.

API section presents details of working with Zabbix API.

Detailed lists of technical information are included in Appendixes. This is where you will also find a FAQ section.
2 What is Zabbix

Overview

Zabbix was created by Alexei Vladishev, and currently is actively developed and supported by Zabbix SIA.

Zabbix is an enterprise-class open source distributed monitoring solution.

Zabbix is a software that monitors numerous parameters of a network and the health and integrity of servers, virtual machines, applications, services, databases, websites, the cloud and more. Zabbix uses a flexible notification mechanism that allows users to configure e-mail based alerts for virtually any event. This allows a fast reaction to server problems. Zabbix offers excellent reporting and data visualization features based on the stored data. This makes Zabbix ideal for capacity planning.

Zabbix supports both polling and trapping. All Zabbix reports and statistics, as well as configuration parameters, are accessed through a web-based frontend. A web-based frontend ensures that the status of your network and the health of your servers can be assessed from any location. Properly configured, Zabbix can play an important role in monitoring IT infrastructure. This is equally true for small organizations with a few servers and for large companies with a multitude of servers.

Zabbix is free of cost. Zabbix is written and distributed under the GPL General Public License version 2. It means that its source code is freely distributed and available for the general public.

Commercial support is available and provided by Zabbix Company and its partners around the world.

Learn more about Zabbix features.

Users of Zabbix

Many organizations of different size around the world rely on Zabbix as a primary monitoring platform.

3 Zabbix features

Overview

Zabbix is a highly integrated network monitoring solution, offering a multiplicity of features in a single package.

Data gathering

- availability and performance checks
- support for SNMP (both trapping and polling), IPMI, JMX, VMware monitoring
- custom checks
- gathering desired data at custom intervals
- performed by server/proxy and by agents

Flexible threshold definitions

- you can define very flexible problem thresholds, called triggers, referencing values from the backend database

Highly configurable alerting

- sending notifications can be customized for the escalation schedule, recipient, media type
- notifications can be made meaningful and helpful using macro variables
- automatic actions include remote commands

Real-time graphing

- monitored items are immediately graphed using the built-in graphing functionality

Web monitoring capabilities

- Zabbix can follow a path of simulated mouse clicks on a website and check for functionality and response time

Extensive visualization options

- ability to create custom graphs that can combine multiple items into a single view
- network maps
- slideshows in a dashboard-style overview
- reports
- high-level (business) view of monitored resources

Historical data storage
• data stored in a database
• configurable history
• built-in housekeeping procedure

Easy configuration
• add monitored devices as hosts
• hosts are picked up for monitoring, once in the database
• apply templates to monitored devices

Use of templates
• grouping checks in templates
• templates can inherit other templates

Network discovery
• automatic discovery of network devices
• agent autoregistration
• discovery of file systems, network interfaces and SNMP OIDs

Fast web interface
• a web-based frontend in PHP
• accessible from anywhere
• you can click your way through
• audit log

Zabbix API
• Zabbix API provides programmable interface to Zabbix for mass manipulations, 3rd party software integration and other purposes.

Permissions system
• secure user authentication
• certain users can be limited to certain views

Full featured and easily extensible agent
• deployed on monitoring targets
• can be deployed on both Linux and Windows

Binary daemons
• written in C, for performance and small memory footprint
• easily portable

Ready for complex environments
• remote monitoring made easy by using a Zabbix proxy

4 Zabbix overview

Architecture
Zabbix consists of several major software components. Their responsibilities are outlined below.

Server
Zabbix server is the central component to which agents report availability and integrity information and statistics. The server is the central repository in which all configuration, statistical and operational data are stored.

Database storage
All configuration information as well as the data gathered by Zabbix is stored in a database.

Web interface
For an easy access to Zabbix from anywhere and from any platform, the web-based interface is provided. The interface is part of Zabbix server, and usually (but not necessarily) runs on the same physical machine as the one running the server.

Proxy
Zabbix proxy can collect performance and availability data on behalf of Zabbix server. A proxy is an optional part of Zabbix deployment; however, it may be very beneficial to distribute the load of a single Zabbix server.

Agent

Zabbix agents are deployed on monitoring targets to actively monitor local resources and applications and report the gathered data to Zabbix server. Since Zabbix 4.4, there are two types of agents available: the Zabbix agent (lightweight, supported on many platforms, written in C) and the Zabbix agent 2 (extra-flexible, easily extendable with plugins, written in Go).

Data flow

In addition it is important to take a step back and have a look at the overall data flow within Zabbix. In order to create an item that gathers data you must first create a host. Moving to the other end of the Zabbix spectrum you must first have an item to create a trigger. You must have a trigger to create an action. Thus if you want to receive an alert that your CPU load is too high on Server X you must first create a host entry for Server X followed by an item for monitoring its CPU, then a trigger which activates if the CPU is too high, followed by an action which sends you an email. While that may seem like a lot of steps, with the use of templating it really isn’t. However, due to this design it is possible to create a very flexible setup.

5 What’s new in Zabbix 6.2.0

**Single problem suppression** In the new version you may prioritize the problem list by hiding those issues that can be dealt with at a later time. This is done by suppressing single problems for a defined time period.

A problem can be suppressed through the problem update window that is opened when acknowledging a problem:

For more details see [Problem suppression](#).

**Multiple LDAP sources** It is now possible to define multiple LDAP servers for **LDAP authentication**. This is useful when different LDAP servers are used to authenticate different groups of users.

Once the servers are configured in Zabbix, it becomes possible to select the required LDAP server for the respective user group, in **user group** configuration.

**Storage of secrets** It is now possible to store some sensitive information from Zabbix in CyberArk Vault CV2. Similarly to storing secrets in HashiCorp Vault, introduced in Zabbix 5.2, CyberArk Vault can be used for:

- user macro values
• database access credentials

Zabbix provides read-only access to the secrets in vault.

See also: CyberArk configuration

Secure password hashing  In Zabbix 5.0 the password hashing algorithm was changed from MD5 to the more secure bcrypt. However, MD5 cryptography remained supported to ensure smooth upgrades from previous versions. MD5 hashing was only used for some users upon the first login after an upgrade - to convert passwords with not reliable hashes from MD5 to bcrypt. Now support of MD5 cryptography has been dropped completely.

Reload proxy configuration  It is now possible to force the reload of configuration for proxies from the server. It can be done in two ways:

  • by Zabbix server runtime control command (e.g. `zabbix_server -R proxy_config_cache_reload`)
  • from the frontend (in the list of proxies or the proxy editing form)

It is also now possible for passive proxies to request configuration from the server using the `config_cache_reload proxy runtime control` command.

Separate groups for templates  Previously, host groups were used to organize both hosts and templates. Now this functionality has been split into template groups, which may contain templates only, and host groups, which may contain hosts only.

A new subsection Template groups has been added to the Configuration menu.

User role and user group permissions are now defined separately for host groups and template groups.

AWS EC2 monitoring  A new template AWS EC2 by HTTP has been added allowing to quickly deploy Zabbix monitoring of AWS EC2 and attached AWS EBS volumes by HTTP.

You can get this template:

  • In Configuration → Templates in new installations;
  • When upgrading from previous versions, the latest templates can be downloaded from the Zabbix Git repository and manually imported into Zabbix in the Configuration → Templates section. If a template with the same name already exists, check the Delete missing option before importing to achieve a clean import. This way the items that have been excluded from the updated template will be removed (note, that history of the deleted items will be lost).

Configuration cache  Faster configuration sync

Incremental configuration cache synchronization has been added for hosts, host tags, items, item tags, item preprocessing, triggers, trigger tags and functions to lessen synchronization time and database load when configuration is being updated on an already running Zabbix server or Zabbix proxy.

User macro cache

To reduce configuration cache locking and, therefore, improve performance, user macro values are now stored in a separate user macro cache instead of the configuration cache.

Items  Immediate checks for new items

Previously, newly added items were first checked at a random time within their update interval. Now new items and discovery rules will be checked within 60 seconds of their creation, unless they have Scheduling or Flexible update interval with the Update interval parameter set to 0.

Windows registry monitoring

Windows registry monitoring is now supported out-of-the-box in Zabbix. Two new keys have been added to the Windows Zabbix agent and agent 2:
• `registry.data[key,<value name>]` - return data for the specified value name in the Windows Registry key
• `registry.get[key,<mode>,<name regexp>]` - list of Windows Registry values or keys located at given key; returns JSON

See also: Windows Zabbix agent items

Low-level discovery of OS processes

Zabbix now provides a native solution to discover running OS processes. A new item `proc.get[]` can be used in discovery rules to return a list of running processes/threads or summarized data grouped by process name.

See also:
• Zabbix agent item keys
• Process parameters returned by `proc.get` item

Extended VMware monitoring

Multiple new items are now available for discovering and monitoring VMware:

- vSphere Distributed Switch ports: `vmware.dvswitch.discovery[]`, `vmware.dvswitch.fetchports.get[]`
- Virtual machines: `vmware.vm.state[]`, `vmware.vm.tools[]`, `vmware.vm.snapshot.get[]`, `vmware.vm.consolidationneeded[]`,
  `vmware.vm.attribute[]`
- Hypervisors: `vmware hv.connectionstate[]`, `vmware hv.hw.serialnumber[]`, `vmware hv.hw.sensors.get[]`,
  `vmware hv.net.if.discovery[]`, `vmware hv.network.linkspeed[]`
- Resource pools: `vmware rp.cpu.usage[]`, `vmware rp.memory[]`

The items `vmware hv.network.in[]` and `vmware hv.network.out[]`, used for monitoring hypervisor network traffic, now support additional NIC counters. Four new mode options have been added to both items: packets, dropped, errors, broadcast.

The item `vmware vm.discovery[]`, used for virtual machine discovery, now returns additional discovery fields, including user-defined custom attribute values.

The items `vmware hv.discovery[]`, used for hypervisor discovery, and `vmware cluster.discovery[]`, used for clusters discovery, now return information about resource pools.

Active checks affect host availability   Active Zabbix agent items now also affect host availability as seen in Monitoring -> Hosts or Configuration -> Hosts.

To determine active check availability heartbeat messages are now sent in the active check thread. The frequency of the heartbeat messages is set by the new `HeartbeatFrequency` parameter in Zabbix agent and agent 2 configurations (60 seconds by default, 0-3600 range). Active checks are considered unavailable when the active check heartbeat is older than 2 x HeartbeatFrequency seconds.

This functionality will only work if the latest version of Zabbix agent or Zabbix agent 2 is used. Agents of older versions are not sending any heartbeats, so the availability of their hosts will remain unknown.

Active agent availability is counted towards the total agent availability in the same way as a passive interface is, for example:

- If a passive agent interface is available, and active checks are available - the total availability is green (all available)
- If a passive agent interface is available, but active checks are not available - the total availability is yellow (mixed)
- If a passive agent interface is available, but active checks are unknown - the total availability is gray (unknown)

Additionally, active agent availability is listed, as a separate row, in the popup list of available agent interfaces. This popup is opened when hovering over the host availability icon for the Zabbix agent interface in Monitoring -> Hosts or Configuration -> Hosts.
A new zabbix[host,active_agent,available] internal item allows to monitor the availability of active checks.

**Discovered host customization**  For hosts, created from host prototypes, the following parameters can now be modified after discovery:

- Tags
- Macros
- Templates

It is possible to link additional templates and add more tags and macros as well as update or remove existing tags and macros. Templates inherited from a host prototype cannot be unlinked from a discovered host.

**Graph widget**  The vector graph widget has been improved and it is now possible to:

- display stacked graph (with filled areas)
- add a list of items
- clone a data set
- show lines for minimum, maximum and average values
- show the trigger line for simple triggers
- show percentiles
- show working time
- show minimum, maximum and average item values in the legend
- have columns in the legend

**Frontend**  Minimum required PHP version

The minimum required PHP version has been raised from 7.2.5 to 7.4.0.

**Default dashboard updated**

The “Global view” default dashboard for new Zabbix installations has been updated to include the latest dashboard widgets.

**Digital clock widget**

The Clock dashboard widget has been updated to display a digital clock as well.
Direct links to documentation

All frontend forms now have direct links to the corresponding parts of the documentation.

This is implemented by adding a help link to the frontend form headers:

Retrieve latest item value in latest data

The option to retrieve the latest item value immediately is now available in the latest data page, both as an “Execute now” button below the list of items and as an option in the item menu when clicking on the item name.

In previous versions the same functionality was available in the Configuration section (item/discovery rule form and lists) only.

As another improvement, it is now also possible to “Execute now” dependent items.

The option to “execute now” depends on host permissions and user role settings; for more details see:

- Latest data
- Execute now

Filter settings remembered

In several Monitoring pages (Problems, Hosts, Latest data) current filter settings are now remembered in the user profile. When the user opens the page again, the filter settings will have stayed the same.

Additionally, the marking of a changed (but not saved) favorite filter is now a green dot next to the filter name, instead of the filter name in italics.

Miscellaneous

- The form for API token creation and editing is now opened in a modal (popup) window.
- Locale “British English” (en_GB) is available again in Zabbix frontend.
- German and Vietnamese languages are now enabled.

Macros

- {INVENTORY{.*}} macros are now supported in script-type items and manual host action scripts for Zabbix server and Zabbix proxy.

HMAC function for JavaScript  A new function has been added to the JavaScript engine allowing to return HMAC hash:

- hmac(‘<hash type>’,key,string)

This is useful for cases when hash-based message authentication code (HMAC) is required for signing requests. MD5 and SHA256 hash types are supported, e. g.:

- hmac(‘md5’,key,string)
- hmac(‘sha256’,key,string)
**Breaking changes**  
PHP version

PHP versions below 7.4.0 are no longer supported. The minimum required PHP version has been raised from 7.2.5 to 7.4.0.

---

**6 What’s new in Zabbix 6.2.1**

**MariaDB 10.8 support**  
The maximum supported version for MariaDB is now 10.8.X.

**TimescaleDB 2.6 support**  
The maximum supported version for TimescaleDB is now 2.6.

**Templates**  
A new template HPE Synergy by HTTP is available.

See setup instructions for [HTTP templates](#).

You can get this template:

- In Configuration → Templates in new installations;
- If you are upgrading from previous versions, you can download new templates from Zabbix [Git repository](#) or find them in the `templates` directory of the downloaded latest Zabbix version. Then, while in Configuration → Templates you can import them manually into Zabbix.

**Extended VMware monitoring**  
New items are now available for:

- Monitoring VMware alarms: `vmware.alarms.get[]`, `vmware.cluster.alarms.get[]`, `vmware.datastore.alarms.get[]`, `vmware.dc.alarms.get[]`, `vmware.hv.alarms.get[]`, `vmware.vm.alarms.get[]`.
- Collecting tag information from different VMware components: `vmware.cluster.tags.get[]`, `vmware.datastore.tags.get[]`, `vmware.dc.tags.get[]`, `vmware.hv.tags.get[]`, `vmware.vm.tags.get[]`.
- Returning property values: `vmware.cluster.property[]`, `vmware.datastore.property[]`, `vmware.hv.property[]`, `vmware.vm.property[]`.

Multiple VMware discovery items now also return a JSON array containing tags.

The items `vmware.datastore.discovery[]` and `vmware.hv.datastore.discovery[]` now also return `{#DATASTORE.UUID}` macro with a datacenter identifier.

---

**7 What’s new in Zabbix 6.2.2**

**Items**  
The `proc.get[]` agent item on FreeBSD now also returns the jail name in the `jname` property.

**Month abbreviated with capital letter**  
A “month” is now abbreviated with the capital “M” in the frontend. Previously it was abbreviated with the small “m”, overlapping with the abbreviation of a minute.

---

**2. Definitions**

**Overview**  
In this section you can learn the meaning of some terms commonly used in Zabbix.

**Definitions**

**host**  
a networked device that you want to monitor, with IP/DNS.

**host group**  
a logical grouping of hosts. Host groups are used when assigning access rights to hosts for different user groups.

**item**  
a particular piece of data that you want to receive off of a host, a metric of data.

**value preprocessing**
- a transformation of received metric value before saving it to the database.

**trigger**

- a logical expression that defines a problem threshold and is used to “evaluate” data received in items.

When received data are above the threshold, triggers go from ‘Ok’ into a ‘Problem’ state. When received data are below the threshold, triggers stay in/return to an ‘Ok’ state.

**event**

- a single occurrence of something that deserves attention such as a trigger changing state or a discovery/agent autoregistration taking place.

**event tag**

- a pre-defined marker for the event. It may be used in event correlation, permission granulation, etc.

**event correlation**

- a method of correlating problems to their resolution flexibly and precisely.

For example, you may define that a problem reported by one trigger may be resolved by another trigger, which may even use a different data collection method.

**problem**

- a trigger that is in “Problem” state.

**problem update**

- problem management options provided by Zabbix, such as adding comment, acknowledging, changing severity or closing manually.

**action**

- a predefined means of reacting to an event.

An action consists of operations (e.g. sending a notification) and conditions (when the operation is carried out)

**escalation**

- a custom scenario for executing operations within an action; a sequence of sending notifications/executing remote commands.

**media**

- a means of delivering notifications; delivery channel.

**notification**

- a message about some event sent to a user via the chosen media channel.

**remote command**

- a pre-defined command that is automatically executed on a monitored host upon some condition.

**template**

- a set of entities (items, triggers, graphs, low-level discovery rules, web scenarios) ready to be applied to one or several hosts.

The job of templates is to speed up the deployment of monitoring tasks on a host; also to make it easier to apply mass changes to monitoring tasks. Templates are linked directly to individual hosts.

**template group**

- a logical grouping of templates. Template groups are used when assigning access rights to templates for different user groups.

**web scenario**

- one or several HTTP requests to check the availability of a website.

**frontend**

- the web interface provided with Zabbix.

**dashboard**

- customizable section of the web interface displaying summaries and visualizations of important information in visual units called widgets.

**widget**
visual unit displaying information of a certain kind and source (a summary, a map, a graph, the clock, etc), used in the dashboard.

**Zabbix API**

- Zabbix API allows you to use the JSON RPC protocol to create, update and fetch Zabbix objects (like hosts, items, graphs and others) or perform any other custom tasks.

**Zabbix server**

- A central process of Zabbix software that performs monitoring, interacts with Zabbix proxies and agents, calculates triggers, sends notifications; a central repository of data.

**Zabbix proxy**

- A process that may collect data on behalf of Zabbix server, taking some processing load off of the server.

**Zabbix agent**

- A process deployed on monitoring targets to actively monitor local resources and applications.

**Zabbix agent 2**

- A new generation of Zabbix agent to actively monitor local resources and applications, allowing to use custom plugins for monitoring.

  Because Zabbix agent 2 shares much functionality with Zabbix agent, the term “Zabbix agent” in documentation stands for both - Zabbix agent and Zabbix agent 2, if the functional behavior is the same. Zabbix agent 2 is only specifically named where its functionality differs.

**Encryption**

- Support of encrypted communications between Zabbix components (server, proxy, agent, zabbix_sender and zabbix_get utilities) using Transport Layer Security (TLS) protocol.

**Network discovery**

- Automated discovery of network devices.

**Low-level discovery**

- Automated discovery of low-level entities on a particular device (e.g. file systems, network interfaces, etc).

**Low-level discovery rule**

- Set of definitions for automated discovery of low-level entities on a device.

**Item prototype**

- A metric with certain parameters as variables, ready for low-level discovery. After low-level discovery the variables are automatically substituted with the real discovered parameters and the metric automatically starts gathering data.

**Trigger prototype**

- A trigger with certain parameters as variables, ready for low-level discovery. After low-level discovery the variables are automatically substituted with the real discovered parameters and the trigger automatically starts evaluating data.

Prototypes of some other Zabbix entities are also in use in low-level discovery - graph prototypes, host prototypes, host group prototypes.

**Agent autoregistration**

- Automated process whereby a Zabbix agent itself is registered as a host and started to monitor.

### 3. Zabbix processes

Please use the sidebar to access content in the Zabbix process section.

#### 1 Server

Overview

Zabbix server is the central process of Zabbix software.
The server performs the polling and trapping of data, it calculates triggers, sends notifications to users. It is the central component to which Zabbix agents and proxies report data on availability and integrity of systems. The server can itself remotely check networked services (such as web servers and mail servers) using simple service checks.

The server is the central repository in which all configuration, statistical and operational data is stored, and it is the entity in Zabbix that will actively alert administrators when problems arise in any of the monitored systems.

The functioning of a basic Zabbix server is broken into three distinct components; they are: Zabbix server, web frontend and database storage.

All of the configuration information for Zabbix is stored in the database, which both the server and the web frontend interact with. For example, when you create a new item using the web frontend (or API) it is added to the items table in the database. Then, about once a minute Zabbix server will query the items table for a list of the items which are active that is then stored in a cache within the Zabbix server. This is why it can take up to two minutes for any changes made in Zabbix frontend to show up in the latest data section.

Running server

If installed as package

Zabbix server runs as a daemon process. The server can be started by executing:

shell> service zabbix-server start

This will work on most of GNU/Linux systems. On other systems you may need to run:

shell> /etc/init.d/zabbix-server start

Similarly, for stopping/restarting/viewing status, use the following commands:

shell> service zabbix-server stop
shell> service zabbix-server restart
shell> service zabbix-server status

Start up manually

If the above does not work you have to start it manually. Find the path to the zabbix_server binary and execute:

shell> zabbix_server

You can use the following command line parameters with Zabbix server:

- c --config <file> path to the configuration file (default is /usr/local/etc/zabbix_server.conf)
- f --foreground run Zabbix server in foreground
- R --runtime-control <option> perform administrative functions
- h --help give this help
- V --version display version number

Examples of running Zabbix server with command line parameters:

shell> zabbix_server -c /usr/local/etc/zabbix_server.conf
shell> zabbix_server --help
shell> zabbix_server -V

Runtime control

Runtime control options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>config_cache_reload</td>
<td>Reload configuration cache. Ignored if cache is being currently loaded.</td>
<td></td>
</tr>
<tr>
<td>diaginfo=target</td>
<td>Gather diagnostic information in the server log file.</td>
<td>historycache - history cache statistics valuecache - value cache statistics preprocessing - preprocessing manager statistics alerting - alert manager statistics lld - LLD manager statistics locks - list of mutexes (is empty on *<em>BSD</em> systems)</td>
</tr>
<tr>
<td>ha_status</td>
<td>Log high availability (HA) cluster status.</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Target</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>ha_remove_node=target</td>
<td>Remove the high availability (HA) node specified by its name or ID.</td>
<td>target - name or ID of the node (can be obtained by running ha_status)</td>
</tr>
<tr>
<td>ha_set_failover_delay=delay</td>
<td>Set high availability (HA) failover delay. Time suffixes are supported, e.g., 10s, 1m.</td>
<td></td>
</tr>
<tr>
<td>proxy_config_cache_reload</td>
<td>Reload proxy configuration cache.</td>
<td></td>
</tr>
<tr>
<td>services_cache_reload</td>
<td>Reload the service manager cache.</td>
<td></td>
</tr>
<tr>
<td>snmp_cache_reload</td>
<td>Reload SNMP cache, clear the SNMP properties (engine time, engine boots, engine id, credentials) for all hosts.</td>
<td></td>
</tr>
<tr>
<td>secrets_reload</td>
<td>Reload secrets from Vault.</td>
<td></td>
</tr>
<tr>
<td>log_level_increase=target</td>
<td>Increase log level, affects all processes if target is not specified.</td>
<td></td>
</tr>
<tr>
<td>log_level_decrease=target</td>
<td>Decrease log level, affects all processes if target is not specified.</td>
<td></td>
</tr>
</tbody>
</table>

Example of using runtime control to reload the server configuration cache:

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R config_cache_reload
```

Examples of using runtime control to reload the proxy configuration:

```
shell> zabbix_server -R proxy_config_cache_reload
```

Example of using runtime control to gather diagnostic information:

```
shell> zabbix_server -R diaginfo
```

Example of using runtime control to reload the SNMP cache:

```
shell> zabbix_server -R snmp_cache_reload
```

Example of using runtime control to trigger execution of housekeeper:

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R housekeeper_execute
```

Example of using runtime control to change log level:

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase
```

Increase log level of second poller process:

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase=poller,2
```
Increase log level of process with PID 1234:
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase=1234

Decrease log level of all http poller processes:
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_decrease="http poller"

Example of setting the HA failover delay to the minimum of 10 seconds:
shell> zabbix_server -R ha_set_failover_delay=10s

Process user

Zabbix server is designed to run as a non-root user. It will run as whatever non-root user it is started as. So you can run server as any non-root user without any issues.

If you will try to run it as ‘root’, it will switch to a hardcoded ‘zabbix’ user, which must be present on your system. You can only run server as ‘root’ if you modify the ‘AllowRoot’ parameter in the server configuration file accordingly.

If Zabbix server and agent are run on the same machine it is recommended to use a different user for running the server than for running the agent. Otherwise, if both are run as the same user, the agent can access the server configuration file and any Admin level user in Zabbix can quite easily retrieve, for example, the database password.

Configuration file

See the configuration file options for details on configuring zabbix_server.

Start-up scripts

The scripts are used to automatically start/stop Zabbix processes during system’s start-up/shutdown. The scripts are located under directory misc/init.d.

Server process types

- alert manager - alert queue manager
- alert syncer - alert DB writer
- alerter - process for sending notifications
- availability manager - process for host availability updates
- configuration syncer - process for managing in-memory cache of configuration data
- discoverer - process for discovery of devices
- escallator - process for escalation of actions
- history poller - process for handling calculated checks requiring a database connection
- history syncer - history DB writer
- housekeeper - process for removal of old historical data
- http poller - web monitoring poller
- icmp pinger - poller for icmp ping checks
- ipmi manager - IPMI poller manager
- ipmi poller - poller for IPMI checks
- java poller - poller for Java checks
- lld manager - manager process of low-level discovery tasks
- lld worker - worker process of low-level discovery tasks
- odbc poller - poller for ODBC checks
- poller - normal poller for passive checks
- preprocessing manager - manager of preprocessing tasks
- preprocessing worker - process for data preprocessing
- problem housekeeper - process for removing problems of deleted triggers
- proxy poller - poller for passive proxies
- report manager- manager of scheduled report generation tasks
- report writer - process for generating scheduled reports
- self-monitoring - process for collecting internal server statistics
- snmp trapper - trapper for SNMP traps
- task manager - process for remote execution of tasks requested by other components (e.g. close problem, acknowledge problem, check item value now, remote command functionality)
- timer - timer for processing maintenances
- trapper - trapper for active checks, traps, proxy communication
- unreachable poller - poller for unreachable devices
- vmware collector - VMware data collector responsible for data gathering from VMware services

The server log file can be used to observe these process types.

Various types of Zabbix server processes can be monitored using the `zabbix[process,<type>,<mode>,<state>]` internal item.
Supported platforms

Due to the security requirements and mission-critical nature of server operation, UNIX is the only operating system that can consistently deliver the necessary performance, fault tolerance and resilience. Zabbix operates on market leading versions.

Zabbix server is tested on the following platforms:

- Linux
- Solaris
- AIX
- HP-UX
- Mac OS X
- FreeBSD
- OpenBSD
- NetBSD
- SCO Open Server
- Tru64/OSF1

Zabbix may work on other Unix-like operating systems as well.

Locale

Note that the server requires a UTF-8 locale so that some textual items can be interpreted correctly. Most modern Unix-like systems have a UTF-8 locale as default, however, there are some systems where that may need to be set specifically.

1 High availability

Overview

High availability (HA) is typically required in critical infrastructures that can afford virtually no downtime. So for any service that may fail there must be a failover option in place to take over should the current service fail.

Zabbix offers a native high-availability solution that is easy to set up and does not require any previous HA expertise. Native Zabbix HA may be useful for an extra layer of protection against software/hardware failures of Zabbix server or to have less downtime due to maintenance.

In the Zabbix high availability mode multiple Zabbix servers are run as nodes in a cluster. While one Zabbix server in the cluster is active, others are on standby, ready to take over if necessary.

Switching to Zabbix HA is non-committal. You may switch back to standalone operation at any point.

See also: Implementation details

Enabling high availability

Starting Zabbix server as cluster node

Two parameters are required in the server configuration to start a Zabbix server as cluster node:

- **HANodeName** parameter must be specified for each Zabbix server that will be an HA cluster node.

This is the name (e.g. zabbix-x-node-01) that the server will be referred to in agent and proxy configurations. If you do not specify HANodeName, then the server will be started in standalone mode.
• **NodeAddress** parameter must be specified for each node.

The NodeAddress parameter (address:port) will be used by Zabbix frontend to connect to the active server node. NodeAddress must match the IP or FQDN name of the respective Zabbix server.

Restart all Zabbix servers after making changes to the configuration files. They will now be started as cluster nodes. The new status of the servers can be seen in Reports → System information and also by running:

```
zabbix_server -R ha_status
```

This runtime command will log the current HA cluster status into the Zabbix server log:

```
<table>
<thead>
<tr>
<th>ID</th>
<th>Node</th>
<th>Address</th>
<th>Status</th>
<th>Last Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>zabbix-node-01</td>
<td>64.227.66.193:10051</td>
<td>standby</td>
<td>0s</td>
</tr>
<tr>
<td>2</td>
<td>zabbix-node-02</td>
<td>64.227.74.25:10051</td>
<td>active</td>
<td>3s</td>
</tr>
</tbody>
</table>
```

### Preparing frontend

Make sure that Zabbix server address:port is **not defined** in the frontend configuration (found in `conf/zabbix.conf.php` of the frontend files directory).

```
// Uncomment and set to desired values to override Zabbix hostname/IP and port.
// $ZBX_SERVER = '';
// $ZBX_SERVER_PORT = '';
```

Zabbix frontend will autodetect the active node by reading settings from the nodes table in Zabbix database. Node address of the active node will be used as the Zabbix server address.

### Proxy configuration

HA cluster nodes (servers) must be listed in the configuration of either passive or active Zabbix proxy.

For a passive proxy, the node names must be listed in the Server **parameter** of the proxy, separated by a **comma**.

```
Server=zabbix-node-01,zabbix-node-02
```

For an active proxy, the node names must be listed in the Server **parameter** of the proxy, separated by a **semicolon**.

```
ServerActive=zabbix-node-01;zabbix-node-02
```

### Agent configuration

HA cluster nodes (servers) must be listed in the configuration of Zabbix agent or Zabbix agent 2.

To enable passive checks, the node names must be listed in the Server **parameter**, separated by a **comma**.

```
Server=zabbix-node-01,zabbix-node-02
```

To enable active checks, the node names must be listed in the ServerActive **parameter**. Note that for active checks the nodes must be separated by a comma from any other servers, while the nodes themselves must be separated by a **semicolon**, e.g.:

```
ServerActive=zabbix-node-01;zabbix-node-02
```
Failover to standby node

Zabbix will fail over to another node automatically if the active node stops. There must be at least one node in standby status for the failover to happen.

How fast will the failover be? All nodes update their last access time (and status, if it is changed) every 5 seconds. So:

- If the active node shuts down and manages to report its status as "stopped", another node will take over within **5 seconds**.
- If the active node shuts down/becomes unavailable without being able to update its status, standby nodes will wait for the **failover delay + 5 seconds** to take over.

The failover delay is configurable, with the supported range between 10 seconds and 15 minutes (one minute by default). To change the failover delay, you may run:

```
zabbix_server -R ha_set_failover_delay=5m
```

Managing HA cluster

The current status of the HA cluster can be managed using the dedicated runtime control options:

```bash
ha_status - log HA cluster status in the Zabbix server log
ha_remove_node=target - remove an HA node identified by its <target> - number of the node in the list (the number can be obtained from the output of running ha_status), e.g.:
```

```
zabbix_server -R ha_remove_node=2
```

Note that active/standby nodes cannot be removed.

- **ha_set_failover_delay=delay** - set HA failover delay (between 10 seconds and 15 minutes; time suffixes are supported, e.g. 10s, 1m)

Node status can be monitored:

- in Reports → **System information**
- in the System information dashboard widget
- using the **ha_status** runtime control option of the server (see above).

The `zabbix[cluster,discovery,nodes]` internal item can be used for node discovery, as it returns a JSON with the high-availability node information.

Disabling HA cluster

To disable a high availability cluster:

- make backup copies of configuration files
- stop standby nodes
- remove the HANodeName parameter from the active primary server
- restart the primary server (it will start in standalone mode)

Implementation details

The high availability (HA) cluster is an opt-in solution and it is supported for Zabbix server. The native HA solution is designed to be simple in use, it will work across sites and does not have specific requirements for the databases that Zabbix recognizes. Users are free to use the native Zabbix HA solution, or a third party HA solution, depending on what best suits the high availability requirements in their environment.

The solution consists of multiple `zabbix_server` instances or nodes. Every node:

- is configured separately
- uses the same database
- may have several modes: active, standby, unavailable, stopped

Only one node can be active (working) at a time. A standby node runs only one process - the HA manager. A standby node does no data collection, processing or other regular server activities; they do not listen on ports; they have minimum database connections.

Both active and standby nodes update their last access time every 5 seconds. Each standby node monitors the last access time of the active node. If the last access time of the active node is over ‘failover delay’ seconds, the standby node switches itself to be the active node and assigns ‘unavailable’ status to the previously active node.

The active node monitors its own database connectivity - if it is lost for more than **failover delay** – 5 seconds, it must stop all processing and switch to standby mode. The active node also monitors the status of the standby nodes - if the last access time of a standby node is over ‘failover delay’ seconds, the standby node is assigned the ‘unavailable’ status.

The nodes are designed to be compatible across minor Zabbix versions.
Overview

Zabbix agent is deployed on a monitoring target to actively monitor local resources and applications (hard drives, memory, processor statistics etc).

The agent gathers operational information locally and reports data to Zabbix server for further processing. In case of failures (such as a hard disk running full or a crashed service process), Zabbix server can actively alert the administrators of the particular machine that reported the failure.

Zabbix agents are extremely efficient because of use of native system calls for gathering statistical information.

Passive and active checks

Zabbix agents can perform passive and active checks.

In a passive check the agent responds to a data request. Zabbix server (or proxy) asks for data, for example, CPU load, and Zabbix agent sends back the result.

Active checks require more complex processing. The agent must first retrieve a list of items from Zabbix server for independent processing. Then it will periodically send new values to the server.

Whether to perform passive or active checks is configured by selecting the respective monitoring item type. Zabbix agent processes items of type ‘Zabbix agent’ or ‘Zabbix agent (active)’.

Supported platforms

Zabbix agent is supported for:

- Linux
- IBM AIX
- FreeBSD
- NetBSD
- OpenBSD
- HP-UX
- Mac OS X
- Solaris: 9, 10, 11
- Windows: all desktop and server versions since XP

Agent on UNIX-like systems

Zabbix agent on UNIX-like systems is run on the host being monitored.

Installation

See the package installation section for instructions on how to install Zabbix agent as package.

Alternatively see instructions for manual installation if you do not want to use packages.

In general, 32bit Zabbix agents will work on 64bit systems, but may fail in some cases.

If installed as package

Zabbix agent runs as a daemon process. The agent can be started by executing:

shell> service zabbix-agent start

This will work on most of GNU/Linux systems. On other systems you may need to run:

shell> /etc/init.d/zabbix-agent start

Similarly, for stopping/restarting/viewing status of Zabbix agent, use the following commands:

shell> service zabbix-agent stop
shell> service zabbix-agent restart
shell> service zabbix-agent status

Start up manually

If the above does not work you have to start it manually. Find the path to the zabbix_agentd binary and execute:

shell> zabbix_agentd
Agent on Windows systems

Zabbix agent on Windows runs as a Windows service.

Preparation

Zabbix agent is distributed as a zip archive. After you download the archive you need to unpack it. Choose any folder to store Zabbix agent and the configuration file, e.g.

C:\zabbix

Copy bin\zabbix_agentd.exe and conf\zabbix_agentd.conf files to c:\zabbix.

Edit the c:\zabbix\zabbix_agentd.conf file to your needs, making sure to specify a correct “Hostname” parameter.

Installation

After this is done use the following command to install Zabbix agent as Windows service:

C:\> c:\zabbix\zabbix_agentd.exe -c c:\zabbix\zabbix_agentd.conf -i

Now you should be able to configure “Zabbix agent” service normally as any other Windows service.

See more details on installing and running Zabbix agent on Windows.

Other agent options

It is possible to run multiple instances of the agent on a host. A single instance can use the default configuration file or a configuration file specified in the command line. In case of multiple instances each agent instance must have its own configuration file (one of the instances can use the default configuration file).

The following command line parameters can be used with Zabbix agent:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX and Windows agent</td>
<td>Path to the configuration file. You may use this option to specify a configuration file that is not the default one. On UNIX, default is /usr/local/etc/zabbix_agentd.conf or as set by compile-time variables --sysconfdir or --prefix. On Windows, default is c:\zabbix_agentd.conf.</td>
</tr>
<tr>
<td>-c --config &lt;config-file&gt;</td>
<td>Print known items and exit. Note: To return user parameter results as well, you must specify the configuration file (if it is not in the default location).</td>
</tr>
<tr>
<td>-p --print</td>
<td>Test specified item and exit. Note: To return user parameter results as well, you must specify the configuration file (if it is not in the default location).</td>
</tr>
<tr>
<td>-t --test &lt;item key&gt;</td>
<td>Display help information</td>
</tr>
<tr>
<td>-h --help</td>
<td>Display version number</td>
</tr>
<tr>
<td>UNIX agent only</td>
<td>Perform administrative functions. See runtime control.</td>
</tr>
<tr>
<td>-R --runtime-control &lt;option&gt;</td>
<td>Windows agent only</td>
</tr>
<tr>
<td>Windows agent only (functions)</td>
<td>Install Zabbix Windows agent as service</td>
</tr>
<tr>
<td>-i --install</td>
<td>Uninstall Zabbix Windows agent service</td>
</tr>
<tr>
<td>-d --uninstall</td>
<td>Start Zabbix Windows agent service</td>
</tr>
<tr>
<td>-s --start</td>
<td>Stop Zabbix Windows agent service</td>
</tr>
<tr>
<td>-x --stop</td>
<td>Specific examples of using command line parameters:</td>
</tr>
<tr>
<td>-</td>
<td>printing all built-in agent items with values</td>
</tr>
<tr>
<td>-</td>
<td>testing a user parameter with “mysql.ping” key defined in the specified configuration file</td>
</tr>
<tr>
<td>-</td>
<td>installing a “Zabbix Agent” service for Windows using the default path to configuration file c:\zabbix_agentd.conf</td>
</tr>
</tbody>
</table>
• installing a "Zabbix Agent [Hostname]" service for Windows using the configuration file zabbix_agentd.conf located in the same folder as agent executable and make the service name unique by extending it by Hostname value from the config file

shell> zabbix_agentd --print
shell> zabbix_agentd -t "mysql.ping" -c /etc/zabbix/zabbix_agentd.conf
shell> zabbix_agentd.exe -i
shell> zabbix_agentd.exe -i -m -c zabbix_agentd.conf

Runtime control

With runtime control options you may change the log level of agent processes.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>log_level_decrease[=target]</td>
<td>Decrease log level.</td>
<td>Target can be specified as:</td>
</tr>
<tr>
<td></td>
<td>If target is not specified, all processes are</td>
<td>process type - all processes of specified type</td>
</tr>
<tr>
<td></td>
<td>affected.</td>
<td>(e.g., listener)</td>
</tr>
<tr>
<td></td>
<td>See all agent process types.</td>
<td>process type,N - process type and number (e.g.,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>listener,3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pid - process identifier (1 to 65535). For larger</td>
</tr>
<tr>
<td></td>
<td></td>
<td>values specify target as 'process-type,N'.</td>
</tr>
<tr>
<td>userparameter_reload</td>
<td>Reload user parameters from the current</td>
<td></td>
</tr>
<tr>
<td></td>
<td>configuration file.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note that UserParameter is the only agent</td>
<td></td>
</tr>
<tr>
<td></td>
<td>configuration option that will be reloaded.</td>
<td></td>
</tr>
</tbody>
</table>

Examples:

• increasing log level of all processes
• increasing log level of the third listener process
• increasing log level of process with PID 1234
• decreasing log level of all active check processes

shell> zabbix_agentd -R log_level_increase
shell> zabbix_agentd -R log_level_increase=listener,3
shell> zabbix_agentd -R log_level_increase=1234
shell> zabbix_agentd -R log_level_decrease="active checks"

Runtime control is not supported on OpenBSD, NetBSD and Windows.

Agent process types

• active checks - process for performing active checks
• collector - process for data collection
• listener - process for listening to passive checks

The agent log file can be used to observe these process types.

Process user

Zabbix agent on UNIX is designed to run as a non-root user. It will run as whatever non-root user it is started as. So you can run agent as any non-root user without any issues.

If you will try to run it as 'root', it will switch to a hardcoded 'zabbix' user, which must be present on your system. You can only run agent as 'root' if you modify the 'AllowRoot' parameter in the agent configuration file accordingly.

Configuration file

For details on configuring Zabbix agent see the configuration file options for zabbix_agentd or Windows agent.

Locale

Note that the agent requires a UTF-8 locale so that some textual agent items can return the expected content. Most modern Unix-like systems have a UTF-8 locale as default, however, there are some systems where that may need to be set specifically.

Exit code
Before version 2.2 Zabbix agent returned 0 in case of successful exit and 255 in case of failure. Starting from version 2.2 and higher Zabbix agent returns 0 in case of successful exit and 1 in case of failure.

3 Agent 2

Overview

Zabbix agent 2 is a new generation of Zabbix agent and may be used in place of Zabbix agent. Zabbix agent 2 has been developed to:

- reduce the number of TCP connections
- provide improved concurrency of checks
- be easily extendible with plugins. A plugin should be able to:
  - provide trivial checks consisting of only a few simple lines of code
  - provide complex checks consisting of long-running scripts and standalone data gathering with periodic sending back of the data
- be a drop-in replacement for Zabbix agent (in that it supports all the previous functionality)

Agent 2 is written in Go (with some C code of Zabbix agent reused). A configured Go environment with a currently supported Go version is required for building Zabbix agent 2.

Agnet 2 does not have built-in daemonization support on Linux; it can be run as a Windows service.

Passive checks work similarly to Zabbix agent. Active checks support scheduled/flexible intervals and check concurrency within one active server.

By default, after a restart, Zabbix agent 2 will schedule the first data collection for active checks at a conditionally random time within the item’s update interval to prevent spikes in resource usage. To perform active checks that do not have Scheduling update interval immediately after the agent restart, set ForceActiveChecksOnStart parameter (global-level) or Plugins.<Plugin name>.System.ForceActiveChecksOnStart (affects only specific plugin checks) in the configuration file. Plugin-level parameter, if set, will override the global parameter. Forcing active checks on start is supported since Zabbix 6.0.2.

Check concurrency

Checks from different plugins can be executed concurrently. The number of concurrent checks within one plugin is limited by the plugin capacity setting. Each plugin may have a hardcoded capacity setting (100 being default) that can be lowered using the Plugins.<PluginName>.System.Capacity=N setting in the Plugins configuration parameter. Former name of this parameter Plugins.<PluginName>.Capacity is still supported, but has been deprecated in Zabbix 6.0.

See also: Plugin development guidelines.

Supported platforms

Agent 2 is supported for Linux and Windows platforms.

If installing from packages, Agent 2 is supported on:

- RHEL/CentOS 6, 7, 8
- SLES 15 SP1+
- Debian 9, 10
- Ubuntu 18.04, 20.04

On Windows, the agent 2 is supported on all desktop and server versions, on which an up-to-date supported Go version can be installed.

Installation

Zabbix agent 2 is available in pre-compiled Zabbix packages. To compile Zabbix agent 2 from sources you have to specify the --enable-agent2 configure option.

Options

The following command line parameters can be used with Zabbix agent 2:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-c --config &lt;config-file&gt;</td>
<td>Path to the configuration file. You may use this option to specify a configuration file that is not the default one. On UNIX, default is /usr/local/etc/zabbix_agent2.conf or as set by compile-time variables --sysconfdir or --prefix</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>-f</td>
<td>Run Zabbix agent in foreground (default: true).</td>
</tr>
<tr>
<td>-p</td>
<td>Print known items and exit. Note: To return user parameter results as well, you must specify the configuration file (if it is not in the default location).</td>
</tr>
<tr>
<td>-t</td>
<td>Test specified item and exit. Note: To return user parameter results as well, you must specify the configuration file (if it is not in the default location).</td>
</tr>
<tr>
<td>-h</td>
<td>Print help information and exit.</td>
</tr>
<tr>
<td>-v</td>
<td>Print debugging information. Use this option with -p and -t options.</td>
</tr>
<tr>
<td>-V</td>
<td>Print agent version number and exit.</td>
</tr>
<tr>
<td>-R</td>
<td>Perform administrative functions. See runtime control.</td>
</tr>
</tbody>
</table>

Specific examples of using command line parameters:

- print all built-in agent items with values
- test a user parameter with "mysql.ping" key defined in the specified configuration file

shell> zabbix_agent2 --print
shell> zabbix_agent2 -t "mysql.ping" -c /etc/zabbix/zabbix_agentd.conf

Runtime control

Runtime control provides some options for remote control.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>log_level_increase</td>
<td>Increase log level.</td>
</tr>
<tr>
<td>log_level_decrease</td>
<td>Decrease log level.</td>
</tr>
<tr>
<td>metrics</td>
<td>List available metrics.</td>
</tr>
<tr>
<td>version</td>
<td>Display agent version.</td>
</tr>
<tr>
<td>userparameter_reload</td>
<td>Reload user parameters from the current configuration file. Note that UserParameter is the only agent configuration option that will be reloaded.</td>
</tr>
<tr>
<td>help</td>
<td>Display help information on runtime control.</td>
</tr>
</tbody>
</table>

Examples:

- increasing log level for agent 2
- print runtime control options

shell> zabbix_agent2 -R log_level_increase
shell> zabbix_agent2 -R help

Configuration file

The configuration parameters of agent 2 are mostly compatible with Zabbix agent with some exceptions.

<table>
<thead>
<tr>
<th>New parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ControlSocket</td>
<td>The runtime control socket path. Agent 2 uses a control socket for runtime commands.</td>
</tr>
<tr>
<td>EnablePersistentBuffer,</td>
<td>These parameters are used to configure persistent storage on agent 2 for active items.</td>
</tr>
<tr>
<td>PersistentBufferFile,</td>
<td></td>
</tr>
<tr>
<td>PersistentBufferPeriod</td>
<td></td>
</tr>
<tr>
<td>ForceActiveChecksOnStart</td>
<td>Determines whether the agent should perform active checks immediately after restart or spread evenly over time.</td>
</tr>
<tr>
<td>Plugins</td>
<td>Plugins may have their own parameters, in the format Plugins.&lt;Plugin name&gt;.&lt;Parameter&gt;=&lt;value&gt;. A common plugin parameter is System.Capacity, setting the limit of checks that can be executed at the same time.</td>
</tr>
<tr>
<td>StatusPort</td>
<td>The port agent 2 will be listening on for HTTP status request and display of a list of configured plugins and some internal parameters</td>
</tr>
<tr>
<td>Dropped parameters</td>
<td>Description</td>
</tr>
<tr>
<td>AllowRoot, User</td>
<td>Not supported because daemonization is not supported.</td>
</tr>
<tr>
<td>LoadModule,</td>
<td>Loadable modules are not supported.</td>
</tr>
<tr>
<td>LoadModulePath</td>
<td></td>
</tr>
<tr>
<td>New parameters</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>StartAgents</td>
<td>This parameter was used in Zabbix agent to increase passive check concurrency or disable them. In Agent 2, the concurrency is configured at a plugin level and can be limited by a capacity setting. Whereas disabling passive checks is not currently supported.</td>
</tr>
<tr>
<td>HostInterface,</td>
<td>Not yet supported.</td>
</tr>
<tr>
<td>HostInterfaceItem</td>
<td></td>
</tr>
</tbody>
</table>

For more details see the configuration file options for [zabbix_agent2](#).

**Exit codes**

Starting from version 4.4.8 Zabbix agent 2 can also be compiled with older OpenSSL versions (1.0.1, 1.0.2).

In this case Zabbix provides mutexes for locking in OpenSSL. If a mutex lock or unlock fails then an error message is printed to the standard error stream (STDERR) and Agent 2 exits with return code 2 or 3, respectively.

### 4 Proxy

**Overview**

Zabbix proxy is a process that may collect monitoring data from one or more monitored devices and send the information to the Zabbix server, essentially working on behalf of the server. All collected data is buffered locally and then transferred to the Zabbix server the proxy belongs to.

Deploying a proxy is optional, but may be very beneficial to distribute the load of a single Zabbix server. If only proxies collect data, processing on the server becomes less CPU and disk I/O hungry.

A Zabbix proxy is the ideal solution for centralized monitoring of remote locations, branches and networks with no local administrators.

Zabbix proxy requires a separate database.

Note that databases supported with Zabbix proxy are SQLite, MySQL and PostgreSQL. Using Oracle is at your own risk and may contain some limitations as, for example, in return values of low-level discovery rules.

See also: [Using proxies in a distributed environment](#)

**Running proxy**

If installed as package

Zabbix proxy runs as a daemon process. The proxy can be started by executing:

```
shell> service zabbix-proxy start
```

This will work on most of GNU/Linux systems. On other systems you may need to run:

```
shell> /etc/init.d/zabbix-proxy start
```

Similarly, for stopping/restarting/viewing status of Zabbix proxy, use the following commands:

```
shell> service zabbix-proxy stop
shell> service zabbix-proxy restart
shell> service zabbix-proxy status
```

Start up manually

If the above does not work you have to start it manually. Find the path to the zabbix_proxy binary and execute:

```
shell> zabbix_proxy
```

You can use the following command line parameters with Zabbix proxy:

- `-c --config <file>` path to the configuration file
- `-f --foreground` run Zabbix proxy in foreground
- `-R --runtime-control <option>` perform administrative functions
- `-h --help` give this help
- `-V --version` display version number

Examples of running Zabbix proxy with command line parameters:
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf
shell> zabbix_proxy --help
shell> zabbix_proxy -V

Runtime control

Runtime control options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>config_cache_reload</td>
<td>Reload configuration cache. Ignored if cache is being currently loaded.</td>
<td>config_cache_reload</td>
</tr>
<tr>
<td></td>
<td>Active Zabbix proxy will connect to the Zabbix server and request configuration data.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Passive Zabbix proxy will request configuration data from Zabbix server the next time when the server connects to the proxy.</td>
<td></td>
</tr>
<tr>
<td>diaginfo=&lt;target&gt;</td>
<td>Gather diagnostic information in the proxy log file.</td>
<td>diaginfo</td>
</tr>
<tr>
<td>snmp_cache_reload</td>
<td>Reload SNMP cache, clear the SNMP properties (engine time, engine boots, engine id, credentials) for all hosts.</td>
<td>snmp_cache_reload</td>
</tr>
<tr>
<td>housekeeper_execute</td>
<td>Start the housekeeping procedure. Ignored if the housekeeping procedure is currently in progress.</td>
<td>housekeeper_execute</td>
</tr>
<tr>
<td>log_level_increase</td>
<td>Increase log level, affects all processes if target is not specified.</td>
<td>log_level_increase</td>
</tr>
<tr>
<td></td>
<td>Not supported on *<em>BSD</em> systems.</td>
<td></td>
</tr>
<tr>
<td>log_level_decrease</td>
<td>Decrease log level, affects all processes if target is not specified.</td>
<td>log_level_decrease</td>
</tr>
<tr>
<td></td>
<td>Not supported on *<em>BSD</em> systems.</td>
<td></td>
</tr>
</tbody>
</table>

Example of using runtime control to reload the proxy configuration cache:
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R config_cache_reload

Examples of using runtime control to gather diagnostic information:
Gather all available diagnostic information in the proxy log file:
shell> zabbix_proxy -R diaginfo

Gather history cache statistics in the proxy log file:
shell> zabbix_proxy -R diaginfo=historycache

Example of using runtime control to reload the SNMP cache:
shell> zabbix_proxy -R snmp_cache_reload

Example of using runtime control to trigger execution of housekeeper
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R housekeeper_execute

Examples of using runtime control to change log level:
Increase log level of all processes:
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase

Increase log level of second poller process:
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase=poller,2

Increase log level of process with PID 1234:
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase=1234
Decrease log level of all http poller processes:
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_decrease="http poller"

Process user
Zabbix proxy is designed to run as a non-root user. It will run as whatever non-root user it is started as. So you can run proxy as any non-root user without any issues.

If you will try to run it as ‘root’, it will switch to a hardcoded ‘zabbix’ user, which must be present on your system. You can only run proxy as ‘root’ if you modify the ‘AllowRoot’ parameter in the proxy configuration file accordingly.

Configuration file
See the configuration file options for details on configuring zabbix_proxy.

Proxy process types
• availability manager - process for host availability updates
• configuration syncer - process for managing in-memory cache of configuration data
• data sender - proxy data sender
• discoverer - process for discovery of devices
• heartbeat sender - proxy heartbeat sender
• history syncer - history DB writer
• housekeeper - process for removal of old historical data
• http poller - web monitoring poller
• icmp pinger - poller for icmp ping checks
• ipmi manager - IPMI poller manager
• ipmi poller - poller for IPMI checks
• java poller - poller for Java checks
• odbc poller - poller for ODBC checks
• poller - normal poller for passive checks
• preprocessing manager - manager of preprocessing tasks
• preprocessing worker - process for data preprocessing
• self-monitoring - process for collecting internal server statistics
• snmp trapper - trapper for SNMP traps
• task manager - process for remote execution of tasks requested by other components (e.g. close problem, acknowledge problem, check item value now, remote command functionality)
• trapper - trapper for active checks, traps, proxy communication
• unreachable poller - poller for unreachable devices
• vmware collector - VMware data collector responsible for data gathering from VMware services

The proxy log file can be used to observe these process types.

Various types of Zabbix proxy processes can be monitored using the zabbix[process,<type>,<mode>,<state>] internal item.

Supported platforms
Zabbix proxy runs on the same list of server#supported platforms as Zabbix server.

Locale
Note that the proxy requires a UTF-8 locale so that some textual items can be interpreted correctly. Most modern Unix-like systems have a UTF-8 locale as default, however, there are some systems where that may need to be set specifically.

5 Java gateway

Overview
Native support for monitoring JMX applications exists in the form of a Zabbix daemon called "Zabbix Java gateway", available since Zabbix 2.0. Zabbix Java gateway is a daemon written in Java. To find out the value of a particular JMX counter on a host, Zabbix server queries Zabbix Java gateway, which uses the JMX management API to query the application of interest remotely. The application does not need any additional software installed, it just has to be started with -Dcom.sun.management.jmxremote option on the command line.

Java gateway accepts incoming connection from Zabbix server or proxy and can only be used as a “passive proxy”. As opposed to Zabbix proxy, it may also be used from Zabbix proxy (Zabbix proxies cannot be chained). Access to each Java gateway is configured directly in Zabbix server or proxy configuration file, thus only one Java gateway may be configured per Zabbix server.
or Zabbix proxy. If a host will have items of type JMX agent and items of other type, only the JMX agent items will be passed to Java gateway for retrieval.

When an item has to be updated over Java gateway, Zabbix server or proxy will connect to the Java gateway and request the value, which Java gateway in turn retrieves and passes back to the server or proxy. As such, Java gateway does not cache any values.

Zabbix server or proxy has a specific type of processes that connect to Java gateway, controlled by the option StartJavaPollers. Internally, Java gateway starts multiple threads, controlled by the START_POLLERS option. On the server side, if a connection takes more than Timeout seconds, it will be terminated, but Java gateway might still be busy retrieving value from the JMX counter. To solve this, there is the TIMEOUT option in Java gateway that allows to set timeout for JMX network operations.

Zabbix server or proxy will try to pool requests to a single JMX target together as much as possible (affected by item intervals) and send them to the Java gateway in a single connection for better performance.

It is suggested to have StartJavaPollers less than or equal to START_POLLERS, otherwise there might be situations when no threads are available in the Java gateway to service incoming requests; in such a case Java gateway uses ThreadPoolExecutor:CallerRunsPolicy, meaning that the main thread will service the incoming request and temporarily will not accept any new requests.

Getting Java gateway

You can install Java gateway either from the sources or packages downloaded from Zabbix website.

Using the links below you can access information how to get and run Zabbix Java gateway, how to configure Zabbix server (or Zabbix proxy) to use Zabbix Java gateway for JMX monitoring, and how to configure Zabbix items in Zabbix frontend that correspond to particular JMX counters.

<table>
<thead>
<tr>
<th>Installation from</th>
<th>Instructions</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sources</td>
<td>Installation</td>
<td>Setup</td>
</tr>
<tr>
<td>RHEL/CentOS packages</td>
<td>Installation</td>
<td>Setup</td>
</tr>
<tr>
<td>Debian/Ubuntu packages</td>
<td>Installation</td>
<td>Setup</td>
</tr>
</tbody>
</table>

1 Setup from sources

Overview

If installed from sources, the following information will help you in setting up Zabbix Java gateway.

Overview of files

If you obtained Java gateway from sources, you should have ended up with a collection of shell scripts, JAR and configuration files under $PREFIX/sbin/zabbix_java. The role of these files is summarized below.

bin/zabbix-java-gateway-$VERSION.jar

Java gateway JAR file itself.

lib/logback-core-0.9.27.jar
lib/logback-classic-0.9.27.jar
lib/slf4j-api-1.6.1.jar
lib/android-json-4.3_r3.1.jar

Dependencies of Java gateway: Logback, SLF4J, and Android JSON library.

lib/logback.xml
lib/logback-console.xml

Configuration files for Logback.

shutdown.sh
startup.sh

Convenience scripts for starting and stopping Java gateway.

settings.sh

Configuration file that is sourced by startup and shutdown scripts above.

Configuring and running Java gateway

By default, Java gateway listens on port 10052. If you plan on running Java gateway on a different port, you can specify that in settings.sh script. See the description of Java gateway configuration file for how to specify this and other options.
Port 10052 is not IANA registered.

Once you are comfortable with the settings, you can start Java gateway by running the startup script:

```
$ ./startup.sh
```

Likewise, once you no longer need Java gateway, run the shutdown script to stop it:

```
$ ./shutdown.sh
```

Note that unlike server or proxy, Java gateway is lightweight and does not need a database.

Configuring server for use with Java gateway

With Java gateway up and running, you have to tell Zabbix server where to find Zabbix Java gateway. This is done by specifying JavaGateway and JavaGatewayPort parameters in the server configuration file. If the host on which JMX application is running is monitored by Zabbix proxy, then you specify the connection parameters in the proxy configuration file instead.

```
JavaGateway=192.168.3.14
JavaGatewayPort=10052
```

By default, server does not start any processes related to JMX monitoring. If you wish to use it, however, you have to specify the number of pre-forked instances of Java pollers. You do this in the same way you specify regular pollers and trappers.

```
StartJavaPollers=5
```

Do not forget to restart server or proxy, once you are done with configuring them.

Debugging Java gateway

In case there are any problems with Java gateway or an error message that you see about an item in the frontend is not descriptive enough, you might wish to take a look at Java gateway log file.

By default, Java gateway logs its activities into /tmp/zabbix_java.log file with log level "info". Sometimes that information is not enough and there is a need for information at log level "debug". In order to increase logging level, modify file lib/logback.xml and change the level attribute of <root> tag to "debug":

```
<root level="debug">
    <appender-ref ref="FILE" />
</root>
```

Note that unlike Zabbix server or Zabbix proxy, there is no need to restart Zabbix Java gateway after changing logback.xml file - changes in logback.xml will be picked up automatically. When you are done with debugging, you can return the logging level to "info".

If you wish to log to a different file or a completely different medium like database, adjust logback.xml file to meet your needs. See Logback Manual for more details.

Sometimes for debugging purposes it is useful to start Java gateway as a console application rather than a daemon. To do that, comment out PID_FILE variable in settings.sh. If PID_FILE is omitted, startup.sh script starts Java gateway as a console application and makes Logback use lib/logback-console.xml file instead, which not only logs to console, but has logging level "debug" enabled as well.

Finally, note that since Java gateway uses SLF4J for logging, you can replace Logback with the framework of your choice by placing an appropriate JAR file in lib directory. See SLF4J Manual for more details.

JMX monitoring

See JMX monitoring page for more details.

### 2 Setup from RHEL/CentOS packages

Overview

If installed from RHEL/CentOS packages, the following information will help you in setting up Zabbix Java gateway.

Configuring and running Java gateway

Configuration parameters of Zabbix Java gateway may be tuned in the file:

```
/etc/zabbix/zabbix_java_gateway.conf
```

For more details, see Zabbix Java gateway configuration parameters.

To start Zabbix Java gateway:
To automatically start Zabbix Java gateway on boot:

RHEL 7 and later:

# systemctl enable zabbix-java-gateway

RHEL prior to 7:

# chkconfig --level 12345 zabbix-java-gateway on

Configuring server for use with Java gateway

With Java gateway up and running, you have to tell Zabbix server where to find Zabbix Java gateway. This is done by specifying JavaGateway and JavaGatewayPort parameters in the server configuration file. If the host on which JMX application is running is monitored by Zabbix proxy, then you specify the connection parameters in the proxy configuration file instead.

JavaGateway=192.168.3.14
JavaGatewayPort=10052

By default, server does not start any processes related to JMX monitoring. If you wish to use it, however, you have to specify the number of pre-forked instances of Java pollers. You do this in the same way you specify regular pollers and trappers.

StartJavaPollers=5

Do not forget to restart server or proxy, once you are done with configuring them.

Debugging Java gateway

Zabbix Java gateway log file is:

/var/log/zabbix/zabbix_java_gateway.log

If you like to increase the logging, edit the file:

/etc/zabbix/zabbix_java_gateway_logback.xml

and change level="info" to "debug" or even "trace" (for deep troubleshooting):

<configuration scan="true" scanPeriod="15 seconds">

    <root level="info">
        <appender-ref ref="FILE" />
    </root>

</configuration>

JMX monitoring

See JMX monitoring page for more details.

3 Setup from Debian/Ubuntu packages

Overview

If installed from Debian/Ubuntu packages, the following information will help you in setting up Zabbix Java gateway.

Configuring and running Java gateway

Java gateway configuration may be tuned in the file:

/etc/zabbix/zabbix_java_gateway.conf

For more details, see Zabbix Java gateway configuration parameters.

To start Zabbix Java gateway:

# service zabbix-java-gateway restart

To automatically start Zabbix Java gateway on boot:

# systemctl enable zabbix-java-gateway
Configuring server for use with Java gateway

With Java gateway up and running, you have to tell Zabbix server where to find Zabbix Java gateway. This is done by specifying JavaGateway and JavaGatewayPort parameters in the server configuration file. If the host on which JMX application is running is monitored by Zabbix proxy, then you specify the connection parameters in the proxy configuration file instead.

```
JavaGateway=192.168.3.14
JavaGatewayPort=10052
```

By default, server does not start any processes related to JMX monitoring. If you wish to use it, however, you have to specify the number of pre-forked instances of Java pollers. You do this in the same way you specify regular pollers and trappers.

```
StartJavaPollers=5
```

Do not forget to restart server or proxy, once you are done with configuring them.

Debugging Java gateway

Zabbix Java gateway log file is:
```
/var/log/zabbix/zabbix_java_gateway.log
```

If you like to increase the logging, edit the file:
```
/etc/zabbix/zabbix_java_gateway_logback.xml
```

and change `level="info"` to "debug" or even "trace" (for deep troubleshooting):

```
<configuration scan="true" scanPeriod="15 seconds">
    [...]  
    <root level="info">
        <appender-ref ref="FILE" />
    </root>
</configuration>
```

JMX monitoring

See JMX monitoring page for more details.

6 Sender

Overview

Zabbix sender is a command line utility that may be used to send performance data to Zabbix server for processing.

The utility is usually used in long running user scripts for periodical sending of availability and performance data.

For sending results directly to Zabbix server or proxy, a trapper item type must be configured.

Running Zabbix sender

An example of running Zabbix UNIX sender:
```
shell> cd bin
shell> ./zabbix_sender -z zabbix -s "Linux DB3" -k db.connections -o 43
```

where:

- z - Zabbix server host (IP address can be used as well)
- s - technical name of monitored host (as registered in Zabbix frontend)
- k - item key
- o - value to send

Options that contain whitespaces, must be quoted using double quotes.

Zabbix sender can be used to send multiple values from an input file. See the Zabbix sender manpage for more information.

If a configuration file is specified, Zabbix sender uses all addresses defined in the agent ServerActive configuration parameter for sending data. If sending to one address fails, the sender tries sending to the other addresses. If sending of batch data fails to one address, the following batches are not sent to this address.

Zabbix sender accepts strings in UTF-8 encoding (for both UNIX-like systems and Windows) without byte order mark (BOM) first in the file.
Zabbix sender on Windows can be run similarly:

```
zabbix_sender.exe [options]
```

Since Zabbix 1.8.4, zabbix_sender realtime sending scenarios have been improved to gather multiple values passed to it in close succession and send them to the server in a single connection. A value that is not further apart from the previous value than 0.2 seconds can be put in the same stack, but maximum pooling time still is 1 second.

Zabbix sender will terminate if invalid (not following parameter=value notation) parameter entry is present in the specified configuration file.

### 7 Get

**Overview**

Zabbix get is a command line utility which can be used to communicate with Zabbix agent and retrieve required information from the agent.

The utility is usually used for the troubleshooting of Zabbix agents.

**Running Zabbix get**

An example of running Zabbix get under UNIX to get the processor load value from the agent:

```
shell> cd bin
shell> ./zabbix_get -s 127.0.0.1 -p 10050 -k system.cpu.load[all,avg1]
```

Another example of running Zabbix get for capturing a string from a website:

```
shell> cd bin
shell> ./zabbix_get -s 192.168.1.1 -p 10050 -k "web.page.regexp[www.example.com,,,"USA: ([a-zA-Z0-9.-]+)]",\1"
```

Note that the item key here contains a space so quotes are used to mark the item key to the shell. The quotes are not part of the item key; they will be trimmed by the shell and will not be passed to Zabbix agent.

Zabbix get accepts the following command line parameters:

```
-s --host <host name or IP> Specify host name or IP address of a host.
-p --port <port number> Specify port number of agent running on the host. Default is 10050.
-I --source-address <IP address> Specify source IP address.
-t --timeout <seconds> Specify timeout. Valid range: 1-30 seconds (default: 30 seconds).
-k --key <item key> Specify key of item to retrieve value of.
-h --help Give this help.
-V --version Display version number.
```

See also Zabbix get manpage for more information.

Zabbix get on Windows can be run similarly:

```
zabbix_get.exe [options]
```

### 8 JS

**Overview**

zabbix_js is a command line utility that can be used for embedded script testing.

This utility will execute a user script with a string parameter and print the result. Scripts are executed using the embedded Zabbix scripting engine.

In case of compilation or execution errors zabbix_js will print the error in stderr and exit with code 1.

**Usage**

```
zabbix_js -s script-file -p input-param [-l log-level] [-t timeout]
zabbix_js -s script-file -i input-file [-l log-level] [-t timeout]
zabbix_js -h
zabbix_js -V
```

zabbix_js accepts the following command line parameters:
9 Web service

Overview

Zabbix web service is a process that is used for communication with external web services. Currently, Zabbix web service is used for generating and sending scheduled reports with plans to add additional functionality in the future.

Zabbix server connects to the web service via HTTP(S). Zabbix web service requires Google Chrome to be installed on the same host; on some distributions the service may also work with Chromium (see known issues).

Installation

Zabbix web service is available in pre-compiled Zabbix packages available for download at Zabbix website. To compile Zabbix web service from sources, specify the --enable-webservice configure option.

See also:

• Configuration file options for zabbix_web_service;
• Setting up scheduled reports

4. Installation

Please use the sidebar to access content in the Installation section.

1 Getting Zabbix

Overview

There are four ways of getting Zabbix:

• Install it from the distribution packages
• Download the latest source archive and compile it yourself
• Install it from the containers
• Download the virtual appliance

To download the latest distribution packages, pre-compiled sources or the virtual appliance, go to the Zabbix download page, where direct links to latest versions are provided.

Getting Zabbix source code

There are several ways of getting Zabbix source code:

• You can download the released stable versions from the official Zabbix website
• You can download nightly builds from the official Zabbix website developer page
• You can get the latest development version from the Git source code repository system:
  - The primary location of the full repository is at https://git.zabbix.com/scm/zbx/zabbix.git
  - Master and supported releases are also mirrored to Github at https://github.com/zabbix/zabbix

A Git client must be installed to clone the repository. The official commandline Git client package is commonly called git in distributions. To install, for example, on Debian/Ubuntu, run:
To grab all Zabbix source, change to the directory you want to place the code in and execute:

git clone https://git.zabbix.com/scm/zbx/zabbix.git

## 2 Requirements

### Hardware

#### Memory

Zabbix requires both physical and disk memory. The amount of required disk memory obviously depends on the number of hosts and parameters that are being monitored. If you’re planning to keep a long history of monitored parameters, you should be thinking of at least a couple of gigabytes to have enough space to store the history in the database. Each Zabbix daemon process requires several connections to a database server. The amount of memory allocated for the connection depends on the configuration of the database engine.

The more physical memory you have, the faster the database (and therefore Zabbix) works!

#### CPU

Zabbix and especially Zabbix database may require significant CPU resources depending on number of monitored parameters and chosen database engine.

### Other hardware

A serial communication port and a serial GSM modem are required for using SMS notification support in Zabbix. USB-to-serial converter will also work.

### Examples of OS/hardware configuration

The table provides examples of OS/hardware configurations:

<table>
<thead>
<tr>
<th>Installation size/OS version (dedicated servers/on-prem/virtual/clouds)</th>
<th>Large/very large installations (dedicated servers/on-prem/virtual/clouds)</th>
<th>Containers base</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>Medium</td>
<td>Large</td>
</tr>
</tbody>
</table>

**Linux-based operating systems**

- **Alma Linux** 8+, 9+
- **Alpine** 3.14-, 3.16+
- **Amazon Linux** Preview release 1
- **Debian** 11+
- **OpenSUSE** 15.4+
- **Oracle Linux** 8+, 9+
- **Raspbian** 11+
- **Rocky Linux** 8+, 9+
- **Red Hat Enterprise Linux** 8+, 9+
- **Ubuntu LTS** 20.04+, 22.04+
<table>
<thead>
<tr>
<th>UNIX/BSD/other operating systems</th>
<th>Large/very large installations (dedicated servers/on-prem/virtual/clouds)</th>
<th>Containers base (dedicated servers/on-prem/virtual/clouds)</th>
<th>Jails/Zones/LPAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>FreeBSD</td>
<td>13.1+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>HP-UX</td>
<td>HP-UX v3</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>IBM AIX</td>
<td>AIX 7.3+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>macOS</td>
<td>12.4+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>NetBSD</td>
<td>9+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>OmniOS</td>
<td>Long-term support (LTS) release</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>OpenBSD</td>
<td>7.1+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Solaris</td>
<td>11.4+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hardware examples</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
<th>Very large</th>
<th>All</th>
<th>Database</th>
<th>Server</th>
<th>Frontend</th>
<th>Proxy</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU (2018+ m.y.)</td>
<td>2 core</td>
<td></td>
<td>8 core</td>
<td>4+ core²</td>
<td>4 core</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU High clock frequency (2018+ m.y.)</td>
<td>1 core</td>
<td>8 core</td>
<td>16+ core</td>
<td>8+ core</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAM (2018+ m.y.)</td>
<td>1 GB</td>
<td>2 GB</td>
<td>16 GB</td>
<td>64+ GB</td>
<td>32+ GB</td>
<td>16 GB</td>
<td>8+ GB²</td>
<td>4 GB</td>
<td></td>
</tr>
<tr>
<td>NUMA (check/setup if &gt;2 sockets or &gt;8 cores VM)</td>
<td>RAID</td>
<td>vRAID</td>
<td>vRAID</td>
<td>vRAID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage/SAN</td>
<td>RAID</td>
<td>vRAID</td>
<td>vRAID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flash storage/SAN</td>
<td>RAID</td>
<td>vRAID</td>
<td>vRAID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSD write-intensive or mixed usage</td>
<td>RAID</td>
<td>vRAID</td>
<td>vRAID</td>
<td>vRAID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NVME</td>
<td>RAID</td>
<td>vRAID</td>
<td>vRAID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Any storage for OS and logs</td>
<td>RAID</td>
<td>vRAID</td>
<td>vRAID</td>
<td>vRAID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hosts</td>
<td>100</td>
<td>500</td>
<td>&gt;1000</td>
<td>&gt;10000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Database examples | Oracle Database | | | | | | | | |
|-------------------|-----------------| | | | | | | | |
| Percona Server     | +               | +     | +     | +     | +   |           |        |          |       |
Installation size (dedicated servers/on-prem/virtual/clouds) | Containers base (dedicated servers/on-prem/virtual/clouds)
--- | ---
MariaDB Enterprise/Community Server | + + + + + +
MySQL Community Edition/Commercial | + + + + + +
PostgreSQL/PostgreSQL Pro/EnterpriseDB | + + + + + +
TimescaleDB | + + + + + +

### Filesystem examples
- BTRFS
- EXT4
- XFS
- ZFS

### Networking examples
- Synchronous cluster (RTT/Latency) - Single DC
- Asynchronous replication (RTT/Latency) - <5 seconds
- NIC teaming/bonding
- Bandwidth
  - Recommended: 100mbits/sec+ to 1000mbits/sec+
  - Recommended: 100mbits/sec+ to 1000mbits/sec+

1. For adoption and testing
2. According to the amount of users
3. Depending on the workload and requirements

Actual configuration depends on the number of active items and refresh rates very much (see database size section of this page for details). It is highly recommended to run the database on a separate box for large installations.

**Supported platforms**

Due to security requirements and the mission-critical nature of the monitoring server, UNIX is the only operating system that can consistently deliver the necessary performance, fault tolerance, and resilience. Zabbix operates on market-leading versions.

Zabbix components are available and tested for the following platforms:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Server</th>
<th>Agent</th>
<th>Agent2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>IBM AIX</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>FreeBSD</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>NetBSD</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>OpenBSD</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>HP-UX</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>Mac OS X</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>Solaris</td>
<td>x</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>Windows</td>
<td>-</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Zabbix server/agent may work on other Unix-like operating systems as well. Zabbix agent is supported on all Windows desktop and server versions since XP.

Zabbix disables core dumps if compiled with encryption and does not start if the system does not allow disabling of core dumps.

**Required software**

---

38
Zabbix is built around modern web servers, leading database engines, and PHP scripting language.

**Third-party external surrounding software**

Mandatory requirements are needed always. Optional requirements are needed for the support of the specific function.

<table>
<thead>
<tr>
<th>Software</th>
<th>Mandatory status</th>
<th>Supported versions</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL/Percona</td>
<td>One of 8.0.X</td>
<td>Required if MySQL (or Percona) is used as Zabbix backend database. InnoDB engine is required. We recommend using the MariaDB Connector/C library for building server/proxy.</td>
<td></td>
</tr>
<tr>
<td>MariaDB</td>
<td>10.5.00-10.8.X</td>
<td>InnoDB engine is required. We recommend using the MariaDB Connector/C library for building server/proxy.</td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>19c - 21c</td>
<td>Required if Oracle is used as Zabbix backend database.</td>
<td></td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>13.0-14.X</td>
<td>Required if PostgreSQL is used as Zabbix backend database.</td>
<td></td>
</tr>
<tr>
<td>TimescaleDB for PostgreSQL</td>
<td>2.0.1-2.7</td>
<td>Required if TimescaleDB is used as a PostgreSQL database extension. Make sure to install TimescaleDB Community Edition, which supports compression.</td>
<td></td>
</tr>
<tr>
<td>SQLite</td>
<td>Optional 3.3.5-3.34.X</td>
<td>SQLite is only supported with Zabbix proxies. Required if SQLite is used as Zabbix proxy database.</td>
<td></td>
</tr>
<tr>
<td>smartmontools</td>
<td>who 7.1 or later</td>
<td>Required for Zabbix agent 2.</td>
<td></td>
</tr>
<tr>
<td>dpkg</td>
<td></td>
<td>Required for the user count plugin.</td>
<td></td>
</tr>
<tr>
<td>pktool</td>
<td></td>
<td>Required for the system.sw.packages plugin.</td>
<td></td>
</tr>
<tr>
<td>rpm</td>
<td></td>
<td>Required for the system.sw.packages plugin.</td>
<td></td>
</tr>
<tr>
<td>pacman</td>
<td></td>
<td>Required for the system.sw.packages plugin.</td>
<td></td>
</tr>
</tbody>
</table>

Although Zabbix can work with databases available in the operating systems, for the best experience, we recommend using databases installed from the official database developer repositories.

**Frontend**

The minimum supported screen width for Zabbix frontend is 1200px.

Mandatory requirements are needed always. Optional requirements are needed for the support of the specific function.

<table>
<thead>
<tr>
<th>Software</th>
<th>Mandatory status</th>
<th>Version</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>yes</td>
<td>1.3.12 or later</td>
<td></td>
</tr>
<tr>
<td>PHP</td>
<td></td>
<td>7.4.0 or later, 8.0.8.1</td>
<td></td>
</tr>
<tr>
<td>PHP extensions:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gd</td>
<td>yes</td>
<td>2.0.28 or later</td>
<td>PHP GD extension must support PNG images (--with-png-dir), JPEG (--with-jpeg-dir) images and FreeType 2 (--with-freetype-dir). php-bcmath (--enable-bcmath) php-ctype (--enable-ctype) php-xml, if provided as a separate package by the distributor.</td>
</tr>
<tr>
<td>bcmath</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctype</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libXML</td>
<td></td>
<td>2.6.15 or later</td>
<td>php-xml, if provided as a separate package by the distributor.</td>
</tr>
<tr>
<td>xmlreader</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xmlwriter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>session</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sockets</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mbstring</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gettext</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ldap</td>
<td>No</td>
<td></td>
<td>php-ldap. Required only if LDAP authentication is used in the frontend.</td>
</tr>
<tr>
<td>openssl</td>
<td></td>
<td></td>
<td>php-openssl. Required only if SAML authentication is used in the frontend.</td>
</tr>
<tr>
<td>mysqli</td>
<td></td>
<td></td>
<td>Required if MySQL is used as Zabbix backend database.</td>
</tr>
<tr>
<td>oci8</td>
<td></td>
<td></td>
<td>Required if Oracle is used as Zabbix backend database.</td>
</tr>
<tr>
<td>psql</td>
<td></td>
<td></td>
<td>Required if PostgreSQL is used as Zabbix backend database.</td>
</tr>
</tbody>
</table>
Third-party frontend libraries that are supplied with Zabbix:

<table>
<thead>
<tr>
<th>Library</th>
<th>Mandatory status</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>jQuery JavaScript Library</td>
<td>Yes</td>
<td>JavaScript library that simplifies the process of cross-browser development. Version 3.4.1 or higher.</td>
</tr>
<tr>
<td>jQuery UI</td>
<td></td>
<td>A set of user interface interactions, effects, widgets, and themes built on top of jQuery. Version 1.12.1 or higher.</td>
</tr>
<tr>
<td>OneLogin's SAML PHP Toolkit</td>
<td></td>
<td>A PHP toolkit that adds SAML 2.0 authentication support to be able to sign in to Zabbix. Version 3.4.1 or higher.</td>
</tr>
<tr>
<td>Symfony Yaml Component</td>
<td></td>
<td>Adds support to export and import Zabbix configuration elements in the YAML format. Version 5.1.13 or higher.</td>
</tr>
</tbody>
</table>

Zabbix may work on previous versions of Apache, MySQL, Oracle, and PostgreSQL as well.

For other fonts than the default DejaVu, PHP function `imagerotate` might be required. If it is missing, these fonts might be rendered incorrectly when a graph is displayed. This function is only available if PHP is compiled with bundled GD, which is not the case in Debian and other distributions.

Third-party libraries used for writing and debugging Zabbix frontend code:

<table>
<thead>
<tr>
<th>Library</th>
<th>Mandatory status</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Composer</td>
<td>No</td>
<td>1.10.1</td>
<td>An application-level package manager for PHP that provides a standard format for managing dependencies of PHP software and required libraries.</td>
</tr>
<tr>
<td>PHPUnit</td>
<td></td>
<td>5.7.27</td>
<td>A PHP unit testing framework for testing Zabbix frontend.</td>
</tr>
<tr>
<td>SASS</td>
<td></td>
<td>3.4.22</td>
<td>A preprocessor scripting language that is interpreted and compiled into Cascading Style Sheets (CSS).</td>
</tr>
</tbody>
</table>

Web browser on client side

Cookies and JavaScript must be enabled.

The latest stable versions of Google Chrome, Mozilla Firefox, Microsoft Edge, Apple Safari, and Opera are supported.

The same-origin policy for IFrames is implemented, which means that Zabbix cannot be placed in frames on a different domain.

Still, pages placed into a Zabbix frame will have access to Zabbix frontend (through JavaScript) if the page that is placed in the frame and Zabbix frontend are on the same domain. A page like `http://secure-zabbix.com/cms/page.html`, if placed into dashboards on `http://secure-zabbix.com/zabbix/`, will have full JS access to Zabbix.

Server/proxy

Mandatory requirements are needed always. Optional requirements are needed for the support of the specific function.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Mandatory status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>libpcre</td>
<td>Yes</td>
<td>PCRE/PCRE2 library is required for Perl Compatible Regular Expression (PCRE) support. The naming may differ depending on the GNU/Linux distribution, for example ‘libpcre3’ or ‘libpcre1’. PCRE v8.x and PCRE2 v10.x (from Zabbix 6.0.0) are supported.</td>
</tr>
<tr>
<td>libevent</td>
<td></td>
<td>Required for bulk metric support and IPMI monitoring. Version 1.4 or higher. Note that for Zabbix proxy this requirement is optional; it is needed for IPMI monitoring support.</td>
</tr>
<tr>
<td>pthread</td>
<td></td>
<td>Required for mutex and read-write lock support.</td>
</tr>
<tr>
<td>zlib</td>
<td></td>
<td>Required for compression support.</td>
</tr>
<tr>
<td>resolv</td>
<td></td>
<td>Required for DNS resolution.</td>
</tr>
<tr>
<td>iconv</td>
<td></td>
<td>Required for text encoding/format conversion. Mandatory for Zabbix server on Linux.</td>
</tr>
</tbody>
</table>
### Requirement

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Mandatory status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>libmysqlclient</td>
<td>One of</td>
<td>Required if MySQL is used.</td>
</tr>
<tr>
<td>libmariadb</td>
<td></td>
<td>Required if MariaDB is used.</td>
</tr>
<tr>
<td>libpq</td>
<td></td>
<td>Required if PostgreSQL is used. Version 9.2 or higher.</td>
</tr>
<tr>
<td>libsqlite3</td>
<td></td>
<td>Required if Sqlite is used. Required for Zabbix proxy only.</td>
</tr>
<tr>
<td>openipmi</td>
<td></td>
<td>Required for IPMI support. Required for Zabbix server only.</td>
</tr>
<tr>
<td>libssh2 or libssh</td>
<td></td>
<td>Required for <strong>SSH checks</strong>. Version 1.0 or higher (libssh2); 0.6.0 or higher (libssh). libssh is supported since Zabbix 4.4.6.</td>
</tr>
<tr>
<td>fping</td>
<td></td>
<td>Required for <strong>ICMP ping items</strong>.</td>
</tr>
<tr>
<td>libcurl</td>
<td></td>
<td>Required for web monitoring, VMware monitoring, SMTP authentication, web . page . * Zabbix agent <strong>items</strong>, HTTP agent items and Elasticsearch (if used). Version 7.28.0 or higher is recommended. Libcurl version requirements: - SMTP authentication: version 7.20.0 or higher - Elasticsearch: version 7.28.0 or higher</td>
</tr>
<tr>
<td>libxml2</td>
<td></td>
<td>Required for VMware monitoring and XML XPath preprocessing.</td>
</tr>
<tr>
<td>netsnmp</td>
<td></td>
<td>Required for SNMP support. Version 5.3.0 or higher.</td>
</tr>
<tr>
<td>libunivodbc</td>
<td></td>
<td>Required for database monitoring. Required for Zabbix server only.</td>
</tr>
<tr>
<td>libgnutls or libopenssl</td>
<td></td>
<td>Required when using encryption. Minimum versions: libgnutls - 3.1.18, libopenssl - 1.0.1</td>
</tr>
<tr>
<td>ldap</td>
<td></td>
<td>Required for LDAP support.</td>
</tr>
</tbody>
</table>

### Agent

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Mandatory status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>libcxx</td>
<td>Yes</td>
<td>PCRE/PCRE2 library is required for <strong>Perl Compatible Regular Expression (PCRE) support</strong>. The naming may differ depending on the GNU/Linux distribution, for example ‘libpcre3’ or ‘libpcre1’. PCRE v8.x and PCRE2 v10.x (from Zabbix 6.0.0) are supported.</td>
</tr>
<tr>
<td>pthread</td>
<td></td>
<td>Required for mutex and read-write lock support.</td>
</tr>
<tr>
<td>zlib</td>
<td></td>
<td>Required for compression support.</td>
</tr>
<tr>
<td>resolv</td>
<td></td>
<td>Required for DNS resolution.</td>
</tr>
<tr>
<td>iconv</td>
<td></td>
<td>Required for text encoding/format conversion. Mandatory on Linux.</td>
</tr>
<tr>
<td>libgnutls or libopenssl</td>
<td></td>
<td>Minimum versions: libgnutls - 3.1.18, libopenssl - 1.0.1 On Microsoft Windows OpenSSL 1.1.1 or later is required.</td>
</tr>
<tr>
<td>ldap</td>
<td></td>
<td>Required for LDAP support.</td>
</tr>
<tr>
<td>libcurl</td>
<td></td>
<td>Required for web monitoring, VMware monitoring, SMTP authentication, web . page . * Zabbix agent <strong>items</strong>, HTTP agent items and Elasticsearch (if used). Version 7.28.0 or higher is recommended.</td>
</tr>
<tr>
<td>libxml2</td>
<td></td>
<td>Required for VMware monitoring and XML XPath preprocessing.</td>
</tr>
<tr>
<td>libmodbus</td>
<td></td>
<td>Required for Modbus support. Version 3.0 or higher.</td>
</tr>
<tr>
<td>libssh</td>
<td></td>
<td>Required for <strong>SSH checks</strong>. Version 0.6.0 or higher (libssh).</td>
</tr>
</tbody>
</table>

Starting from version 5.0.3, Zabbix agent will not work on AIX platforms below versions 6.1 TL07 / AIX 7.1 TL01.

### Agent 2

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Mandatory status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>libcxx</td>
<td>Yes</td>
<td>PCRE library is required for <strong>Perl Compatible Regular Expression (PCRE) support</strong>. The naming may differ depending on the GNU/Linux distribution, for example ‘libpcre3’ or ‘libpcre1’. PCRE v8.x and PCRE2 v10.x (from Zabbix 6.0.0) are supported.</td>
</tr>
</tbody>
</table>
### libopenssl

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Mandatory status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>libopenssl</td>
<td>No</td>
<td>Required when using encryption. OpenSSL 1.0.1 or later is required on UNIX platforms. The OpenSSL library must have PSK support enabled. LibreSSL is not supported. On Microsoft Windows systems OpenSSL 1.1.1 or later is required.</td>
</tr>
</tbody>
</table>

### Golang libraries

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Mandatory status</th>
<th>Minimum version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>git.zabbix.com/ap/plugin-support</td>
<td>Yes</td>
<td>0.0.0</td>
<td>Zabbix own support library. Mostly for plugins. Also used in MongoDB plugin.</td>
</tr>
<tr>
<td>github.com/BurntSushi/locker</td>
<td></td>
<td>0.0.0</td>
<td>Named read/write locks, access sync.</td>
</tr>
<tr>
<td>github.com/chromedp/cdproto</td>
<td></td>
<td>0.0.0</td>
<td>Generated commands, types, and events for the Chrome DevTools Protocol domains.</td>
</tr>
<tr>
<td>github.com/chromedp/chromedp</td>
<td></td>
<td>0.6.0</td>
<td>Chrome DevTools Protocol support (report generation).</td>
</tr>
<tr>
<td>github.com/dustin/gomemcached</td>
<td></td>
<td>0.0.0</td>
<td>A memcached binary protocol toolkit for go.</td>
</tr>
<tr>
<td>github.com/eclipse/paho.mqtt.golang</td>
<td></td>
<td>1.2.0</td>
<td>A library to handle MQTT connections.</td>
</tr>
<tr>
<td>github.com/fsnotify/fsnotify</td>
<td></td>
<td>1.4.9</td>
<td>Cross-platform file system notifications for Go.</td>
</tr>
<tr>
<td>github.com/go-lap/lap</td>
<td></td>
<td>3.0.3</td>
<td>Basic LDAP v3 functionality for the GO programming language.</td>
</tr>
<tr>
<td>github.com/go-ole/go-ole</td>
<td></td>
<td>1.2.4</td>
<td>Win32 ole implementation for golang.</td>
</tr>
<tr>
<td>github.com/godbus/dbus</td>
<td></td>
<td>4.1.0</td>
<td>Native Go bindings for D-Bus.</td>
</tr>
<tr>
<td>github.com/go-sql-driver/mysql</td>
<td></td>
<td>1.5.0</td>
<td>MySQL driver.</td>
</tr>
<tr>
<td>github.com/godror/godror</td>
<td></td>
<td>0.20.1</td>
<td>Oracle DB driver.</td>
</tr>
<tr>
<td>github.com/jack/pgx/v4</td>
<td></td>
<td>4.8.2</td>
<td>PostgreSQL driver.</td>
</tr>
<tr>
<td>github.com/mattin/go-sqlite3</td>
<td></td>
<td>2.0.3</td>
<td>Sqlite3 driver.</td>
</tr>
<tr>
<td>github.com/mediocregopher/radix/v3</td>
<td></td>
<td>3.3.5.0</td>
<td>Redis client.</td>
</tr>
<tr>
<td>github.com/memcached/mc/v3</td>
<td></td>
<td>3.0.1</td>
<td>Binary Memcached client.</td>
</tr>
<tr>
<td>github.com/miekg/dns</td>
<td></td>
<td>1.1.43</td>
<td>DNS library.</td>
</tr>
<tr>
<td>github.com/omeid/go-yarn</td>
<td></td>
<td>0.0.1</td>
<td>Embeddable filesystem mapped key-string store.</td>
</tr>
<tr>
<td>github.com/goburrow/modbus</td>
<td></td>
<td>0.1.0</td>
<td>Fault-tolerant implementation of Modbus.</td>
</tr>
<tr>
<td>golang.org/x/sys</td>
<td></td>
<td>0.0.0</td>
<td>Go packages for low-level interactions with the operating system. Also used in plugin support lib. Used in MongoDB plugin.</td>
</tr>
<tr>
<td>github.com/natefinch/ng</td>
<td></td>
<td>0.0.0</td>
<td>Windows named pipe implementation. Also used in plugin support lib. Used in MongoDB plugin.</td>
</tr>
<tr>
<td>github.com/goburrow/serial</td>
<td></td>
<td>0.1.0</td>
<td>Serial library for Modbus.</td>
</tr>
<tr>
<td>github.com/pkg/errors</td>
<td></td>
<td>0.9.1</td>
<td>Simple error handling primitives. Used in MongoDB plugin.</td>
</tr>
<tr>
<td>golang.org/x/xerrors</td>
<td></td>
<td>0.0.0</td>
<td>Functions to manipulate errors.</td>
</tr>
<tr>
<td>gopkg.in/asn1-ber.v1</td>
<td></td>
<td>1.0.0</td>
<td>Encoding/decoding library for ASN1 BER.</td>
</tr>
<tr>
<td>gopkg.in/yaml.v2</td>
<td></td>
<td>2.2.8</td>
<td>Go package to encode and decode YAML values.</td>
</tr>
<tr>
<td>github.com/go-stack/stack</td>
<td></td>
<td>1.8.0</td>
<td>Required package for MongoDB plugin mongo-driver lib.</td>
</tr>
<tr>
<td>github.com/golang/snap</td>
<td></td>
<td>0.0.1</td>
<td>Required package for MongoDB plugin mongo-driver lib.</td>
</tr>
<tr>
<td>github.com/klauspost/compress</td>
<td></td>
<td>1.13.6</td>
<td>Required package for MongoDB plugin mongo-driver lib. Required package for MongoDB plugin mongo-driver lib.</td>
</tr>
<tr>
<td>github.com/xdg/go/pbkdf2</td>
<td></td>
<td>1.0.0</td>
<td>Required package for MongoDB plugin mongo-driver lib. Required package for MongoDB plugin mongo-driver lib.</td>
</tr>
<tr>
<td>github.com/xdg/go/scram</td>
<td></td>
<td>1.0.2</td>
<td>Required package for MongoDB plugin mongo-driver lib.</td>
</tr>
<tr>
<td>github.com/xdg/go/stringprep</td>
<td></td>
<td>1.0.2</td>
<td>Required package for MongoDB plugin mongo-driver lib.</td>
</tr>
<tr>
<td>github.com/youmark/pkcs8</td>
<td></td>
<td>0.0.0</td>
<td>Required package for MongoDB plugin mongo-driver lib.</td>
</tr>
</tbody>
</table>
### Java gateway

If you obtained Zabbix from the source repository or an archive, then the necessary dependencies are already included in the source tree.

If you obtained Zabbix from your distribution’s package, then the necessary dependencies are already provided by the packaging system.

In both cases above, the software is ready to be used and no additional downloads are necessary.

If, however, you wish to provide your versions of these dependencies (for instance, if you are preparing a package for some Linux distribution), below is the list of library versions that Java gateway is known to work with. Zabbix may work with other versions of these libraries, too.

The following table lists JAR files that are currently bundled with Java gateway in the original code:

<table>
<thead>
<tr>
<th>Library</th>
<th>Mandatory status</th>
<th>Minimum version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>android-json</td>
<td>Yes</td>
<td>Version 4.3r1 or higher.</td>
<td>JSON (JavaScript Object Notation) is a lightweight data-interchange format. This is the org.json compatible Android implementation extracted from the Android SDK.</td>
</tr>
<tr>
<td>logback-classic</td>
<td></td>
<td>Version 1.2.9 or higher.</td>
<td></td>
</tr>
<tr>
<td>logback-core</td>
<td></td>
<td>Version 1.2.9 or higher.</td>
<td></td>
</tr>
<tr>
<td>slf4j-api</td>
<td></td>
<td>Version 1.7.32 or higher.</td>
<td></td>
</tr>
</tbody>
</table>

Java gateway can be built using either Oracle Java or open-source OpenJDK (version 1.6 or newer). Packages provided by Zabbix are compiled using OpenJDK. The table below provides information about OpenJDK versions used for building Zabbix packages by distribution:

<table>
<thead>
<tr>
<th>Distribution</th>
<th>OpenJDK version</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHEL/CentOS 8</td>
<td>1.8.0</td>
</tr>
<tr>
<td>RHEL/CentOS 7</td>
<td>1.8.0</td>
</tr>
<tr>
<td>SLES 15</td>
<td>11.0.4</td>
</tr>
<tr>
<td>SLES 12</td>
<td>1.8.0</td>
</tr>
<tr>
<td>Debian 10</td>
<td>11.0.8</td>
</tr>
<tr>
<td>Ubuntu 20.04</td>
<td>11.0.8</td>
</tr>
<tr>
<td>Ubuntu 18.04</td>
<td>11.0.8</td>
</tr>
</tbody>
</table>

**Default port numbers**

The following table lists default port numbers that Zabbix components listen on:

<table>
<thead>
<tr>
<th>Zabbix component</th>
<th>Port number</th>
<th>Protocol</th>
<th>Type of connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zabbix agent</td>
<td>10050</td>
<td>TCP</td>
<td>on demand</td>
</tr>
<tr>
<td>Zabbix server</td>
<td>10051</td>
<td>TCP</td>
<td>on demand</td>
</tr>
<tr>
<td>Zabbix proxy</td>
<td>10051</td>
<td>TCP</td>
<td>on demand</td>
</tr>
<tr>
<td>Zabbix Java gateway</td>
<td>10052</td>
<td>TCP</td>
<td>on demand</td>
</tr>
</tbody>
</table>

**Database size**

Zabbix configuration data require a fixed amount of disk space and do not grow much.

Zabbix database size mainly depends on these variables, which define the amount of stored historical data:

- Number of processed values per second
This is the average number of new values Zabbix server receives every second. For example, if we have 3000 items for monitoring with a refresh rate of 60 seconds, the number of values per second is calculated as \( \frac{3000}{60} = 50 \).

It means that 50 new values are added to Zabbix database every second.

- **Housekeeper settings for history**

Zabbix keeps values for a fixed period of time, normally several weeks or months. Each new value requires a certain amount of disk space for data and index.

So, if we would like to keep 30 days of history and we receive 50 values per second, the total number of values will be around \((30 \times 24 \times 3600) \times 50 = 129,600,000\), or about 130M of values.

Depending on the database engine used, type of received values (floats, integers, strings, log files, etc), the disk space for keeping a single value may vary from 40 bytes to hundreds of bytes. Normally it is around 90 bytes per value for numeric items\(^2\). In our case, it means that 130M of values will require \(130M \times 90 \text{ bytes} = 10.9\text{ GB}\) of disk space.

The size of text/log item values is impossible to predict exactly, but you may expect around 500 bytes per value.

- **Housekeeper setting for trends**

Zabbix keeps a 1-hour max/min/avg/count set of values for each item in the table **trends**. The data is used for trending and long period graphs. The one hour period can not be customized.

Zabbix database, depending on the database type, requires about 90 bytes per each total. Suppose we would like to keep trend data for 5 years. Values for 3000 items will require \(3000 \times 24 \times 365 \times 90 = 2.2\text{ GB}\) per year, or \(11\text{ GB}\) for 5 years.

- **Housekeeper settings for events**

Each Zabbix event requires approximately 250 bytes of disk space\(^1\). It is hard to estimate the number of events generated by Zabbix daily. In the worst-case scenario, we may assume that Zabbix generates one event per second.

For each recovered event, an event_recovery record is created. Normally most of the events will be recovered so we can assume one event_recovery record per event. That means additional 80 bytes per event.

Optionally events can have tags, each tag record requiring approximately 100 bytes of disk space\(^1\). The number of tags per event (#tags) depends on configuration. So each will need an additional #tags \times 100 bytes of disk space.

It means that if we want to keep 3 years of events, this would require \(3 \times 365 \times 24 \times 3600 \times (250 + 80 + \#tags \times 100) = \sim 30\text{ GB} + \#tags \times 100\text{B disk space}\(^2\).

\(^1\) More when having non-ASCII event names, tags and values.

\(^2\) The size approximations are based on MySQL and might be different for other databases.

The table contains formulas that can be used to calculate the disk space required for Zabbix system:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Formula for required disk space (in bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zabbix configuration</td>
<td>Fixed size. Normally 10MB or less.</td>
</tr>
<tr>
<td>History</td>
<td>days*(items/refresh rate)<em>24</em>3600*bytes</td>
</tr>
<tr>
<td></td>
<td>items : number of items</td>
</tr>
<tr>
<td></td>
<td>days : number of days to keep history</td>
</tr>
<tr>
<td></td>
<td>refresh rate : average refresh rate of items</td>
</tr>
<tr>
<td></td>
<td>bytes : number of bytes required to keep single value, depends on database engine, normally \sim 90 bytes.</td>
</tr>
<tr>
<td>Trends</td>
<td>days*(items/3600)<em>24</em>3600*bytes</td>
</tr>
<tr>
<td></td>
<td>items : number of items</td>
</tr>
<tr>
<td></td>
<td>days : number of days to keep history</td>
</tr>
<tr>
<td></td>
<td>bytes : number of bytes required to keep single trend, depends on the database engine, normally \sim 90 bytes.</td>
</tr>
<tr>
<td>Events</td>
<td>days<em>events</em>24<em>3600</em>bytes</td>
</tr>
<tr>
<td></td>
<td>events : number of event per second. One (1) event per second in worst-case scenario.</td>
</tr>
<tr>
<td></td>
<td>days : number of days to keep history</td>
</tr>
<tr>
<td></td>
<td>bytes : number of bytes required to keep single trend, depends on the database engine, normally \sim 330 + average number of tags per event \times 100 bytes.</td>
</tr>
</tbody>
</table>

So, the total required disk space can be calculated as: **Configuration + History + Trends + Events**

The disk space will NOT be used immediately after Zabbix installation. Database size will grow then it will stop growing at some point, which depends on housekeeper settings.
Time synchronization

It is very important to have precise system time on the server with Zabbix running. ntpd is the most popular daemon that synchronizes the host’s time with the time of other machines. It’s strongly recommended to maintain synchronized system time on all systems Zabbix components are running on.

Best practices for secure Zabbix setup

Overview

This section contains best practices that should be observed in order to set up Zabbix in a secure way.

The practices contained here are not required for the functioning of Zabbix. They are recommended for better security of the system.

Access control

Principle of least privilege

The principle of least privilege should be used at all times for Zabbix. This principle means that user accounts (in Zabbix frontend) or process user (for Zabbix server/proxy or agent) have only those privileges that are essential to perform intended functions. In other words, user accounts at all times should run with as few privileges as possible.

Giving extra permissions to ‘zabbix’ user will allow it to access configuration files and execute operations that can compromise the overall security of the infrastructure.

When implementing the least privilege principle for user accounts, Zabbix frontend user types should be taken into account. It is important to understand that while a “Admin” user type has less privileges than “Super Admin” user type, it has administrative permissions that allow managing configuration and execute custom scripts.

Some information is available even for non-privileged users. For example, while Administration → Scripts is not available for non-Super Admins, scripts themselves are available for retrieval by using Zabbix API. Limiting script permissions and not adding sensitive information (like access credentials, etc) should be used to avoid exposure of sensitive information available in global scripts.

Secure user for Zabbix agent

In the default configuration, Zabbix server and Zabbix agent processes share one ‘zabbix’ user. If you wish to make sure that the agent cannot access sensitive details in server configuration (e.g. database login information), the agent should be run as a different user:

1. Create a secure user
2. Specify this user in the agent configuration file (`User` parameter)
3. Restart the agent with administrator privileges. Privileges will be dropped to the specified user.

Revoke write access to SSL configuration file in Windows

Zabbix Windows agent compiled with OpenSSL will try to reach the SSL configuration file in c:\openssl-64bit. The "openssl-64bit" directory on disk C: can be created by non-privileged users.

So for security hardening, it is required to create this directory manually and revoke write access from non-admin users.

Please note that the directory names will be different on 32-bit and 64-bit versions of Windows.

Cryptography

Setting up SSL for Zabbix frontend

On RHEL/Centos, install mod_ssl package:

```
yum install mod_ssl
```

Create directory for SSL keys:

```
mkdir -p /etc/httpd/ssl/private
chmod 700 /etc/httpd/ssl/private
```

Create SSL certificate:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/httpd/ssl/private/apache-selfsigned.key -days 365 -nodes -out /etc/httpd/ssl/private/apache-selfsigned.crt
```

Fill out the prompts appropriately. The most important line is the one that requests the Common Name. You need to enter the domain name that you want to be associated with your server. You can enter the public IP address instead if you do not have a domain name. We will use example.com in this article.
Edit Apache SSL configuration:

/etc/httpd/conf.d/ssl.conf

DocumentRoot "/usr/share/zabbix"
ServerName example.com:443
SSLCertificateFile /etc/httpd/ssl/apache-selfsigned.crt
SSLCertificateKeyFile /etc/httpd/ssl/private/apache-selfsigned.key

Restart the Apache service to apply the changes:

systemctl restart httpd.service

Web server hardening

Enabling Zabbix on root directory of URL

Add a virtual host to Apache configuration and set permanent redirect for document root to Zabbix SSL URL. Do not forget to replace example.com with the actual name of the server.

/etc/httpd/conf/httpd.conf

#Add lines

<VirtualHost *:*>
  ServerName example.com
  Redirect permanent / https://example.com
</VirtualHost>

Restart the Apache service to apply the changes:

systemctl restart httpd.service

Enabling HTTP Strict Transport Security (HSTS) on the web server

To protect Zabbix frontend against protocol downgrade attacks, we recommend to enable HSTS policy on the web server.

For example, to enable HSTS policy for your Zabbix frontend in Apache configuration:

/etc/httpd/conf/httpd.conf

add the following directive to your virtual host’s configuration:

<VirtualHost *:443>
  Header set Strict-Transport-Security "max-age=31536000"
</VirtualHost>

Restart the Apache service to apply the changes:

systemctl restart httpd.service

Disabling web server information exposure

It is recommended to disable all web server signatures as part of the web server hardening process. The web server is exposing software signature by default:
The signature can be disabled by adding two lines to the Apache (used as an example) configuration file:

```plaintext
ServerSignature Off
ServerTokens Prod
```

PHP signature (X-Powered-By HTTP header) can be disabled by changing the php.ini configuration file (signature is disabled by default):

```plaintext
expose_php = Off
```

Web server restart is required for configuration file changes to be applied.

Additional security level can be achieved by using the mod_security (package libapache2-mod-security2) with Apache. mod_security allows to remove server signature instead of only removing version from server signature. Signature can be altered to any value by changing "SecServerSignature" to any desired value after installing mod_security.

Please refer to documentation of your web server to find help on how to remove/change software signatures.

Disabling default web server error pages

It is recommended to disable default error pages to avoid information exposure. Web server is using built-in error pages by default:

```
Not Found
The requested URL /custom-text was not found on this server.

Apache/2.4.18 (Ubuntu) Server at localhost Port 80
```

Default error pages should be replaced/removed as part of the web server hardening process. The "ErrorDocument" directive can be used to define a custom error page/text for Apache web server (used as an example).

Please refer to documentation of your web server to find help on how to replace/remove default error pages.

Removing web server test page

It is recommended to remove the web server test page to avoid information exposure. By default, web server webroot contains a test page called index.html (Apache2 on Ubuntu is used as an example):
The test page should be removed or should be made unavailable as part of the web server hardening process.

Set X-Frame-Options HTTP response header

By default, Zabbix is configured with X-Frame-Options HTTP response header set to SAMEORIGIN, meaning that content can only be loaded in a frame that has the same origin as the page itself.

Zabbix frontend elements that pull content from external URLs (namely, the URL dashboard widget) display retrieved content in a sandbox with all sandboxing restrictions enabled.

These settings enhance the security of the Zabbix frontend and provide protection against XSS and clickjacking attacks. Super Admins can modify iframe sandboxing and X-Frame-Options HTTP response header parameters as needed. Please carefully weigh the risks and benefits before changing default settings. Turning sandboxing or X-Frame-Options off completely is not recommended.

Hiding the file with list of common passwords

To increase the complexity of password brute force attacks, it is suggested to limit access to the file ui/data/top_passwords.txt by modifying web server configuration. This file contains a list of the most common and context-specific passwords, and is used to prevent users from setting such passwords if Avoid easy-to-guess passwords parameter is enabled in the password policy.

For example, on NGINX file access can be limited by using the location directive:

```location = /data/top_passwords.txt {
  deny all;
  return 404;
}
```

On Apache by using .htaccess file:

```<Files "top_passwords.txt">
  Order Allow,Deny
  Deny from all
</Files>
```

UTF-8 encoding

UTF-8 is the only encoding supported by Zabbix. It is known to work without any security flaws. Users should be aware that there are known security issues if using some of the other encodings.

Zabbix Security Advisories and CVE database

See Zabbix Security Advisories and CVE database.

### 3 Installation from sources

You can get the very latest version of Zabbix by compiling it from the sources.

A step-by-step tutorial for installing Zabbix from the sources is provided here.

1 Installing Zabbix daemons

1 Download the source archive

Go to the Zabbix download page and download the source archive. Once downloaded, extract the sources, by running:

```
$ tar -zxvf zabbix-6.2.0.tar.gz
```
Enter the correct Zabbix version in the command. It must match the name of the downloaded archive.

2 Create user account

For all of the Zabbix daemon processes, an unprivileged user is required. If a Zabbix daemon is started from an unprivileged user account, it will run as that user.

However, if a daemon is started from a 'root' account, it will switch to a 'zabbix' user account, which must be present. To create such a user account (in its own group, "zabbix"),

on a RedHat-based system, run:

```
groupadd --system zabbix
useradd --system -g zabbix -d /usr/lib/zabbix -s /sbin/nologin -c "Zabbix Monitoring System" zabbix
```

on a Debian-based system, run:

```
addgroup --system --quiet zabbix
adduser --quiet --system --disabled-login --ingroup zabbix --home /var/lib/zabbix --no-create-home zabbix
```

Zabbix processes do not need a home directory, which is why we do not recommend creating it. However, if you are using some functionality that requires it (e.g., store MySQL credentials in $HOME/.my.cnf) you are free to create it using the following commands.

On RedHat-based systems, run:

```
mkdir -m u=rwx,g=rwx,o= -p /usr/lib/zabbix
chown zabbix:zabbix /usr/lib/zabbix
```

On Debian-based systems, run:

```
mkdir -m u=rwx,g=rwx,o= -p /var/lib/zabbix
chown zabbix:zabbix /var/lib/zabbix
```

A separate user account is not required for Zabbix frontend installation.

If Zabbix server and agent are run on the same machine it is recommended to use a different user for running the server than for running the agent. Otherwise, if both are run as the same user, the agent can access the server configuration file and any Admin level user in Zabbix can quite easily retrieve, for example, the database password.

Running Zabbix as root, bin, or any other account with special rights is a security risk.

3 Create Zabbix database

For Zabbix server and proxy daemons, as well as Zabbix frontend, a database is required. It is not needed to run Zabbix agent.

SQL scripts are provided for creating database schema and inserting the dataset. Zabbix proxy database needs only the schema while Zabbix server database requires also the dataset on top of the schema.

Having created a Zabbix database, proceed to the following steps of compiling Zabbix.

4 Configure the sources

When configuring the sources for a Zabbix server or proxy, you must specify the database type to be used. Only one database type can be compiled with a server or proxy process at a time.

To see all of the supported configuration options, inside the extracted Zabbix source directory run:

```
./configure --help
```

To configure the sources for a Zabbix server and agent, you may run something like:

```
./configure --enable-server --enable-agent --with-mysql --enable-ipv6 --with-net-snmp --with-libcurl --with-openipmi
```

To configure the sources for a Zabbix server (with PostgreSQL etc.), you may run:

```
./configure --enable-server --with-postgresql --with-net-snmp
```

To configure the sources for a Zabbix proxy (with SQLite etc.), you may run:

```
./configure --prefix=/usr --enable-proxy --with-net-snmp --with-sqlite3 --with-ssh2
```

To configure the sources for a Zabbix agent, you may run:

```
./configure --enable-agent
```

or, for Zabbix agent 2:
A configured Go environment with a currently supported Go version is required for building Zabbix agent 2. See golang.org for installation instructions.

Notes on compilation options:

- Command-line utilities zabbix_get and zabbix_sender are compiled if --enable-agent option is used.
- --with-libcurl and --with-libxml2 configuration options are required for virtual machine monitoring; --with-libcurl is also required for SMTP authentication and web.page.* Zabbix agent items. Note that cURL 7.20.0 or higher is required with the --with-libcurl configuration option.
- Zabbix always compiles with the PCRE library (since version 3.4.0); installing it is not optional. --with-libpcre=[DIR] only allows pointing to a specific base install directory, instead of searching through a number of common places for the libpcre files.
- You may use the --enable-static flag to statically link libraries. If you plan to distribute compiled binaries among different servers, you must use this flag to make these binaries work without required libraries. Note that --enable-static does not work in Solaris.
- Using --enable-static option is not recommended when building server. In order to build the server statically, you must have a static version of every external library needed. There is no strict check for that in configure script.
- Add optional path to the MySQL configuration file --with-mysql=/<path_to_the_file>/mysql_config to select the desired MySQL client library when there is a need to use one that is not located in the default location. It is useful when there are several versions of MySQL installed or MariaDB installed alongside MySQL on the same system.
- Use --with-oracle flag to specify location of the OCI API.

If ./configure fails due to missing libraries or some other circumstance, please see the config.log file for more details on the error. For example, if libssl is missing, the immediate error message may be misleading:

checking for main in -lmysqlclient... no
configure: error: Not found mysqlclient library

While config.log has a more detailed description:

/usr/bin/ld: cannot find -lssl
/usr/bin/ld: cannot find -lcrypto

See also:
- Compiling Zabbix with encryption support for encryption support
- Known issues with compiling Zabbix agent on HP-UX

5 Make and install everything

If installing from Zabbix Git repository, it is required to run first:

$ make dbschema

make install

This step should be run as a user with sufficient permissions (commonly 'root', or by using sudo).

Running make install will by default install the daemon binaries (zabbix_server, zabbix_agentd, zabbix_proxy) in /usr/local/sbin and the client binaries (zabbix_get, zabbix_sender) in /usr/local/bin.

To specify a different location than /usr/local, use a --prefix key in the previous step of configuring sources, for example --prefix=/home/zabbix. In this case daemon binaries will be installed under <prefix>/sbin, while utilities under <prefix>/bin. Man pages will be installed under <prefix>/share.

6 Review and edit configuration files

- edit the Zabbix agent configuration file /usr/local/etc/zabbix_agentd.conf
- edit the Zabbix server configuration file /usr/local/etc/zabbix_server.conf
- edit the Zabbix proxy configuration file /usr/local/etc/zabbix_proxy.conf

You need to configure this file for every host with zabbix_agentd installed.

You must specify the Zabbix server IP address in the file. Connections from other hosts will be denied.

You must specify the database name, user and password (if using any).

The rest of the parameters will suit you with their defaults if you have a small installation (up to ten monitored hosts). You should change the default parameters if you want to maximize the performance of Zabbix server (or proxy) though. See the performance tuning section for more details.

- If you have installed a Zabbix proxy, edit the proxy configuration file /usr/local/etc/zabbix_proxy.conf
You must specify the server IP address and proxy hostname (must be known to the server), as well as the database name, user and password (if using any).

With SQLite the full path to database file must be specified; DB user and password are not required.

7 Start up the daemons

Run zabbix_server on the server side.

    shell> zabbix_server

Make sure that your system allows allocation of 36MB (or a bit more) of shared memory, otherwise the server may not start and you will see "Cannot allocate shared memory for <type of cache>." in the server log file. This may happen on FreeBSD, Solaris 8. See the "See also" section at the bottom of this page to find out how to configure shared memory.

Run zabbix_agentd on all the monitored machines.

    shell> zabbix_agentd

Make sure that your system allows allocation of 2MB of shared memory, otherwise the agent may not start and you will see "Cannot allocate shared memory for collector." in the agent log file. This may happen on Solaris 8.

If you have installed Zabbix proxy, run zabbix_proxy.

    shell> zabbix_proxy

2 Installing Zabbix web interface

Copying PHP files

Zabbix frontend is written in PHP, so to run it a PHP supported webserver is needed. Installation is done by simply copying the PHP files from the ui directory to the webserver HTML documents directory.

Common locations of HTML documents directories for Apache web servers include:

- /usr/local/apache2/htdocs (default directory when installing Apache from source)
- /srv/www/htdocs (OpenSUSE, SLES)
- /var/www/html (Debian, Ubuntu, Fedora, RHEL, CentOS)

It is suggested to use a subdirectory instead of the HTML root. To create a subdirectory and copy Zabbix frontend files into it, execute the following commands, replacing the actual directory:

    mkdir <htdocs>/zabbix
    cd ui
    cp -a . <htdocs>/zabbix

If planning to use any other language than English, see Installation of additional frontend languages for instructions.

Installing frontend

Please see Web interface installation page for information about Zabbix frontend installation wizard.

3 Installing Java gateway

It is required to install Java gateway only if you want to monitor JMX applications. Java gateway is lightweight and does not require a database.

To install from sources, first download and extract the source archive.

To compile Java gateway, run the ./configure script with --enable-java option. It is advisable that you specify the --prefix option to request installation path other than the default /usr/local, because installing Java gateway will create a whole directory tree, not just a single executable.

    $ ./configure --enable-java --prefix=$PREFIX

To compile and package Java gateway into a JAR file, run make. Note that for this step you will need javac and jar executables in your path.

    $ make

Now you have a zabbix-java-gateway-$VERSION.jar file in src/zabbix_java/bin. If you are comfortable with running Java gateway from src/zabbix_java in the distribution directory, then you can proceed to instructions for configuring and running Java gateway. Otherwise, make sure you have enough privileges and run make install.

    $ make install
Proceed to setup for more details on configuring and running Java gateway.

4 Installing Zabbix web service

Installing Zabbix web service is only required if you want to use scheduled reports.

To install from sources, first download and extract the source archive.

To compile Zabbix web service, run the ./configure script with --enable-webservice option.

A configured Go version 1.13+ environment is required for building Zabbix web service.

Run zabbix_web_service on the machine, where the web service is installed:

shell> zabbix_web_service

Proceed to setup for more details on configuring Scheduled reports generation.

See also

1. How to configure shared memory for Zabbix daemons

Building Zabbix agent 2 on Windows

Overview

This section demonstrates how to build Zabbix agent 2 (Windows) from sources.

Installing MinGW Compiler

1. Download MinGW-w64 with SJLJ (set jump/long jump) Exception Handling and Windows threads (for example x86_64-8.1.0-release-win32-sjlj-y6-rev0.7z)
2. Extract and move to c:\mingw
3. Setup environmental variable

@echo off
set PATH=%PATH%;c:\mingw\bin
cmd

When compiling use Windows prompt instead of MSYS terminal provided by MinGW

Compiling PCRE development libraries

The following instructions will compile and install 64-bit PCRE libraries in c:\dev\pcre and 32-bit libraries in c:\dev\pcre32:

1. Download PCRE library version 8.XX from pcre.org (http://ftp.pcre.org/pub/pcre/) and extract
2. Open cmd and navigate to the extracted sources

Build 64bit PCRE

1. Delete old configuration/cache if exists:
   del CMakeCache.txt
   rmdir /q /s CMakeFiles
2. Run cmake (CMake can be installed from https://cmake.org/download/):
   cmake -G "MinGW Makefiles" -DCMAKE_C_COMPILER=gcc -DCMAKE_C_FLAGS="-O2 -g" -DCMAKE_CXX_FLAGS="-O2 -g" -DCMAKE_EXE_LINKER_FLAGS="-Wl,-mi386pe"
3. Next, run:
   mingw32-make clean
   mingw32-make install

Build 32bit PCRE

1. Run:
   mingw32-make clean
2. Delete CMakeCache.txt:
   del CMakeCache.txt
   rmdir /q /s CMakeFiles
3. Run cmake:
   cmake -G "MinGW Makefiles" -DCMAKE_C_COMPILER=gcc -DCMAKE_C_FLAGS="-m32 -O2 -g" -DCMAKE_CXX_FLAGS="-m32 -O2 -g" -DCMAKE_EXE_LINKER_FLAGS="-Wl,-mi386pe"
4. Next, run:

```bash
mingw32-make install
```

Installing OpenSSL development libraries

1. Download 32 and 64 bit builds from [https://curl.se/windows/](https://curl.se/windows/)
2. Extract files into `c:\dev\openssl32` and `c:\dev\openssl` directories accordingly.
3. After that, remove extracted `*.dll.a` (dll call wrapper libraries) as MinGW prioritizes them before static libraries.

Compiling Zabbix agent 2

32 bit

Open MinGW environment (Windows command prompt) and navigate to `build/mingw` directory in the Zabbix source tree.

Run:

```bash
mingw32-make clean
mingw32-make ARCH=x86 PCRE=c:\dev\pcre32 OPENSSL=c:\dev\openssl32
```

64 bit

Open MinGW environment (Windows command prompt) and navigate to `build/mingw` directory in the Zabbix source tree.

Run:

```bash
mingw32-make clean
mingw32-make PCRE=c:\dev\pcre OPENSSL=c:\dev\openssl
```

Both 32- and 64-bit versions can be built on a 64-bit platform, but only a 32-bit version can be built on a 32-bit platform. When working on the 32-bit platform, follow the same steps as for 64-bit version on 64-bit platform.

**Building Zabbix agent on macOS**

Overview

This section demonstrates how to build Zabbix macOS agent binaries from sources with or without TLS.

Prerequisites

You will need command line developer tools (Xcode is not required), Automake, pkg-config and PCRE (v8.x) or PCRE2 (v10.x). If you want to build agent binaries with TLS, you will also need OpenSSL or GnuTLS.

To install Automake and pkg-config, you will need a Homebrew package manager from [https://brew.sh/](https://brew.sh/). To install it, open terminal and run the following command:

```bash
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Then install Automake and pkg-config:

```bash
$ brew install automake
$ brew install pkg-config
```

Preparing PCRE, OpenSSL and GnuTLS libraries depends on the way how they are going to be linked to the agent.

If you intend to run agent binaries on a macOS machine that already has these libraries, you can use precompiled libraries that are provided by Homebrew. These are typically macOS machines that use Homebrew for building Zabbix agent binaries or for other purposes.

If agent binaries will be used on macOS machines that don’t have the shared version of libraries, you should compile static libraries from sources and link Zabbix agent with them.

Building agent binaries with shared libraries

Install PCRE2 (replace `pcre2` with `pcre` in the commands below, if needed):

```bash
$ brew install pcre2
```

When building with TLS, install OpenSSL and/or GnuTLS:

```bash
$ brew install openssl
$ brew install gnutls
```

Download Zabbix source:

```bash
$ git clone https://git.zabbix.com/scm/zbx/zabbix.git
```
Build agent without TLS:

```
$ cd zabbix
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6
$ make
$ make install
```

Build agent with OpenSSL:

```
$ cd zabbix
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-openssl=/usr/local/opt/openssl
$ make
$ make install
```

Build agent with GnuTLS:

```
$ cd zabbix-source/
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-gnutls=/usr/local/opt/gnutls
$ make
$ make install
```

Building agent binaries with static libraries without TLS

Let's assume that PCRE static libraries will be installed in $HOME/static-libs. We will use PCRE2 10.39.

```
$ PCRE_PREFIX="$HOME/static-libs/pcre2-10.39"
```

Download and build PCRE with Unicode properties support:

```
$ mkdir static-libs-source
$ cd static-libs-source
$ curl --remote-name https://github.com/PhilipHazel/pcre2/releases/download/pcre2-10.39/pcre2-10.39.tar.gz
$ tar xf pcre2-10.39.tar.gz
$ cd pcre2-10.39
$ ./configure --prefix="$PCRE_PREFIX" --disable-shared --enable-static --enable-unicode-properties
$ make
$ make check
$ make install
```

Download Zabbix source and build agent:

```
$ git clone https://git.zabbix.com/scm/zbx/zabbix.git
$ cd zabbix
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre2="$PCRE_PREFIX"
$ make
$ make install
```

Building agent binaries with static libraries with OpenSSL

When building OpenSSL, it's recommended to run `make test` after successful building. Even if building was successful, tests sometimes fail. If this is the case, problems should be researched and resolved before continuing.

Let's assume that PCRE and OpenSSL static libraries will be installed in $HOME/static-libs. We will use PCRE2 10.39 and OpenSSL 1.1.1a.

```
$ PCRE_PREFIX="$HOME/static-libs/pcre2-10.39"
$ OPENSSL_PREFIX="$HOME/static-libs/openssl-1.1.1a"
```

Let's build static libraries in static-libs-source:

```
$ mkdir static-libs-source
$ cd static-libs-source
```

Download and build PCRE with Unicode properties support:

```
$ curl --remote-name https://github.com/PhilipHazel/pcre2/releases/download/pcre2-10.39/pcre2-10.39.tar.gz
$ tar xf pcre2-10.39.tar.gz
$ cd pcre2-10.39
$ ./configure --prefix="$PCRE_PREFIX" --disable-shared --enable-static --enable-unicode-properties
```
$ make
$ make check
$ make install
$ cd ..

Download and build OpenSSL:

$ curl --remote-name https://www.openssl.org/source/openssl-1.1.1a.tar.gz
$ tar xf openssl-1.1.1a.tar.gz
$ cd openssl-1.1.1a
$ ./Configure --prefix="$OPENSSL_PREFIX" --openssldir="$OPENSSL_PREFIX" --api=1.1.0 no-shared no-capieng no-srp no-gost no-dgram no-dtls1-method no-dtls1_2-method darwin64-x86_64-cc
$ make
$ make test
$ make install_sw
$ cd ..

Download Zabbix source and build agent:

$ git clone https://git.zabbix.com/scm/zbx/zabbix.git
$ cd zabbix
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre2="$PCRE_PREFIX" --with-openssl="$OPENSSL_PREFIX"
$ make
$ make install

Building agent binaries with static libraries with GnuTLS

GnuTLS depends on the Nettle crypto backend and GMP arithmetic library. Instead of using full GMP library, this guide will use mini-gmp which is included in Nettle.

When building GnuTLS and Nettle, it's recommended to run make check after successful building. Even if building was successful, tests sometimes fail. If this is the case, problems should be researched and resolved before continuing.

Let's assume that PCRE, Nettle and GnuTLS static libraries will be installed in $HOME/static-libs. We will use PCRE 10.39, Nettle 3.4.1 and GnuTLS 3.6.5.

$ PCRE_PREFIX="$HOME/static-libs/pcre2-10.39"
$ NETTLE_PREFIX="$HOME/static-libs/nettle-3.4.1"
$ GNUTLS_PREFIX="$HOME/static-libs/gnutls-3.6.5"

Let's build static libraries in static-libs-source:

$ mkdir static-libs-source
$ cd static-libs-source

Download and build Nettle:

$ curl --remote-name https://ftp.gnu.org/gnu/nettle/nettle-3.4.1.tar.gz
$ tar xf nettle-3.4.1.tar.gz
$ cd nettle-3.4.1
$ ./configure --prefix="$NETTLE_PREFIX" --enable-static --disable-shared --disable-documentation --disable-assembler --enable-x86-aesni --enable-mini-gmp
$ make
$ make check
$ make install
$ cd ..

Download and build GnuTLS:

$ curl --remote-name https://www.gnupg.org/ftp/gcrypt/gnutls/v3.6/gnutls-3.6.5.tar.xz
$ tar xf gnutls-3.6.5.tar.xz
$ cd gnutls-3.6.5
$ PKG_CONFIG_PATH="$NETTLE_PREFIX/lib/pkgconfig" ./configure --prefix="$GNUTLS_PREFIX" --enable-static --disable-shared --disable-documentation --disable-egenius --disable-sslv23 --disable-java --with-nettle-mini
$ make
$ make check
$ make install
$ cd ..

Download Zabbix source and build agent:

$ git clone https://git.zabbix.com/scm/zbx/zabbix.git
$ cd zabbix
Building Zabbix agent on Windows

Overview

This section demonstrates how to build Zabbix Windows agent binaries from sources with or without TLS.

Compiling OpenSSL

The following steps will help you to compile OpenSSL from sources on MS Windows 10 (64-bit).

1. For compiling OpenSSL you will need on Windows machine:
   1. C compiler (e.g. VS 2017 RC),
   2. NASM (https://www.nasm.us/),
   3. Perl (e.g. Strawberry Perl from http://strawberryperl.com/),
   4. Perl module Text::Template (cpan Text::Template).
2. Get OpenSSL sources from https://www.openssl.org/. OpenSSL 1.1.1 is used here.
3. Unpack OpenSSL sources, for example, in E:\openssl-1.1.1.
4. Open a command line window e.g. the x64 Native Tools Command Prompt for VS 2017 RC.
5. Go to the OpenSSL source directory, e.g. E:\openssl-1.1.1.
   1. Verify that NASM can be found: e:\openssl-1.1.1> nasm --version NASM version 2.13.01 compiled on May 1 2017
6. Configure OpenSSL, for example: e:\openssl-1.1.1> perl E:\openssl-1.1.1\Configure VC-WIN64A no-shared no-capieng no-srp no-gost no-dgram no-dtls1-method no-dtls1_2-method --api=1.1.0 --prefix=C:\OpenSSL --openssldir=C:\OpenSSL-Win64-111-static
   • Note the option ‘no-shared’: if ‘no-shared’ is used then the OpenSSL static libraries libcrypto.lib and libssl.lib will be ‘self-sufficient’ and resulting Zabbix binaries will include OpenSSL in themselves, no need for external OpenSSL DLLs. Advantage: Zabbix binaries can be copied to other Windows machines without OpenSSL libraries. Disadvantage: when a new OpenSSL bugfix version is released, Zabbix agent needs to be recompiled and reinstalled.
   • If ‘no-shared’ is not used, then the static libraries libcrypto.lib and libssl.lib will be using OpenSSL DLLs at runtime. Advantage: when a new OpenSSL bugfix version is released, probably you can upgrade only OpenSSL DLLs, without recompiling Zabbix agent. Disadvantage: copying Zabbix agent to another machine requires copying OpenSSL DLLs, too.
7. Compile OpenSSL, run tests, install: e:\openssl-1.1.1> nmake e:\openssl-1.1.1> nmake test ...
   All tests successful. Files=152, Tests=1152, 501 wallclock secs ( 0.67 usr + 0.61 sys = 1.28 CPU) Result: PASS e:\openssl-1.1.1> nmake install_sw
   ‘install_sw’ installs only software components (i.e. libraries, header files, but no documentation). If you want everything, use “nmake install”.

Compiling PCRE

1. Download PCRE or PCRE2 (supported since Zabbix 6.0) library from pcre.org repository: (https://github.com/PhilipHazel/pcre2/releases/download/pcre2-10.39/pcre2-10.39.39.zip)
2. Extract to directory E:\pcre2-10.39
3. Install CMake from https://cmake.org/download/, during install select: and ensure that cmake\bin is on your path (tested version 3.9.4).
4. Create a new, empty build directory, preferably a subdirectory of the source dir. For example, E:\pcre2-10.39\build.
5. Open a command line window e.g. the x64 Native Tools Command Prompt for VS 2017 and from that shell environment run cmake-gui. Do not try to start Cmake from the Windows Start menu, as this can lead to errors.
6. Enter E:\pcre2-10.39 and E:\pcre2-10.39\build for the source and build directories, respectively.
7. Hit the “Configure” button.
8. When specifying the generator for this project select “NMak Makefiles”.
9. Create a new, empty install directory. For example, E:\pcre2-10.39\install.
10. The GUI will then list several configuration options. Make sure the following options are selected:
    • PCRE_SUPPORT_UNICODE_PROPERTIES ON
    • PCRE_SUPPORT_UTF ON
    • CMAKE_INSTALL_PREFIX E:\pcre2-10.39-99
11. Hit “Configure” again. The adjacent “Generate” button should now be active.
12. Hit “Generate”.

$ ./bootstrap.sh
$ CFLAGS="-Wno-unused-command-line-argument -framework Foundation -framework Security" \
> LIBS="-lgnutls -lhogweed -lnettle" \
> LDFLAGS="-L$GNUTLS_PREFIX/lib -L$NETTLE_PREFIX/lib" \
> ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre2="$PCRE_PREFIX" \
$ make \
$ make install
13. In the event that errors occur, it is recommended that you delete the CMake cache before attempting to repeat the CMake build process. In the CMake GUI, the cache can be deleted by selecting “File > Delete Cache”.

14. The build directory should now contain a usable build system - Makefile.

15. Open a commandline window e.g. the x64 Native Tools Command Prompt for VS 2017 and navigate to the Makefile mentioned above.

16. Run NMake command: E:\pcre2-10.39\build> nmake install

Compiling Zabbix

The following steps will help you to compile Zabbix from sources on MS Windows 10 (64-bit). When compiling Zabbix with/without TLS support the only significant difference is in step 4.

1. On a Linux machine check out the source from git:
   $ git clone https://git.zabbix.com/scm/zbx/zabbix.git
   $ cd zabbix
   $ ./bootstrap.sh
   $ ./configure --enable-agent --enable-ipv6 --prefix=`pwd`
   $ make dbschema $ make dist

2. Copy and unpack the archive, e.g. zabbix-4.4.0.tar.gz, on a Windows machine.

3. Let’s assume that sources are in e:\zabbix-4.4.0. Open a commandlinewindow e.g. the x64 Native Tools Command Prompt for VS 2017 RC. Go to e:\zabbix-4.4.0\build\win32\project.

   - without TLS:
     E:\zabbix-4.4.0\build\win32\project> nmake /K PCREINCDIR=E:\pcre2-10.39-install\include
     PCRELIBDIR=E:\pcre2-10.39-install\bin
   - with TLS:
     E:\zabbix-4.4.0\build\win32\project> nmake /K -f Makefile_get TLS=openssl TSLIBDIR=C:\OpenSSL-Win64-111-static\include PCREINCDIR=E:\pcre2-10.39-install\include
     PCRELIBDIR=E:\pcre2-10.39-install\lib
     E:\zabbix-4.4.0\build\win32\project> nmake /K -f Makefile_sender TLS=openssl TSLIBDIR=C:\OpenSSL-Win64-111-static\include
     PCREINCDIR=E:\pcre2-10.39-install\include
     PCRELIBDIR=E:\pcre2-10.39-install\lib
     E:\zabbix-4.4.0\build\win32\project> nmake /K -f Makefile_agent TLS=openssl TSLIBDIR=C:\OpenSSL-Win64-111-static\include
     PCREINCDIR=E:\pcre2-10.39-install\include
     PCRELIBDIR=E:\pcre2-10.39-install\lib

5. New binaries are located in e:\zabbix-4.4.0\bin\win64. Since OpenSSL was compiled with ‘no-shared’ option, Zabbix binaries contain OpenSSL within themselves and can be copied to other machines that do not have OpenSSL.

Compiling Zabbix with LibreSSL

The process is similar to compiling with OpenSSL, but you need to make small changes in files located in the build\win32\project directory:

* In 'Makefile_tls' delete '/DHAVE_OPENSSL_WITH_PSK'. i.e. find <code>CFLAGS = $(CFLAGS) /D!HAVE_OPENSSL /D!HAVE_OPENSSL_WITH_PSK</code> and replace it with <code>CFLAGS = $(CFLAGS) /DHAVE_OPENSSL</code>

* In 'Makefile_common.inc' add '/NODEFAULTLIB:LIBCMT' i.e. find <code>/MANIFESTUAC:"level='asInvoker' uiAccess='false'" /DYNAMICBASE:NO /PDB:$(TARGETDIR)$/$(TARGETNAME).pdb</code> and replace it with <code>/MANIFESTUAC:"level='asInvoker' uiAccess='false'" /DYNAMICBASE:NO /PDB:$(TARGETDIR)$(TARGETNAME)$NODEFAULTLIB:LIBCMT</code>

4 Installation from packages

From Zabbix official repository

Zabbix SIA provides official RPM and DEB packages for:

- Red Hat Enterprise Linux/CentOS
- Debian/Ubuntu/Raspbian
- SUSE Linux Enterprise Server

Package files for yum/dnf, apt and zypper repositories for various OS distributions are available at repo.zabbix.com.

Note, that though some OS distributions (in particular, Debian-based distributions) provide their own Zabbix packages, these packages are not supported by Zabbix. Zabbix packages provided by 3rd parties can be out of date and may lack the latest features and bug fixes. It is recommended to use only official packages from repo.zabbix.com. If you have previously used unofficial Zabbix packages, see notes about upgrading Zabbix packages from OS repositories.
Official Zabbix packages are available for:

| RHEL 8, CentOS 8 and Oracle Linux 8 | Download |

Packages are available with either MySQL/PostgreSQL database and Apache/Nginx webserver support.

Zabbix 6.2 is not released yet. The download links lead to pre-6.2 packages.

Verify CA encryption mode doesn’t work on RHEL 7 with MySQL due to older MySQL libraries.

Zabbix agent packages and utilities Zabbix get and Zabbix sender are available for RHEL 7, RHEL 6 and RHEL 5 as well.

Zabbix official repository provides fping and libssh2 packages as well. These packages are located in the non-supported directory.

Notes on installation

See installation instructions per platform in the download page for:

- installing the repository
- installing server/agent/frontend
- creating initial database, importing initial data
- configuring database for Zabbix server
- configuring PHP for Zabbix frontend
- starting server/agent processes
- configuring Zabbix frontend

If you want to run Zabbix agent as root, see Running agent as root.

Zabbix web service process, which is used for scheduled report generation, requires Google Chrome browser. The browser is not included into packages and has to be installed manually.

Importing data with TimescaleDB

With TimescaleDB, in addition to the import command for PostgreSQL, also run:

```
# cat /usr/share/doc/zabbix-sql-scripts/postgresql/timescaledb.sql | sudo -u zabbix psql zabbix
```

TimescaleDB is supported with Zabbix server only.

SElinux configuration

Zabbix uses socket-based inter-process communication. On systems where SELinux is enabled, it may be required to add SELinux rules to allow Zabbix create/use UNIX domain sockets in the SocketDir directory. Currently socket files are used by server (alerter, preprocessing, IPMI) and proxy (IPMI). Socket files are persistent, meaning they are present while the process is running.

Having SElinux status enabled in enforcing mode, you need to execute the following commands to enable communication between Zabbix frontend and server:

**RHEL 7 and later:**

```bash
# setsebool -P httpd_can_connect_zabbix on
```

If the database is accessible over network (including 'localhost' in case of PostgreSQL), you need to allow:

```bash
# setsebool -P httpd_can_network_connect_db on
```

**RHEL prior to 7:**

```bash
# setsebool -P httpd_can_network_connect on
# setsebool -P zabbix_can_network on
```

After the frontend and SELinux configuration is done, restart the Apache web server:

```bash
# service httpd restart
```

In addition, Zabbix provides the zabbix-selinux-policy package as part of source RPM packages for RHEL 8 and RHEL 7. This package provides a basic default policy for SELinux and makes zabbix components work out-of-the-box by allowing Zabbix to create and use sockets and enabling httpd connection to PostgreSQL (used by frontend).

The source zabbix_policy.te file contains the following rules:
module zabbix_policy 1.2;

require {
    type zabbix_t;
    type zabbix_port_t;
    type zabbix_var_run_t;
    type postgresql_port_t;
    type httpd_t;
    class tcp_socket name_connect;
    class sock_file { create unlink };
    class unix_stream_socket connectto;
}

#========== zabbix_t ==========
allow zabbix_t self:unix_stream_socket connectto;
allow zabbix_t zabbix_port_t:tcp_socket name_connect;
allow zabbix_t zabbix_var_run_t:sock_file create;
allow zabbix_t zabbix_var_run_t:sock_file unlink;
allow httpd_t zabbix_port_t:tcp_socket name_connect;

#========== httpd_t ==========
allow httpd_t postgresql_port_t:tcp_socket name_connect;

This package has been created to prevent users from turning off SELinux because of the configuration complexity. It contains the default policy that is sufficient to speed up Zabbix deployment and configuration. For maximum security level, it is recommended to set custom SELinux settings.

Proxy installation

Once the required repository is added, you can install Zabbix proxy by running:

```
# dnf install zabbix-proxy-mysql zabbix-sql-scripts
```

Substitute `mysql` in the commands with `pgsql` to use PostgreSQL, or with `sqlite3` to use SQLite3 (proxy only).

The package `zabbix-sql-scripts` contains database schemas for all supported database management systems for both Zabbix server and Zabbix proxy and will be used for data import.

Creating database

Create a separate database for Zabbix proxy.

Zabbix server and Zabbix proxy cannot use the same database. If they are installed on the same host, the proxy database must have a different name.

Importing data

Import initial schema:

```
# cat /usr/share/doc/zabbix-sql-scripts/mysql/proxy.sql | mysql -uzabbix -p zabbix
```

For proxy with PostgreSQL (or SQLite):

```
# cat /usr/share/doc/zabbix-sql-scripts/postgresql/proxy.sql | sudo -u zabbix psql zabbix
# cat /usr/share/doc/zabbix-sql-scripts/sqlite3/proxy.sql | sqlite3 zabbix.db
```

Configure database for Zabbix proxy

Edit zabbix_proxy.conf:

```
# vi /etc/zabbix/zabbix_proxy.conf
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=<password>
```

In DBName for Zabbix proxy use a separate database from Zabbix server.

In DBPassword use Zabbix database password for MySQL; PosgreSQL user password for PosgreSQL.

Use DBHost= with PostgreSQL. You might want to keep the default setting DBHost=localhost (or an IP address), but this would make PostgreSQL use a network socket for connecting to Zabbix. See SELinux configuration for instructions.
Starting Zabbix proxy process

To start a Zabbix proxy process and make it start at system boot:

```
# service zabbix-proxy start
# systemctl enable zabbix-proxy
```

Frontend configuration

A Zabbix proxy does not have a frontend; it communicates with Zabbix server only.

Java gateway installation

It is required to install `Java gateway` only if you want to monitor JMX applications. Java gateway is lightweight and does not require a database.

Once the required repository is added, you can install Zabbix Java gateway by running:

```
# dnf install zabbix-java-gateway
```

Proceed to setup for more details on configuring and running Java gateway.

Installing debuginfo packages

Debuginfo packages are currently available for RHEL/CentOS versions 9, 7, 6 and 5.

To enable debuginfo repository, edit `/etc/yum.repos.d/zabbix.repo` file. Change `enabled=0` to `enabled=1` for zabbix-debuginfo repository.

```
[zabbix-debuginfo]
name=Zabbix Official Repository debuginfo - $basearch
baseurl=http://repo.zabbix.com/zabbix/6.2/rhel/7/$basearch/debuginfo/
defined=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-ZABBIX-A14FE591
gpgcheck=1
```

This will allow you to install the zabbix-debuginfo package.

```
# yum install zabbix-debuginfo
```

This single package contains debug information for all binary Zabbix components.

## 2 Debian/Ubuntu/Raspbian

Overview

Official Zabbix packages are available for:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Download</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debian 11 (Bullseye)</td>
<td>Download</td>
</tr>
<tr>
<td>Debian 10 (Buster)</td>
<td>Download</td>
</tr>
<tr>
<td>Debian 9 (Stretch)</td>
<td>Download</td>
</tr>
<tr>
<td>Ubuntu 20.04 (Focal Fossa) LTS</td>
<td>Download</td>
</tr>
<tr>
<td>Ubuntu 18.04 (Bionic Beaver) LTS</td>
<td>Download</td>
</tr>
<tr>
<td>Ubuntu 16.04 (Xenial Xerus) LTS</td>
<td>Download</td>
</tr>
<tr>
<td>Ubuntu 14.04 (Trusty Tahr) LTS</td>
<td>Download</td>
</tr>
<tr>
<td>Raspbian 11 (Bullseye)</td>
<td>Download</td>
</tr>
<tr>
<td>Raspbian 10 (Buster)</td>
<td>Download</td>
</tr>
<tr>
<td>Raspbian 9 (Stretch)</td>
<td>Download</td>
</tr>
</tbody>
</table>

Packages are available with either MySQL/PostgreSQL database and Apache/NGINX webserver support.

Notes on installation

See the installation instructions per platform in the download page for:

- installing the repository
- installing server/agent/frontend
- creating initial database, importing initial data
- configuring database for Zabbix server
- configuring PHP for Zabbix frontend
- starting server/agent processes
• configuring Zabbix frontend

If you want to run Zabbix agent as root, see running agent as root.

Zabbix web service process, which is used for scheduled report generation, requires Google Chrome browser. The browser is not included into packages and has to be installed manually.

Importing data with Timescale DB

With TimescaleDB, in addition to the import command for PostgreSQL, also run:

```
# cat /usr/share/doc/zabbix-sql-scripts/postgresql/timescaledb.sql | sudo -u zabbix psql zabbix
```

TimescaleDB is supported with Zabbix server only.

SELinux configuration

See SELinux configuration for RHEL/CentOS.

After the frontend and SELinux configuration is done, restart the Apache web server:

```
# service apache2 restart
```

Proxy installation

Once the required repository is added, you can install Zabbix proxy by running:

```
# apt install zabbix-proxy-mysql zabbix-sql-scripts
```

Substitute 'mysql' in the command with 'pgsql' to use PostgreSQL, or with 'sqlite3' to use SQLite3.

The package `zabbix-sql-scripts` contains database schemas for all supported database management systems for both Zabbix server and Zabbix proxy and will be used for data import.

Creating database

Create a separated database for Zabbix proxy.

Zabbix server and Zabbix proxy cannot use the same database. If they are installed on the same host, the proxy database must have a different name.

Importing data

Import initial schema:

```
# cat /usr/share/doc/zabbix-sql-scripts/mysql/proxy.sql | mysql -uzabbix -p zabbix
```

For proxy with PostgreSQL (or SQLite):

```
# cat /usr/share/doc/zabbix-sql-scripts/postgresql/proxy.sql | sudo -u zabbix psql zabbix
# cat /usr/share/doc/zabbix-sql-scripts/sqlite3/proxy.sql | sqlite3 zabbix.db
```

Configure database for Zabbix proxy

Edit zabbix_proxy.conf:

```
# vi /etc/zabbix/zabbix_proxy.conf
```

```
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=<password>
```

In DBName for Zabbix proxy use a separate database from Zabbix server.

In DBPassword use Zabbix database password for MySQL; Postgresql user password for Postgresql.

Use DBHost= with PostgreSQL. You might want to keep the default setting DBHost=localhost (or an IP address), but this would make PostgreSQL use a network socket for connecting to Zabbix. Refer to the respective section for RHEL/CentOS for instructions.

Starting Zabbix proxy process

To start a Zabbix proxy process and make it start at system boot:

```
# systemctl restart zabbix-proxy
# systemctl enable zabbix-proxy
```

Frontend configuration

A Zabbix proxy does not have a frontend; it communicates with Zabbix server only.
Java gateway installation

It is required to install Java gateway only if you want to monitor JMX applications. Java gateway is lightweight and does not require a database.

Once the required repository is added, you can install Zabbix Java gateway by running:

```
# apt install zabbix-java-gateway
```

Proceed to setup for more details on configuring and running Java gateway.

3 SUSE Linux Enterprise Server

Overview

Official Zabbix packages are available for:

<table>
<thead>
<tr>
<th>SUSE Linux Enterprise Server</th>
<th>Download</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

Verify CA encryption mode doesn’t work on SLES 12 (all minor OS versions) with MySQL due to older MySQL libraries.

Adding Zabbix repository

Install the repository configuration package. This package contains yum (software package manager) configuration files.

SLES 15:

```
# rpm -Uvh --nosignature https://repo.zabbix.com/zabbix/6.2/sles/15/x86_64/zabbix-release-6.2-1.sles15.noarch.rpm
# zypper --gpg-auto-import-keys refresh 'Zabbix Official Repository'
```

SLES 12:

```
# rpm -Uvh --nosignature https://repo.zabbix.com/zabbix/6.2/sles/12/x86_64/zabbix-release-6.2-1.sles12.noarch.rpm
# zypper --gpg-auto-import-keys refresh 'Zabbix Official Repository'
```

Please note, that Zabbix web service process, which is used for scheduled report generation, requires Google Chrome browser. The browser is not included into packages and has to be installed manually.

Server/frontend/agent installation

To install Zabbix server/frontend/agent with MySQL support:

```
# zypper install zabbix-server-mysql zabbix-web-mysql zabbix-apache-conf zabbix-agent
```

Substitute ‘apache’ in the command with ‘nginx’ if using the package for Nginx web server. See also: Nginx setup for Zabbix on SLES 12/15.

Substitute ‘zabbix-agent’ with ‘zabbix-agent2’ in these commands if using Zabbix agent 2 (only SLES 15 SP1+).

To install Zabbix proxy with MySQL support:

```
# zypper install zabbix-proxy-mysql zabbix-sql-scripts
```

Substitute ‘mysql’ in the commands with ‘pgsql’ to use PostgreSQL.

The package ‘zabbix-sql-scripts’ contains database schemas for all supported database management systems for both Zabbix server and Zabbix proxy and will be used for data import.

Creating database

For Zabbix server and proxy daemons a database is required. It is not needed to run Zabbix agent.

Separate databases are needed for Zabbix server and Zabbix proxy; they cannot use the same database. Therefore, if they are installed on the same host, their databases must be created with different names!

Create the database using the provided instructions for MySQL or PostgreSQL.

Importing data

Now import initial schema and data for the server with MySQL:

```
# zcat /usr/share/doc/packages/zabbix-sql-scripts/mysql/create.sql.gz | mysql -uzabbix -p zabbix
```

You will be prompted to enter your newly created database password.

With PostgreSQL:
# zcat /usr/share/doc/packages/zabbix-sql-scripts/postgresql/create.sql.gz | sudo -u zabbix psql zabbix

With TimescaleDB, in addition to the previous command, also run:

# zcat /usr/share/doc/packages/zabbix-sql-scripts/postgresql/timescaledb.sql.gz | sudo -u <username> psql zabbix

TimescaleDB is supported with Zabbix server only.

For proxy, import initial schema:

# zcat /usr/share/doc/packages/zabbix-sql-scripts/mysql/schema.sql.gz | mysql -uzabbix -p zabbix

For proxy with PostgreSQL:

# zcat /usr/share/doc/packages/zabbix-sql-scripts/postgresql/schema.sql.gz | sudo -u zabbix psql zabbix

Configure database for Zabbix server/proxy

Edit /etc/zabbix/zabbix_server.conf (and zabbix_proxy.conf) to use their respective databases. For example:

# vi /etc/zabbix/zabbix_server.conf
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=<password>

In DBPassword use Zabbix database password for MySQL; PostgreSQL user password for PostgreSQL.

Use DBHost= with PostgreSQL. You might want to keep the default setting DBHost=localhost (or an IP address), but this would make PostgreSQL use a network socket for connecting to Zabbix.

Zabbix frontend configuration

Depending on the web server used (Apache/Nginx) edit the corresponding configuration file for Zabbix frontend:

- For Apache the configuration file is located in /etc/apache2/conf.d/zabbix.conf. Some PHP settings are already configured. But it's necessary to uncomment the "date.timezone" setting and set the right timezone for you.

```
php_value max_execution_time 300
php_value memory_limit 128M
php_value post_max_size 16M
php_value upload_max_filesize 2M
php_value max_input_time 300
php_value max_input_vars 10000
php_value always_populate_raw_post_data -1
# php_value date.timezone Europe/Riga
```

- The zabbix-nginx-conf package installs a separate Nginx server for Zabbix frontend. Its configuration file is located in /etc/nginx/conf.d/zabbix.conf. For Zabbix frontend to work, it's necessary to uncomment and set listen and/or server_name directives.

```
# listen 80;
# server_name example.com;
```

- Zabbix uses its own dedicated php-fpm connection pool with Nginx:

Its configuration file is located in /etc/php7/fpm/php-fpm.d/zabbix.conf. Some PHP settings are already configured. But it's necessary to set the right date.timezone setting for you.

```
php_value [max_execution_time] = 300
php_value [memory_limit] = 128M
php_value [post_max_size] = 16M
php_value [upload_max_filesize] = 2M
php_value [max_input_time] = 300
php_value [max_input_vars] = 10000
; php_value [date.timezone] = Europe/Riga
```

Now you are ready to proceed with frontend installation steps which will allow you to access your newly installed Zabbix.

Note that a Zabbix proxy does not have a frontend; it communicates with Zabbix server only.

Starting Zabbix server/agent process

Start Zabbix server and agent processes and make it start at system boot.

With Apache web server:
# systemctl restart zabbix-server zabbix-agent apache2 php-fpm
# systemctl enable zabbix-server zabbix-agent apache2 php-fpm

Substitute `apache2` with `nginx` for Nginx web server.

Installing debuginfo packages

To enable debuginfo repository edit `/etc/zypp/repos.d/zabbix.repo` file. Change `enabled=0` to `enabled=1` for zabbix-debuginfo repository.

```
[zabbix-debuginfo]
name=Zabbix Official Repository debuginfo
type=rpm-md
baseurl=http://repo.zabbix.com/zabbix/6.2/sles/15/x86_64/debuginfo/
gpgcheck=1
gpgkey=http://repo.zabbix.com/zabbix/6.2/sles/15/x86_64/debuginfo/repodata/repomd.xml.key
enabled=0
update=1
```

This will allow you to install zabbix-<component>-debuginfo packages.

### 4 Windows agent installation from MSI

Overview

Zabbix Windows agent can be installed from Windows MSI installer packages (32-bit or 64-bit) available for [download](#). A 32-bit package cannot be installed on a 64-bit Windows.

All packages come with TLS support, however, configuring TLS is optional.

Both UI and command-line based installation is supported.

Installation steps

To install, double-click the downloaded MSI file.
Welcome to the Zabbix Agent (64-bit) Setup Wizard

The Setup Wizard will install Zabbix Agent (64-bit) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
Accept the license to proceed to the next step.

Specify the following parameters.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hostname</td>
<td>Specify host name.</td>
</tr>
<tr>
<td>Zabbix server IP/DNS</td>
<td>Specify IP/DNS of Zabbix server.</td>
</tr>
<tr>
<td>Agent listen port</td>
<td>Specify agent listen port (10050 by default).</td>
</tr>
<tr>
<td>Server or Proxy for active checks</td>
<td>Specify IP/DNS of Zabbix server/proxy for active agent checks.</td>
</tr>
<tr>
<td>Enable PSK</td>
<td>Mark the checkbox to enable TLS support via pre-shared keys.</td>
</tr>
<tr>
<td>Add agent location to the PATH</td>
<td>Add agent location to the PATH variable.</td>
</tr>
</tbody>
</table>

Enter pre-shared key identity and value. This step is only available if you checked Enable PSK in the previous step.
Select Zabbix components to install - Zabbix agent daemon, Zabbix sender, Zabbix get.
Zabbix components along with the configuration file will be installed in a Zabbix Agent folder in Program Files. `zabbix_agentd.exe` will be set up as Windows service with automatic startup.
Command-line based installation

Supported parameters

The following set of parameters is supported by created MSIs:

<table>
<thead>
<tr>
<th>Number</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LOGTYPE</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>LOGFILE</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SERVER</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>LISTENPORT</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SERVERACTIVE</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>HOSTNAME</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TIMEOUT</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>TLSCONNECT</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TLSACCEPT</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>TLPSKIDENTITY</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>TLPSKFILE</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>TLPSKVVALUE</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>TSCAFILE</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>TSCRLFILE</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>TLSSERVERCERTISSUER</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TLSSERVERCERTSUBJECT</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>TLSCERTFILE</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>TLSKEYFILE</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>LISTENIP</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>HOSTINTERFACE</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>HOSTMETADATA</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>HOSTMETADATAITEM</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>23</td>
<td>STATUSPORT</td>
<td>Zabbix agent 2 only.</td>
</tr>
<tr>
<td>24</td>
<td>ENABLEPERSISTENTBUFFER</td>
<td>Zabbix agent 2 only.</td>
</tr>
<tr>
<td>25</td>
<td>PERSISTENTBUFFERPERIOD</td>
<td>Zabbix agent 2 only.</td>
</tr>
<tr>
<td>26</td>
<td>PERSISTENTBUFFERFILE</td>
<td>Zabbix agent 2 only.</td>
</tr>
<tr>
<td>27</td>
<td>INSTALL_FOLDER</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>ENABLEPATH</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>SKIP</td>
<td>SKIP=fw - do not install firewall exception rule</td>
</tr>
<tr>
<td>30</td>
<td>INCLUDE</td>
<td>Sequence of includes separated by ;</td>
</tr>
<tr>
<td>31</td>
<td>ALLOWDENYKEY</td>
<td>Sequence of &quot;AllowKey&quot; and &quot;DenyKey&quot; parameters separated by ; Use ; to escape the delimiter.</td>
</tr>
</tbody>
</table>

To install you may run, for example:

```bash
SET INSTALLFOLDER=C:\Program Files\za

msiexec /i*v log.txt /i zabbix_agent-6.2.0-x86.msi /qn
LOGTYPE=file
LOGFILE="%INSTALLFOLDER%\za.log"
SERVER=192.168.6.76
LISTENPORT=12345
SERVERACTIVE::1
HOSTNAME=myHost
TLSCONNECT=psk
TLSCONNECT=psk
TLPSKIDENTITY=MyPSKID
TLPSKFILE="%INSTALLFOLDER%\mykey.psk"
TLSCAFILE="c:\temp\f.txt1"
TLSRCLFILE="c:\temp\f.txt2"
TLSSERVERCERTISSUER="My CA"
TLSSERVERCERTSUBJECT="My Cert"
TLSCERTFILE="c:\temp\f.txt5"
TLSKEYFILE="c:\temp\f.txt6"
ENABLEPATH=1
INSTALLFOLDER="%INSTALLFOLDER%"
SKIP=fw
ALLOWDENYKEY="DenyKey=vfs.file.contents[/etc/passwd]"
```

or

```bash
msiexec /i*v log.txt /i zabbix_agent-6.2.0-x86.msi /qn
SERVER=192.168.6.76
TLSCONNECT=psk
TLSCONNECT=psk
TLPSKIDENTITY=MyPSKID
TLPSKVALUE=1f87b595725ac58dd977beef14b97461a7c1045b9a1c9630650025c5473194952
```

5 Mac OS agent installation from PKG

Overview

Zabbix Mac OS agent can be installed from PKG installer packages available for download. Versions with or without encryption are available.

Installing agent

The agent can be installed using the graphical user interface or from the command line, for example:

```bash
sudo installer -pkg zabbix_agent-5.4.0-macos-amd64-openssl.pkg -target /
```

Make sure to use the correct Zabbix package version in the command. It must match the name of the downloaded package.

Running agent

The agent will start automatically after installation or restart.
You may edit the configuration file at /usr/local/etc/zabbix/zabbix_agentd.conf if necessary.

To start the agent manually, you may run:

```bash
sudo launchctl start com.zabbix.zabbix_agentd
```

To stop the agent manually:

```bash
sudo launchctl stop com.zabbix.zabbix_agentd
```

During upgrade, the existing configuration file is not overwritten. Instead a new zabbix_agentd.conf.NEW file is created to be used for reviewing and updating the existing configuration file, if necessary. Remember to restart the agent after manual changes to the configuration file.

Troubleshooting and removing agent

This section lists some useful commands that can be used for troubleshooting and removing Zabbix agent installation.

See if Zabbix agent is running:

```bash
ps aux | grep zabbix_agentd
```

See if Zabbix agent has been installed from packages:

```bash
$ pkgutil --pkgs | grep zabbix
com.zabbix.pkg.ZabbixAgent
```

See the files that were installed from the installer package (note that the initial / is not displayed in this view):

```bash
$ pkgutil --only-files --files com.zabbix.pkg.ZabbixAgent
Library/LaunchDaemons/com.zabbix.zabbix_agentd.plist
usr/local/bin/zabbix_get
usr/local/bin/zabbix_sender
usr/local/etc/zabbix/zabbix_agentd/userparameter_examples.conf.NEW
usr/local/etc/zabbix/zabbix_agentd/userparameter_mysql.conf.NEW
usr/local/etc/zabbix/zabbix_agentd.conf.NEW
usr/local/sbin/zabbix_agentd
```

Stop Zabbix agent if it was launched with launchctl:

```bash
sudo launchctl unload /Library/LaunchDaemons/com.zabbix.zabbix_agentd.plist
```

Remove files (including configuration and logs) that were installed with installer package:

```bash
sudo rm -f /Library/LaunchDaemons/com.zabbix.zabbix_agentd.plist
sudo rm -f /usr/local/sbin/zabbix_agentd
sudo rm -f /usr/local/bin/zabbix_get
sudo rm -f /usr/local/bin/zabbix_sender
sudo rm -rf /usr/local/etc/zabbix
sudo rm -rf /var/log/zabbix
```

Forget that Zabbix agent has been installed:

```bash
sudo pkgutil --forget com.zabbix.pkg.ZabbixAgent
```

## 5 Installation from containers

**Docker**  Zabbix provides Docker images for each Zabbix component as portable and self-sufficient containers to speed up deployment and update procedure.

Zabbix components come with MySQL and PostgreSQL database support, Apache2 and Nginx web server support. These images are separated into different images.

**Docker base images**

Zabbix components are provided on Ubuntu, Alpine Linux and CentOS base images:

<table>
<thead>
<tr>
<th>Image</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpine</td>
<td>3.12</td>
</tr>
<tr>
<td>ubuntu</td>
<td>20.04 (focal)</td>
</tr>
</tbody>
</table>
All images are configured to rebuild latest images if base images are updated.

**Docker file sources**

Everyone can follow Docker file changes using the Zabbix official repository on github.com. You can fork the project or make your own images based on official Docker files.

**Structure**

All Zabbix components are available in the following Docker repositories:

- **Zabbix agent** - zabbix/zabbix-agent
- **Zabbix server**
  - Zabbix server with MySQL database support - zabbix/zabbix-server-mysql
  - Zabbix server with PostgreSQL database support - zabbix/zabbix-server-pgsql
- **Zabbix web-interface**
  - Zabbix web-interface based on Apache2 web server with MySQL database support - zabbix/zabbix-web-apache-mysql
  - Zabbix web-interface based on Apache2 web server with PostgreSQL database support - zabbix/zabbix-web-apache-pgsql
  - Zabbix web-interface based on Nginx web server with MySQL database support - zabbix/zabbix-web-nginx-mysql
  - Zabbix web-interface based on Nginx web server with PostgreSQL database support - zabbix/zabbix-web-nginx-pgsql
- **Zabbix proxy**
  - Zabbix proxy with SQLite3 database support - zabbix/zabbix-proxy-sqlite3
  - Zabbix proxy with MySQL database support - zabbix/zabbix-proxy-mysql
- **Zabbix Java Gateway** - zabbix/zabbix-java-gateway

Additionally there is SNMP trap support. It is provided as additional repository (zabbix/zabbix-snmptraps) based on Ubuntu Trusty only. It could be linked with Zabbix server and Zabbix proxy.

**Versions**

Each repository of Zabbix components contains the following tags:

- `latest` - latest stable version of a Zabbix component based on Alpine Linux image
- `alpine-latest` - latest stable version of a Zabbix component based on Alpine Linux image
- `ubuntu-latest` - latest stable version of a Zabbix component based on Ubuntu image
- `alpine-6.2-latest` - latest minor version of a Zabbix 6.2 component based on Alpine Linux image
- `ubuntu-6.2-latest` - latest minor version of a Zabbix 6.2 component based on Ubuntu image
- `alpine-6.2.*` - different minor versions of a Zabbix 6.2 component based on Alpine Linux image, where * is the minor version of Zabbix component
- `ubuntu-6.2.*` - different minor versions of a Zabbix 6.2 component based on Ubuntu image, where * is the minor version of Zabbix component

**Usage**

**Environment variables**

All Zabbix component images provide environment variables to control configuration. These environment variables are listed in each component repository. These environment variables are options from Zabbix configuration files, but with different naming method. For example, ZBX_LOGSLOWQUERIES is equal to LogSlowQueries from Zabbix server and Zabbix proxy configuration files.

Some of configuration options are not allowed to change. For example, PIDFile and LogType.

Some of components have specific environment variables, which do not exist in official Zabbix configuration files:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Components</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB_SERVER_HOST</td>
<td>Server</td>
<td>This variable is IP or DNS name of MySQL or PostgreSQL server. By default, value is mysql-server or postgres-server for MySQL or PostgreSQL respectively.</td>
</tr>
<tr>
<td></td>
<td>Proxy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Web interface</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Server</td>
<td>This variable is port of MySQL or PostgreSQL server. By default, value is ‘3306’ or ‘5432’ respectively.</td>
</tr>
<tr>
<td></td>
<td>Proxy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Web interface</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Components</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>MYSQL_USER</td>
<td>Server</td>
<td>MySQL database user. By default, value is ‘zabbix’.</td>
</tr>
<tr>
<td></td>
<td>Proxy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Web interface</td>
<td></td>
</tr>
<tr>
<td>MYSQL_PASSWORD</td>
<td>Server</td>
<td>MySQL database password. By default, value is ‘zabbix’.</td>
</tr>
<tr>
<td></td>
<td>Proxy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Web interface</td>
<td></td>
</tr>
<tr>
<td>MYSQL_DATABASE</td>
<td>Server</td>
<td>Zabbix database name. By default, value is ‘zabbix’ for Zabbix server and ‘zabbix_proxy’ for Zabbix proxy.</td>
</tr>
<tr>
<td></td>
<td>Proxy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Web interface</td>
<td></td>
</tr>
<tr>
<td>POSTGRES_USER</td>
<td>Server</td>
<td>PostgreSQL database user. By default, value is ‘zabbix’.</td>
</tr>
<tr>
<td></td>
<td>Web interface</td>
<td></td>
</tr>
<tr>
<td>POSTGRES_PASSWORD</td>
<td>Server</td>
<td>PostgreSQL database password. By default, value is ‘zabbix’.</td>
</tr>
<tr>
<td></td>
<td>Web interface</td>
<td></td>
</tr>
<tr>
<td>POSTGRES_DB</td>
<td>Server</td>
<td>Zabbix database name. By default, value is ‘zabbix’ for Zabbix server and ‘zabbix_proxy’ for Zabbix proxy.</td>
</tr>
<tr>
<td></td>
<td>Web interface</td>
<td></td>
</tr>
<tr>
<td>PHP_TZ</td>
<td>Web interface</td>
<td>Timezone in PHP format. Full list of supported timezones are available on <a href="http://php.net">php.net</a>. By default, value is ‘Europe/Riga’.</td>
</tr>
<tr>
<td>ZBX_SERVER_NAME</td>
<td>Web interface</td>
<td>Visible Zabbix installation name in right top corner of the web interface. By default, value is ‘ZabbixDocker’.</td>
</tr>
<tr>
<td>ZBX_JAVAGATEWAY_ENABLE</td>
<td>Server</td>
<td>Enables communication with Zabbix Java gateway to collect Java related checks. By default, value is “false”</td>
</tr>
<tr>
<td></td>
<td>Proxy</td>
<td></td>
</tr>
<tr>
<td>ZBX_ENABLE_SNMP_TRAPS</td>
<td>Server</td>
<td>Enables SNMP trap feature. It requires <a href="https://github.com/zabbix/zabbix-snmptraps">zabbix-snmptraps</a> instance and shared volume /var/lib/zabbix/snmptraps to Zabbix server or Zabbix proxy.</td>
</tr>
<tr>
<td></td>
<td>Proxy</td>
<td></td>
</tr>
</tbody>
</table>

**Volumes**

The images allow to use some mount points. These mount points are different and depend on Zabbix component type:

<table>
<thead>
<tr>
<th>Volume</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Zabbix agent</strong></td>
<td></td>
</tr>
<tr>
<td>/etc/zabbix/zabbix_agentd.d</td>
<td>The volume allows to include *.conf files and extend Zabbix agent using the UserParameter feature</td>
</tr>
<tr>
<td>/var/lib/zabbix/modules</td>
<td>The volume allows to load additional modules and extend Zabbix agent using the LoadModule feature</td>
</tr>
<tr>
<td>/var/lib/zabbix/enc</td>
<td>The volume is used to store TLS-related files. These file names are specified using ZBX_TLSCAFILE, ZBX_TLSCRFILE, ZBX_TLSKEY_FILE and ZBX_TLSPSKFILE environment variables</td>
</tr>
<tr>
<td><strong>Zabbix server</strong></td>
<td></td>
</tr>
<tr>
<td>/usr/lib/zabbix/alertscripts</td>
<td>The volume is used for custom alert scripts. It is the AlertScriptsPath parameter in zabbix_server.conf</td>
</tr>
<tr>
<td>/usr/lib/zabbix/externalscripts</td>
<td>The volume is used by external checks. It is the ExternalScripts parameter in zabbix_server.conf</td>
</tr>
<tr>
<td>/var/lib/zabbix/modules</td>
<td>The volume allows to load additional modules and extend Zabbix server using the LoadModule feature</td>
</tr>
<tr>
<td>/var/lib/zabbix/enc</td>
<td>The volume is used to store TLS related files. These file names are specified using ZBX_TLSCAFILE, ZBX_TLSCRFILE, ZBX_TLSKEY_FILE and ZBX_TLSPSKFILE environment variables</td>
</tr>
<tr>
<td>/var/lib/zabbix/ssl/certs</td>
<td>The volume is used as location of SSL client certificate files for client authentication. It is the SSLCertLocation parameter in zabbix_server.conf</td>
</tr>
<tr>
<td>/var/lib/zabbix/ssl/keys</td>
<td>The volume is used as location of SSL private key files for client authentication. It is the SSLKeyLocation parameter in zabbix_server.conf</td>
</tr>
<tr>
<td>/var/lib/zabbix/ssl/ca</td>
<td>The volume is used as location of certificate authority (CA) files for SSL server certificate verification. It is the SSLCALocation parameter in zabbix_server.conf</td>
</tr>
<tr>
<td>Volume</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>/var/lib/zabbix/snmptraps</td>
<td>The volume is used as location of snmptraps.log file. It could be shared by zabbix-snmptraps container and inherited using the volumes_from Docker option while creating a new instance of Zabbix server. SNMP trap processing feature could be enabled by using shared volume and switching the ZBX_ENABLE_SNMP_TRAPS environment variable to 'true'</td>
</tr>
<tr>
<td>/var/lib/zabbix/mibs</td>
<td>The volume allows to add new MIB files. It does not support subdirectories, all MIBs must be placed in /var/lib/zabbix/mibs</td>
</tr>
<tr>
<td>/var/lib/zabbix/enc</td>
<td>The volume is used to store TLS related files. These file names are specified using ZBX_TLSCAFILE, ZBX_TLSCRIPFILE, ZBX_TLSKEY_FILE and ZBX_TLSPSKFILE environment variables</td>
</tr>
<tr>
<td>/var/lib/zabbix/ssl/certs</td>
<td>The volume is used as location of SSL client certificate files for client authentication. It is the SSLCertLocation parameter in zabbix_proxy.conf</td>
</tr>
<tr>
<td>/var/lib/zabbix/ssl/keys</td>
<td>The volume is used as location of SSL private key files for client authentication. It is the SSLKeyLocation parameter in zabbix_proxy.conf</td>
</tr>
<tr>
<td>/var/lib/zabbix/ssl_ca</td>
<td>The volume is used as location of certificate authority (CA) files for SSL server certificate verification. It is the SSLCALocation parameter in zabbix_proxy.conf</td>
</tr>
<tr>
<td>/var/lib/zabbix/snmptraps</td>
<td>The volume is used as location of snmptraps.log file. It could be shared by the zabbix-snmptraps container and inherited using the volumes_from Docker option while creating a new instance of Zabbix server. SNMP trap processing feature could be enabled by using shared volume and switching the ZBX_ENABLE_SNMP_TRAPS environment variable to 'true'</td>
</tr>
<tr>
<td>/var/lib/zabbix/mibs</td>
<td>The volume allows to add new MIB files. It does not support subdirectories, all MIBs must be placed in /var/lib/zabbix/mibs</td>
</tr>
</tbody>
</table>

Zabbix proxy

The volume is used by external checks. It is the ExternalScripts parameter in zabbix_proxy.conf

Zabbix modules

The volume allows to load additional modules and extend Zabbix server using the LoadModule feature

Zabbix web interface based on Apache2 web server

/etc/ssl/apache2

The volume allows to enable HTTPS for Zabbix web interface. The volume must contain the two ssl.crt and ssl.key files prepared for Apache2 SSL connections

Zabbix web interface based on Nginx web server

/etc/ssl/nginx

The volume allows to enable HTTPS for Zabbix web interface. The volume must contain the two ssl.crt, ssl.key files and dhparam.pem prepared for Nginx SSL connections

Zabbix snmptraps

The volume contains the snmptraps.log log file named with received SNMP traps

The volume allows to add new MIB files. It does not support subdirectories, all MIBs must be placed in /var/lib/zabbix/mibs

For additional information use Zabbix official repositories in Docker Hub.

Usage examples

**Example 1**

The example demonstrates how to run Zabbix server with MySQL database support, Zabbix web interface based on the Nginx web server and Zabbix Java gateway.

1. Create network dedicated for Zabbix component containers:

   ```bash
   # docker network create --subnet 172.20.0.0/16 --ip-range 172.20.240.0/20 zabbix-net
   ```

2. Start empty MySQL server instance

   ```bash
   # docker run --name mysql-server -t -p 3306:3306 --name mysql-server -d  
   -e MYSQL_DATABASE="zabbix" 
   -e MYSQL_USER="zabbix" 
   -e MYSQL_PASSWORD="zabbix_pwd" 
   -e MYSQL_ROOT_PASSWORD="root_pwd" 
   --network=zabbix-net 
   --restart unless-stopped 
   -d mysql:8.0 
   ```
3. Start Zabbix Java gateway instance

```bash
# docker run --name zabbix-java-gateway -t \
--network=zabbix-net \
--restart unless-stopped \
-d zabbix/zabbix-java-gateway:alpine-6.2-latest
```

4. Start Zabbix server instance and link the instance with created MySQL server instance

```bash
# docker run --name zabbix-server-mysql -t \
-e DB_SERVER_HOST="mysql-server" \
-e MYSQL_DATABASE="zabbix" \
-e MYSQL_USER="zabbix" \
-e MYSQL_PASSWORD="zabbix_pwd" \
-e MYSQL_ROOT_PASSWORD="root_pwd" \
-e ZBX_JAVAGATEWAY="zabbix-java-gateway" \
--network=zabbix-net \
-p 10051:10051 \
--restart unless-stopped \
-d zabbix/zabbix-server-mysql:alpine-6.2-latest
```

Zabbix server instance exposes 10051/TCP port (Zabbix trapper) to host machine.

5. Start Zabbix web interface and link the instance with created MySQL server and Zabbix server instances

```bash
# docker run --name zabbix-web-nginx-mysql -t \
-e DB_SERVER_HOST="mysql-server" \
-e MYSQL_DATABASE="zabbix" \
-e MYSQL_USER="zabbix" \
-e MYSQL_PASSWORD="zabbix_pwd" \
-e MYSQL_ROOT_PASSWORD="root_pwd" \
--network=zabbix-net \
-p 80:8080 \
--restart unless-stopped \
-d zabbix/zabbix-web-nginx-mysql:alpine-6.2-latest
```

Zabbix web interface instance exposes 80/TCP port (HTTP) to host machine.

**Example 2**

The example demonstrates how to run Zabbix server with PostgreSQL database support, Zabbix web interface based on the Nginx web server and SNMP trap feature.

1. Create network dedicated for Zabbix component containers:

```bash
# docker network create --subnet 172.20.0.0/16 --ip-range 172.20.240.0/20 zabbix-net
```

2. Start empty PostgreSQL server instance

```bash
# docker run --name postgres-server -t \
-e POSTGRES_USER="zabbix" \
-e POSTGRES_PASSWORD="zabbix_pwd" \
-e POSTGRES_DB="zabbix" \
--network=zabbix-net \
--restart unless-stopped \
-d postgres:latest
```

3. Start Zabbix snmptraps instance

```bash
# docker run --name zabbix-snmptraps -t \
-v /zbx_instance/snmptraps:/var/lib/zabbix/snmptraps:rw \
-v /var/lib/zabbix/mibs:/usr/share/snmp/mibs:ro \
--network=zabbix-net \
-p 162:1162/udp \
--restart unless-stopped \
-d zabbix/zabbix-snmptraps:alpine-6.2-latest
```
Zabbix snmptrap instance exposes the 162/UDP port (SNMP traps) to host machine.

4. Start Zabbix server instance and link the instance with created PostgreSQL server instance

```bash
# docker run --name zabbix-server-pgsql -t -e DB_SERVER_HOST="postgres-server" -e POSTGRES_USER="zabbix" -e POSTGRES_PASSWORD="zabbix_pwd" -e POSTGRES_DB="zabbix" -e ZBX_ENABLE_SNMP_TRAPS="true" --network=zabbix-net --p 10051:10051 --volumes-from zabbix-snmptraps --restart unless-stopped --d zabbix/zabbix-server-pgsql:alpine-6.2-latest
```

Zabbix server instance exposes the 10051/TCP port (Zabbix trapper) to host machine.

5. Start Zabbix web interface and link the instance with created PostgreSQL server and Zabbix server instances

```bash
```

Zabbix web interface instance exposes the 443/TCP port (HTTPS) to host machine.

Directory `/etc/ssl/nginx` must contain certificate with required name.

**Example 3**

The example demonstrates how to run Zabbix server with MySQL database support, Zabbix web interface based on the Nginx web server and Zabbix Java gateway using podman on Red Hat 8.

1. Create new pod with name zabbix and exposed ports (web-interface, Zabbix server trapper):

   ```bash
   podman pod create --name zabbix -p 80:8080 -p 10051:10051
   `` `

2. (optional) Start Zabbix agent container in zabbix pod location:

   ```bash
   podman run --name zabbix-agent -e ZBX_SERVER_HOST="127.0.0.1,localhost" --restart=always --pod=zabbix --d registry.connect.redhat.com/zabbix/zabbix-agent-50:latest
   `` `

3. Create `.mysql/` directory on host and start Oracle MySQL server 8.0:

   ```bash
   podman run --name mysql-server -t -e MYSQL_DATABASE="zabbix" -e MYSQL_USER="zabbix" -e MYSQL_PASSWORD="zabbix_pwd" -e MYSQL_ROOT_PASSWORD="root_pwd" -v .:/var/lib/mysql:/Z --restart=always --pod=zabbix --d mysql:8.0 --character-set-server=utf8 --collation-server=utf8_bin --default-authentication-plugin=mysql_native_password
   `` `

3. Start Zabbix server container:

   ```bash
   podman run --name zabbix-server-mysql -t --DB_SERVER_HOST="127.0.0.1"
   ```
4. Start Zabbix Java Gateway container:

   podman run --name zabbix-java-gateway -t \
     --restart=always \
     --pod=zabbix \
     -d registry.connect.redhat.com/zabbix/zabbix-java-gateway-50

5. Start Zabbix web-interface container:

   podman run --name zabbix-web-mysql -t \
     -e ZBX_SERVER_HOST="127.0.0.1" \
     -e DB_SERVER_HOST="127.0.0.1" \
     -e MYSQL_DATABASE="zabbix" \
     -e MYSQL_USER="zabbix" \
     -e MYSQL_PASSWORD="zabbix_pwd" \
     -e MYSQL_ROOT_PASSWORD="root_pwd" \
     --restart=always \
     --pod=zabbix \
     -d registry.connect.redhat.com/zabbix/zabbix-web-mysql-50

Pod zabbix exposes 80/TCP port (HTTP) to host machine from 8080/TCP of zabbix-web-mysql container.

**Docker Compose** Zabbix provides compose files also for defining and running multi-container Zabbix components in Docker. These compose files are available in Zabbix docker official repository on github.com: https://github.com/zabbix/zabbix-docker. These compose files are added as examples, they are overloaded. For example, they contain proxies with MySQL and SQLite3 support.

There are a few different versions of compose files:

<table>
<thead>
<tr>
<th>File name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>docker-compose_v3_alpine_mysql_latest.yaml</td>
<td>The compose file runs the latest version of Zabbix 6.2 components on Alpine Linux with MySQL database support.</td>
</tr>
<tr>
<td>docker-compose_v3_alpine_mysql_local.yaml</td>
<td>The compose file locally builds the latest version of Zabbix 6.2 and runs Zabbix components on Alpine Linux with MySQL database support.</td>
</tr>
<tr>
<td>docker-compose_v3_alpine_pgsql_latest.yaml</td>
<td>The compose file runs the latest version of Zabbix 6.2 components on Alpine Linux with PostgreSQL database support.</td>
</tr>
<tr>
<td>docker-compose_v3_alpine_pgsql_local.yaml</td>
<td>The compose file locally builds the latest version of Zabbix 6.2 and runs Zabbix components on Alpine Linux with PostgreSQL database support.</td>
</tr>
<tr>
<td>docker-compose_v3_centos_mysql_latest.yaml</td>
<td>The compose file runs the latest version of Zabbix 6.2 components on CentOS 8 with MySQL database support.</td>
</tr>
<tr>
<td>docker-compose_v3_centos_mysql_local.yaml</td>
<td>The compose file locally builds the latest version of Zabbix 6.2 and runs Zabbix components on CentOS 8 with MySQL database support.</td>
</tr>
<tr>
<td>docker-compose_v3_centos_pgsql_latest.yaml</td>
<td>The compose file runs the latest version of Zabbix 6.2 components on CentOS 8 with PostgreSQL database support.</td>
</tr>
<tr>
<td>docker-compose_v3_centos_pgsql_local.yaml</td>
<td>The compose file locally builds the latest version of Zabbix 6.2 and runs Zabbix components on CentOS 8 with PostgreSQL database support.</td>
</tr>
<tr>
<td>docker-compose_v3_ubuntu_mysql_latest.yaml</td>
<td>The compose file runs the latest version of Zabbix 6.2 components on Ubuntu 20.04 with MySQL database support.</td>
</tr>
<tr>
<td>docker-compose_v3_ubuntu_mysql_local.yaml</td>
<td>The compose file locally builds the latest version of Zabbix 6.2 and runs Zabbix components on Ubuntu 20.04 with MySQL database support.</td>
</tr>
<tr>
<td>docker-compose_v3_ubuntu_pgsql_latest.yaml</td>
<td>The compose file runs the latest version of Zabbix 6.2 components on Ubuntu 20.04 with PostgreSQL database support.</td>
</tr>
<tr>
<td>docker-compose_v3_ubuntu_pgsql_local.yaml</td>
<td>The compose file locally builds the latest version of Zabbix 6.2 and runs Zabbix components on Ubuntu 20.04 with PostgreSQL database support.</td>
</tr>
</tbody>
</table>

Available Docker compose files support version 3 of Docker Compose.
Storage

Compose files are configured to support local storage on a host machine. Docker Compose will create a `zbx_env` directory in the folder with the compose file when you run Zabbix components using the compose file. The directory will contain the same structure as described above in the Volumes section and directory for database storage.

There are also volumes in read-only mode for `/etc/localtime` and `/etc/timezone` files.

Environment files

In the same directory with compose files on github.com you can find files with default environment variables for each component in compose file. These environment files are named like `.env_<type of component>`.

Examples

**Example 1**

```bash
# git checkout 6.2
# docker-compose -f ./docker-compose_v3_alpine_mysql_latest.yaml up -d
```

The command will download latest Zabbix 6.2 images for each Zabbix component and run them in detach mode.

Do not forget to download `.env_<type of component>` files from github.com official Zabbix repository with compose files.

**Example 2**

```bash
# git checkout 6.2
# docker-compose -f ./docker-compose_v3_ubuntu_mysql_local.yaml up -d
```

The command will download base image Ubuntu 20.04 (focal), then build Zabbix 6.2 components locally and run them in detach mode.

6 Web interface installation

This section provides step-by-step instructions for installing Zabbix web interface. Zabbix frontend is written in PHP, so to run it a PHP supported webserver is needed.

Welcome screen

Open Zabbix frontend URL in the browser. If you have installed Zabbix from packages, the URL is:

- for Apache: `http://<server_ip_or_name>/zabbix`
- for Nginx: `http://<server_ip_or_name>`

You should see the first screen of the frontend installation wizard.

Use the Default language drop-down menu to change system default language and continue the installation process in the selected language (optional). For more information, see Installation of additional frontend languages.
Check of pre-requisites

Make sure that all software prerequisites are met.

### Pre-requisite Table

<table>
<thead>
<tr>
<th>Pre-requisite</th>
<th>Minimum value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHP version</td>
<td>7.4.0</td>
<td></td>
</tr>
</tbody>
</table>
| PHP memory_limit option              | 128MB         | In php.ini:  
memory_limit = 128M |
| PHP post_max_size option             | 16MB          | In php.ini:  
post_max_size = 16M |
| PHP upload_max_filesize option       | 2MB           | In php.ini:  
upload_max_filesize = 2M |
| PHP max_execution_time option        | 300 seconds (values 0 and -1 are allowed) | In php.ini:  
max_execution_time = 300 |
| PHP max_input_time option            | 300 seconds (values 0 and -1 are allowed) | In php.ini:  
max_input_time = 300 |
| PHP session.auto_start option        | must be disabled | In php.ini:  
session.auto_start = 0 |
| Database support                     | One of: MySQL, Oracle, PostgreSQL. | One of the following modules must be installed:  
mysql, oci8, pgsql, php-bcmath, php-mbstring |
| bcmath                               |               |             |
| mbstring                             |               |             |
| PHP mbstring.func_overload option    | must be disabled | In php.ini:  
mbstring.func_overload = 0 |
| sockets                              |               |             |
| gd                                   | 2.0.28        | php-net-socket. Required for user script support. |
| xml                                    | 2.6.15        | php-gd. PHP GD extension must support PNG images (--with-png-dir), JPEG (--with-jpeg-dir) images and FreeType 2 (--with-freetype-dir).  
php-xl, php-xmllreader, php-xmllreader, php-ctype, php-session, php-gettext |
| libxml                                |               |             |
| xmlreader                             |               |             |
| ctype                                 |               |             |
| session                               |               |             |
| gettext                               |               |             |

Since Zabbix 2.2.1, the PHP gettext extension is not a mandatory requirement for installing Zabbix. If gettext is not installed, the frontend will work as usual, however, the translations will not be available.
Optional pre-requisites may also be present in the list. A failed optional prerequisite is displayed in orange and has a Warning status. With a failed optional pre-requisite, the setup may continue.

If there is a need to change the Apache user or user group, permissions to the session folder must be verified. Otherwise Zabbix setup may be unable to continue.

Configure DB connection

Enter details for connecting to the database. Zabbix database must already be created.

If the Database TLS encryption option is checked, then additional fields for configuring the TLS connection to the database appear in the form (MySQL or PostgreSQL only).

If Store credentials in is set to HashiCorp Vault or CyberArk Vault, additional parameters will become available:

- for HashiCorp Vault: Vault API endpoint, secret path and authentication token;
- for CyberArk Vault: Vault API endpoint, secret query string and certificates. Upon marking Vault certificates checkbox, two new fields for specifying paths to SSL certificate file and SSL key file will appear.

Settings

Entering a name for Zabbix server is optional, however, if submitted, it will be displayed in the menu bar and page titles.

Set the default time zone and theme for the frontend.
Pre-installation summary

Review a summary of settings.

Install

If installing Zabbix from sources, download the configuration file and place it under conf/ in the webserver HTML documents subdirectory where you copied Zabbix PHP files to.
Providing the webserver user has write access to config/ directory the configuration file would be saved automatically and it would be possible to proceed to the next step right away.

Finish the installation.
Zabbix frontend is ready! The default username is Admin, password zabbix.

Proceed to getting started with Zabbix.

7 Upgrade procedure

Overview

This section provides upgrade information for Zabbix 6.2:

- using packages:
  - for Red Hat Enterprise Linux/CentOS
  - for Debian/Ubuntu
- using sources

Direct upgrade to Zabbix 6.2.x is possible from Zabbix 6.0.x, 5.4.x, 5.2.x, 5.0.x, 4.4.x, 4.2.x, 4.0.x, 3.4.x, 3.2.x, 3.0.x, 2.4.x, 2.2.x and 2.0.x. For upgrading from earlier versions consult Zabbix documentation for 2.0 and earlier.

Upgrade from packages

Overview

This section provides the steps required for a successful upgrade using official RPM and DEB packages provided by Zabbix for:

- Red Hat Enterprise Linux/CentOS
- Debian/Ubuntu

Zabbix packages from OS repositories

Often, OS distributions (in particular, Debian-based distributions) provide their own Zabbix packages. Note, that these packages are not supported by Zabbix, they are typically out of date and lack the latest features and bug fixes. Only the packages from repo.zabbix.com are officially supported.

If you are upgrading from packages provided by OS distributions (or had them installed at some point), follow this procedure to switch to official Zabbix packages:

1. Always uninstall the old packages first.
2. Check for residual files that may have been left after deinstallation.
3. Install official packages following installation instructions provided by Zabbix.
Never do a direct update, as this may result in a broken installation.

## 1 Red Hat Enterprise Linux/CentOS

### Overview

This section provides the steps required for a successful upgrade from Zabbix 6.0.x to Zabbix 6.2.x using official Zabbix packages for Red Hat Enterprise Linux/CentOS.

While upgrading Zabbix agents is not mandatory (but recommended), Zabbix server and proxies must be of the same major version. Therefore, in a server-proxy setup, Zabbix server and all proxies have to be stopped and upgraded. Keeping proxies running during server upgrade no longer will bring any benefit as during proxy upgrade their old data will be discarded and no new data will be gathered until proxy configuration is synced with server.

Note that with SQLite database on proxies, history data from proxies before the upgrade will be lost, because SQLite database upgrade is not supported and the SQLite database file has to be manually removed. When proxy is started for the first time and the SQLite database file is missing, proxy creates it automatically.

Depending on database size the database upgrade to version 6.2 may take a long time.

Before the upgrade make sure to read the relevant upgrade notes!

The following upgrade notes are available:

<table>
<thead>
<tr>
<th>Upgrade from</th>
<th>Read full upgrade notes</th>
<th>Most important changes between versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.0.x LTS</td>
<td>For: Zabbix 6.2</td>
<td>Minimum required PHP version upped from 7.2.5 to 7.4.0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deterministic triggers need to be created during the upgrade. If binary logging is enabled for MySQL/MariaDB, this requires superuser privileges or setting the variable/configuration parameter <code>log_bin_trust_function_creators = 1</code>. See Database creation scripts for instructions how to set the variable.</td>
</tr>
<tr>
<td>5.4.x</td>
<td>For: Zabbix 6.0, Zabbix 6.2</td>
<td>Minimum required database versions upped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Server/proxy will not start if outdated database.</td>
</tr>
<tr>
<td>5.2.x</td>
<td>For: Zabbix 5.4, Zabbix 6.0, Zabbix 6.2</td>
<td>Minimum required database versions upped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Aggregate items removed as a separate type.</td>
</tr>
<tr>
<td>5.0.x LTS</td>
<td>For: Zabbix 5.2, Zabbix 5.4, Zabbix 6.0, Zabbix 6.2</td>
<td>Minimum required PHP version upped from 7.2.0 to 7.2.5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Password hashing algorithm changed from MD5 to bcrypt.</td>
</tr>
<tr>
<td>4.4.x</td>
<td>For: Zabbix 5.0, Zabbix 5.2, Zabbix 5.4, Zabbix 6.0, Zabbix 6.2</td>
<td>Support of IBM DB2 dropped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Minimum required PHP version upped from 5.4.0 to 7.2.0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Minimum required database versions upped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed Zabbix PHP file directory.</td>
</tr>
<tr>
<td>4.2.x</td>
<td>For: Zabbix 4.4, Zabbix 5.0, Zabbix 5.2, Zabbix 5.4, Zabbix 6.0, Zabbix 6.2</td>
<td>Jabber, Ez Texting media types removed.</td>
</tr>
<tr>
<td>4.0.x LTS</td>
<td>For: Zabbix 4.2, Zabbix 4.4, Zabbix 5.0, Zabbix 5.2, Zabbix 5.4, Zabbix 6.0, Zabbix 6.2</td>
<td>Older proxies no longer can report data to an upgraded server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Newer agents no longer will be able to work with an older Zabbix server.</td>
</tr>
<tr>
<td>Upgrade from</td>
<td>Read full upgrade notes</td>
<td>Most important changes between versions</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>3.4.x</td>
<td>For:</td>
<td>'libpthread' and 'zlib' libraries now mandatory.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td>Support for plain text protocol dropped and header is mandatory.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td>Pre-1.4 version Zabbix agents are no longer supported.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td>The Server parameter in passive proxy configuration now mandatory.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td></td>
</tr>
<tr>
<td>3.2.x</td>
<td>For:</td>
<td>SQLite support as backend database dropped for Zabbix server/frontend.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.4</td>
<td>Perl Compatible Regular Expressions (PCRE) supported instead of POSIX extended.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td>'libpcre' and 'libevent' libraries mandatory for Zabbix server.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td>Exit code checks added for user parameters, remote commands and system.run[] items without the ‘nowait’ flag as well as Zabbix server executed scripts.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td>Zabbix Java gateway has to be upgraded to support new functionality.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td></td>
</tr>
<tr>
<td>3.0.x LTS</td>
<td>For:</td>
<td>Database upgrade may be slow, depending on the history table size.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td></td>
</tr>
<tr>
<td>2.4.x</td>
<td>For:</td>
<td>Minimum required PHP version upped from 5.3.0 to 5.4.0</td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.0</td>
<td>LogFile agent parameter must be specified</td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td></td>
</tr>
<tr>
<td>2.2.x LTS</td>
<td>For:</td>
<td>Node-based distributed monitoring removed</td>
</tr>
<tr>
<td></td>
<td>Zabbix 2.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td></td>
</tr>
</tbody>
</table>
You may also want to check the requirements for 6.2.

It may be handy to run two parallel SSH sessions during the upgrade, executing the upgrade steps in one and monitoring the server/proxy logs in another. For example, run `tail -f zabbix_server.log` or `tail -f zabbix_proxy.log` in the second SSH session showing you the latest log file entries and possible errors in real time. This can be critical for production instances.

Upgrade procedure

1 Stop Zabbix processes

Stop Zabbix server to make sure that no new data is inserted into database.

```
# systemctl stop zabbix-server
```

If upgrading the proxy, stop proxy too.

```
# systemctl stop zabbix-proxy
```

It is no longer possible to start the upgraded server and have older, yet unupgraded proxies report data to a newer server. This approach, which was never recommended nor supported by Zabbix, now is officially disabled, as the server will ignore data from unupgraded proxies.

2 Back up the existing Zabbix database

This is a very important step. Make sure that you have a backup of your database. It will help if the upgrade procedure fails (lack of disk space, power off, any unexpected problem).

3 Back up configuration files, PHP files and Zabbix binaries

Make a backup copy of Zabbix binaries, configuration files and the PHP file directory.

Configuration files:

```
# mkdir /opt/zabbix-backup/
# cp /etc/zabbix/zabbix_server.conf /opt/zabbix-backup/
# cp /etc/httpd/conf.d/zabbix.conf /opt/zabbix-backup/
```

PHP files and Zabbix binaries:

```
# cp -R /usr/share/zabbix/ /opt/zabbix-backup/
# cp -R /usr/share/doc/zabbix-* /opt/zabbix-backup/
```

4 Update repository configuration package

To proceed with the upgrade your current repository package has to be updated.

```
# rpm -Uvh https://repo.zabbix.com/zabbix/6.2/rhel/8/x86_64/zabbix-release-6.2-1.el8.noarch.rpm
```

5 Upgrade Zabbix components

To upgrade Zabbix components you may run something like:

```
# yum upgrade zabbix-server-mysql zabbix-web-mysql zabbix-agent
```
If using PostgreSQL, substitute mysql with pgsql in the command. If upgrading the proxy, substitute server with proxy in the command. If upgrading the agent 2, substitute zabbix-agent with zabbix-agent2 in the command.

To upgrade the web frontend with Apache on RHEL 8 correctly, also run:

```
# yum install zabbix-apache-conf
```

and make the necessary changes to this file.

To upgrade the web frontend on RHEL 7 follow distribution-specific instructions (extra steps are required to install PHP 7.2 or newer).

6 Review component configuration parameters

See the upgrade notes for details on mandatory changes.

7 Start Zabbix processes

Start the updated Zabbix components.

```
# systemctl start zabbix-server
# systemctl start zabbix-proxy
# systemctl start zabbix-agent
# systemctl start zabbix-agent2
```

8 Clear web browser cookies and cache

After the upgrade you may need to clear web browser cookies and web browser cache for the Zabbix web interface to work properly.

Upgrade between minor versions

It is possible to upgrade between minor versions of 6.2.x (for example, from 6.2.1 to 6.2.3). Upgrading between minor versions is easy.

To execute Zabbix minor version upgrade it is required to run:

```
$ sudo yum upgrade 'zabbix-*'
```

To execute Zabbix server minor version upgrade run:

```
$ sudo yum upgrade 'zabbix-server-*'
```

To execute Zabbix agent minor version upgrade run:

```
$ sudo yum upgrade 'zabbix-agent-*'
```

or, for Zabbix agent 2:

```
$ sudo yum upgrade 'zabbix-agent2-*'
```

Note that you may also use ‘update’ instead of ‘upgrade’ in these commands. While ‘upgrade’ will delete obsolete packages, ‘update’ will preserve them.

2 Debian/Ubuntu

Overview

This section provides the steps required for a successful upgrade from Zabbix 6.0.x to Zabbix 6.2.x using official Zabbix packages for Debian/Ubuntu.

While upgrading Zabbix agents is not mandatory (but recommended), Zabbix server and proxies must be of the same major version. Therefore, in a server-proxy setup, Zabbix server and all proxies have to be stopped and upgraded. Keeping proxies running during server upgrade no longer will bring any benefit as during proxy upgrade their old data will be discarded and no new data will be gathered until proxy configuration is synced with server.

Note that with SQLite database on proxies, history data from proxies before the upgrade will be lost, because SQLite database upgrade is not supported and the SQLite database file has to be manually removed. When proxy is started for the first time and the SQLite database file is missing, proxy creates it automatically.

Depending on database size the database upgrade to version 6.2 may take a long time.

Before the upgrade make sure to read the relevant upgrade notes!

The following upgrade notes are available:
<table>
<thead>
<tr>
<th>Upgrade from</th>
<th>Read full upgrade notes</th>
<th>Most important changes between versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.0.x LTS</td>
<td>For: Zabbix 6.2</td>
<td>Minimum required PHP version upped from 7.2.5 to 7.4.0. Deterministic triggers need to be created during the upgrade. If binary logging is enabled for MySQL/MariaDB, this requires superuser privileges or setting the variable/configuration parameter <code>log_bin_trust_function_creators = 1</code>. See <a href="#">Database creation scripts</a> for instructions how to set the variable.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td>Server/proxy will not start if outdated database.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td>Audit log records lost because of database structure change.</td>
</tr>
<tr>
<td>5.4.x</td>
<td>For: Zabbix 6.0</td>
<td>Minimum required database versions upped.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td>Aggregate items removed as a separate type.</td>
</tr>
<tr>
<td>5.2.x</td>
<td>For: Zabbix 5.4</td>
<td>Minimum required database versions upped.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td>Aggregate items removed as a separate type.</td>
</tr>
<tr>
<td>5.0.x LTS</td>
<td>For: Zabbix 5.2</td>
<td>Minimum required PHP version upped from 7.2.0 to 7.2.5. Password hashing algorithm changed from MD5 to bcrypt.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td>Aggregate items removed as a separate type.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td>Aggregate items removed as a separate type.</td>
</tr>
<tr>
<td>4.4.x</td>
<td>For: Zabbix 5.0</td>
<td>Support of IBM DB2 dropped.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td>Minimum required PHP version upped from 5.4.0 to 7.2.0.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td>Minimum required database versions upped.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td>Changed Zabbix PHP file directory.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td>Changed Zabbix PHP file directory.</td>
</tr>
<tr>
<td>4.2.x</td>
<td>For: Zabbix 4.4</td>
<td>Jabber, Ez Texting media types removed.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td>Aggregate items removed as a separate type.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td>Aggregate items removed as a separate type.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td>Aggregate items removed as a separate type.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td>Aggregate items removed as a separate type.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td>Aggregate items removed as a separate type.</td>
</tr>
<tr>
<td>4.0.x LTS</td>
<td>For: Zabbix 4.2</td>
<td>Older proxes no longer can report data to an upgraded server.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td>Newer agents no longer will be able to work with an older Zabbix server.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td>Newer agents no longer will be able to work with an older Zabbix server.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td>Newer agents no longer will be able to work with an older Zabbix server.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td>Newer agents no longer will be able to work with an older Zabbix server.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td>Newer agents no longer will be able to work with an older Zabbix server.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td>Newer agents no longer will be able to work with an older Zabbix server.</td>
</tr>
<tr>
<td>3.4.x</td>
<td>For: Zabbix 4.0</td>
<td>‘libpthread’ and ‘zlib’ libraries now mandatory.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td>Support for plain text protocol dropped and header is mandatory.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td>Pre-1.4 version Zabbix agents are no longer supported.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td>The Server parameter in passive proxy configuration now mandatory.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td>The Server parameter in passive proxy configuration now mandatory.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td>The Server parameter in passive proxy configuration now mandatory.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td>The Server parameter in passive proxy configuration now mandatory.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td>The Server parameter in passive proxy configuration now mandatory.</td>
</tr>
<tr>
<td>3.2.x</td>
<td>For: Zabbix 3.4</td>
<td>SQLite support as backend database dropped for Zabbix server/frontend.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td>Perl Compatible Regular Expressions (PCRE) supported instead of POSIX extended.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td>‘libpcre’ and ‘libevent’ libraries mandatory for Zabbix server.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td>Exit code checks added for user parameters, remote commands and system.run[] items without the ‘nowait’ flag as well as Zabbix server executed scripts.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td>Zabbix Java gateway has to be upgraded to support new functionality.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td>Zabbix Java gateway has to be upgraded to support new functionality.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td>Zabbix Java gateway has to be upgraded to support new functionality.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td>Zabbix Java gateway has to be upgraded to support new functionality.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td>Zabbix Java gateway has to be upgraded to support new functionality.</td>
</tr>
<tr>
<td>Upgrade from</td>
<td>Read full upgrade notes</td>
<td>Most important changes between versions</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>3.0.x LTS</td>
<td>For:</td>
<td>Database upgrade may be slow, depending on the history table size.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td></td>
</tr>
<tr>
<td>2.4.x</td>
<td>For:</td>
<td>Minimum required PHP version upped from 5.3.0 to 5.4.0</td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.0</td>
<td>LogFile agent parameter must be specified</td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td></td>
</tr>
<tr>
<td>2.2.x LTS</td>
<td>For:</td>
<td>Node-based distributed monitoring removed</td>
</tr>
<tr>
<td></td>
<td>Zabbix 2.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td></td>
</tr>
<tr>
<td>2.0.x</td>
<td>For:</td>
<td>Minimum required PHP version upped from 5.1.6 to 5.3.0.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 2.2</td>
<td>Case-sensitive MySQL database required for proper server work; character set utf8 and utf8_bin collation is required for Zabbix server to work properly with MySQL database. See database creation scripts. ‘mysqli’ PHP extension required instead of ‘mysql’</td>
</tr>
<tr>
<td></td>
<td>Zabbix 2.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td></td>
</tr>
</tbody>
</table>

You may also want to check the requirements for 6.2.

It may be handy to run two parallel SSH sessions during the upgrade, executing the upgrade steps in one and monitoring the server/proxy logs in another. For example, run `tail -f zabbix_server.log` or `tail -f zabbix_proxy.log` in the second SSH session showing you the latest log file entries and possible errors in real time. This can be critical for production instances.

Upgrade procedure

1 Stop Zabbix processes

Stop Zabbix server to make sure that no new data is inserted into database.
# service zabbix-server stop
If upgrading Zabbix proxy, stop proxy too.
# service zabbix-proxy stop

2 Back up the existing Zabbix database
This is a very important step. Make sure that you have a backup of your database. It will help if the upgrade procedure fails (lack of disk space, power off, any unexpected problem).

3 Back up configuration files, PHP files and Zabbix binaries
Make a backup copy of Zabbix binaries, configuration files and the PHP file directory.
Configuration files:
# mkdir /opt/zabbix-backup/
# cp /etc/zabbix/zabbix_server.conf /opt/zabbix-backup/
# cp /etc/apache2/conf-enabled/zabbix.conf /opt/zabbix-backup/

PHP files and Zabbix binaries:
# cp -R /usr/share/zabbix/ /opt/zabbix-backup/
# cp -R /usr/share/doc/zabbix-* /opt/zabbix-backup/

4 Update repository configuration package
To proceed with the update your current repository package has to be uninstalled.
# rm -Rf /etc/apt/sources.list.d/zabbix.list
Then install the new repository configuration package.
On Debian 11 run:
# dpkg -i zabbix-release_6.2-1+debian11_all.deb
On Debian 10 run:
# dpkg -i zabbix-release_6.2-1+debian10_all.deb
On Debian 9 run:
# dpkg -i zabbix-release_6.2-1+debian9_all.deb
On Ubuntu 20.04 run:
# dpkg -i zabbix-release_6.2-1+ubuntu20.04_all.deb
On Ubuntu 18.04 run:
# dpkg -i zabbix-release_6.2-1+ubuntu18.04_all.deb
On Ubuntu 16.04 run:
# dpkg -i zabbix-release_6.2-1+ubuntu16.04_all.deb
On Ubuntu 14.04 run:
# dpkg -i zabbix-release_6.2-1+ubuntu14.04_all.deb
Update the repository information.
# apt-get update
5 Upgrade Zabbix components
To upgrade Zabbix components you may run something like:
# apt-get install --only-upgrade zabbix-server-mysql zabbix-frontend-php zabbix-agent
If using PostgreSQL, substitute `mysql` with `pgsql` in the command. If upgrading the proxy, substitute `server` with `proxy` in the command. If upgrading the Zabbix agent 2, substitute `zabbix-agent` with `zabbix-agent2` in the command.

Then, to upgrade the web frontend with Apache correctly, also run:

```
# apt-get install zabbix-apache-conf
```

Distributions prior to Debian 10 (buster) / Ubuntu 18.04 (bionic) / Raspbian 10 (buster) do not provide PHP 7.2 or newer, which is required for Zabbix frontend 5.0. See information about installing Zabbix frontend on older distributions.

6 Review component configuration parameters

See the upgrade notes for details on mandatory changes (if any).

For new optional parameters, see the What's new section.

7 Start Zabbix processes

Start the updated Zabbix components.

```
# service zabbix-server start
# service zabbix-proxy start
# service zabbix-agent start
# service zabbix-agent2 start
```

8 Clear web browser cookies and cache

After the upgrade you may need to clear web browser cookies and web browser cache for the Zabbix web interface to work properly.

Upgrade between minor versions

It is possible to upgrade minor versions of 6.2.x (for example, from 6.2.1 to 6.2.3). It is easy.

To upgrade Zabbix minor version please run:

```
$ sudo apt install --only-upgrade 'zabbix.*'
```

To upgrade Zabbix server minor version please run:

```
$ sudo apt install --only-upgrade 'zabbix-server.*'
```

To upgrade Zabbix agent minor version please run:

```
$ sudo apt install --only-upgrade 'zabbix-agent.*'
```

or, for Zabbix agent 2:

```
$ sudo apt install --only-upgrade 'zabbix-agent2.*'
```

Upgrade from sources

Overview

This section provides the steps required for a successful upgrade from Zabbix 6.0.x to Zabbix 6.2.x using official Zabbix sources.

While upgrading Zabbix agents is not mandatory (but recommended), Zabbix server and proxies must be of the same major version. Therefore, in a server-proxy setup, Zabbix server and all proxies have to be stopped and upgraded. Keeping proxies running no longer will bring any benefit as during proxy upgrade their old data will be discarded and no new data will be gathered until proxy configuration is synced with server.

It is no longer possible to start the upgraded server and have older, yet unupgraded proxies report data to a newer server. This approach, which was never recommended nor supported by Zabbix, now is officially disabled, as the server will ignore data from unupgraded proxies.

Note that with SQLite database on proxies, history data from proxies before the upgrade will be lost, because SQLite database upgrade is not supported and the SQLite database file has to be manually removed. When proxy is started for the first time and the SQLite database file is missing, proxy creates it automatically.

Depending on database size the database upgrade to version 6.2 may take a long time.

Before the upgrade make sure to read the relevant upgrade notes!

The following upgrade notes are available:
<table>
<thead>
<tr>
<th>Upgrade from</th>
<th>Read full upgrade notes</th>
<th>Most important changes between versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.0.x LTS</td>
<td></td>
<td>Minimum required PHP version upped from 7.2.5 to 7.4.0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deterministic triggers need to be created during the upgrade. If binary logging is enabled for MySQL/MariaDB, this requires superuser privileges or setting the variable/configuration parameter log_bin_trust_function_creators = 1. See Database creation scripts for instructions how to set the variable.</td>
</tr>
<tr>
<td>5.4.x</td>
<td></td>
<td>Minimum required database versions upped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Server/proxy will not start if outdated database.</td>
</tr>
<tr>
<td>5.2.x</td>
<td></td>
<td>Audit log records lost because of database structure change.</td>
</tr>
<tr>
<td>5.0.x LTS</td>
<td></td>
<td>Minimum required PHP version upped from 7.2.0 to 7.2.5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Password hashing algorithm changed from MD5 to bcrypt.</td>
</tr>
<tr>
<td>4.4.x</td>
<td></td>
<td>Support of IBM DB2 dropped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Minimum required PHP version upped from 5.4.0 to 7.2.0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Minimum required database versions upped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed Zabbix PHP file directory.</td>
</tr>
<tr>
<td>4.2.x</td>
<td></td>
<td>Jabber, Ez Texting media types removed.</td>
</tr>
<tr>
<td>4.0.x LTS</td>
<td></td>
<td>Older proxies no longer can report data to an upgraded server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Newer agents no longer will be able to work with an older Zabbix server.</td>
</tr>
<tr>
<td>3.4.x</td>
<td></td>
<td>‘libpthread’ and ‘zlib’ libraries now mandatory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Support for plain text protocol dropped and header is mandatory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pre-1.4 version Zabbix agents are no longer supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Server parameter in passive proxy configuration now mandatory.</td>
</tr>
<tr>
<td>3.2.x</td>
<td></td>
<td>SQLite support as backend database dropped for Zabbix server/frontend.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Perl Compatible Regular Expressions (PCRE) supported instead of POSIX extended.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘libpcre’ and ‘libevent’ libraries mandatory for Zabbix server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Exit code checks added for user parameters, remote commands and system.run[] items without the ‘nowait’ flag as well as Zabbix server executed scripts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zabbix Java gateway has to be upgraded to support new functionality.</td>
</tr>
<tr>
<td>Upgrade from</td>
<td>Read full upgrade notes</td>
<td>Most important changes between versions</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>3.0.x LTS</td>
<td>For:</td>
<td>Database upgrade may be slow, depending on the history table size.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td></td>
</tr>
<tr>
<td>2.4.x</td>
<td>For:</td>
<td>Minimum required PHP version upped from 5.3.0 to 5.4.0</td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.0</td>
<td>LogFile agent parameter must be specified</td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td></td>
</tr>
<tr>
<td>2.2.x LTS</td>
<td>For:</td>
<td>Node-based distributed monitoring removed</td>
</tr>
<tr>
<td></td>
<td>Zabbix 2.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td></td>
</tr>
<tr>
<td>2.0.x</td>
<td>For:</td>
<td>Minimum required PHP version upped from 5.1.6 to 5.3.0.</td>
</tr>
<tr>
<td></td>
<td>Zabbix 2.2</td>
<td>Case-sensitive MySQL database required for proper server work; character set utf8 and utf8_bin collation is required for Zabbix server to work properly with MySQL database. See database creation scripts. ‘mysqli’ PHP extension required instead of ‘mysql’</td>
</tr>
<tr>
<td></td>
<td>Zabbix 2.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 3.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 4.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 5.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Zabbix 6.2</td>
<td></td>
</tr>
</tbody>
</table>

You may also want to check the requirements for 6.2.

It may be handy to run two parallel SSH sessions during the upgrade, executing the upgrade steps in one and monitoring the server/proxy logs in another. For example, run `tail -f zabbix_server.log` or `tail -f zabbix_proxy.log` in the second SSH session showing you the latest log file entries and possible errors in real time. This can be critical for production instances.

Server upgrade process

1 Stop server

Stop Zabbix server to make sure that no new data is inserted into database.
2 Back up the existing Zabbix database
This is a very important step. Make sure that you have a backup of your database. It will help if the upgrade procedure fails (lack of disk space, power off, any unexpected problem).

3 Back up configuration files, PHP files and Zabbix binaries
Make a backup copy of Zabbix binaries, configuration files and the PHP file directory.

4 Install new server binaries
Use these instructions to compile Zabbix server from sources.

5 Review server configuration parameters
See the upgrade notes for details on mandatory changes.
For new optional parameters, see the What’s new section.

6 Start new Zabbix binaries
Start new binaries. Check log files to see if the binaries have started successfully.
Zabbix server will automatically upgrade the database. When starting up, Zabbix server reports the current (mandatory and optional) and required database versions. If the current mandatory version is older than the required version, Zabbix server automatically executes the required database upgrade patches. The start and progress level (percentage) of the database upgrade is written to the Zabbix server log file. When the upgrade is completed, a “database upgrade fully completed” message is written to the log file. If any of the upgrade patches fail, Zabbix server will not start. Zabbix server will also not start if the current mandatory database version is newer than the required one. Zabbix server will only start if the current mandatory database version corresponds to the required mandatory version.

8673:20161117:104750.259 required mandatory version: 03040000

Before you start the server:
• Make sure the database user has enough permissions (create table, drop table, create index, drop index)
• Make sure you have enough free disk space.

7 Install new Zabbix web interface
The minimum required PHP version is 7.2.5. Update if needed and follow installation instructions.

8 Clear web browser cookies and cache
After the upgrade you may need to clear web browser cookies and web browser cache for the Zabbix web interface to work properly.

Proxy upgrade process
1 Stop proxy
Stop Zabbix proxy.

2 Back up configuration files and Zabbix proxy binaries
Make a backup copy of the Zabbix proxy binary and configuration file.

3 Install new proxy binaries
Use these instructions to compile Zabbix proxy from sources.

4 Review proxy configuration parameters
There are no mandatory changes in this version to proxy parameters.

5 Start new Zabbix proxy
Start the new Zabbix proxy. Check log files to see if the proxy has started successfully.
Zabbix proxy will automatically upgrade the database. Database upgrade takes place similarly as when starting Zabbix server.

Agent upgrade process
Upgrading agents is not mandatory. You only need to upgrade agents if it is required to access the new functionality.
The upgrade procedure described in this section may be used for upgrading both the Zabbix agent and the Zabbix agent 2.

1 Stop agent
Stop Zabbix agent.
2 Back up configuration files and Zabbix agent binaries
Make a backup copy of the Zabbix agent binary and configuration file.

3 Install new agent binaries
Use these instructions to compile Zabbix agent from sources.
Alternatively, you may download pre-compiled Zabbix agents from the Zabbix download page.

4 Review agent configuration parameters
There are no mandatory changes in this version neither to agent nor to agent 2 parameters.

5 Start new Zabbix agent
Start the new Zabbix agent. Check log files to see if the agent has started successfully.
Upgrade between minor versions
When upgrading between minor versions of 6.2.x (for example from 6.2.1 to 6.2.3) it is required to execute the same actions for server/proxy/agent as during the upgrade between major versions. The only difference is that when upgrading between minor versions no changes to the database are made.

8 Known issues

Proxy startup with MySQL 8.0.0-8.0.17
zabbix_proxy on MySQL versions 8.0.0-8.0.17 fails with the following “access denied” error:

[Z3001] connection to database 'zabbix' failed: [1227] Access denied; you need (at least one of) the SUPER
That is due to MySQL 8.0.0 starting to enforce special permissions for setting session variables. However, in 8.0.18 this behavior was removed: As of MySQL 8.0.18, setting the session value of this system variable is no longer a restricted operation.
The workaround is based on granting additional privileges to the zabbix user:
For MySQL versions 8.0.14 - 8.0.17:
grant SESSION_VARIABLES_ADMIN on *.* to 'zabbix'@'localhost';
For MySQL versions 8.0.0 - 8.0.13:
grant SYSTEM_VARIABLES_ADMIN on *.* to 'zabbix'@'localhost';

Timescale DB: high memory usage with a large number of partitions.
PostgreSQL versions 9.6-12 use too much memory when updating tables with a large number of partitions (see problem report). This issue manifests itself when Zabbix updates trends on systems with TimescaleDB if trends are split into relatively small (e.g. 1 day) chunks. This leads to hundreds of chunks present in the trends tables with default housekeeping settings - the condition where PostgreSQL is likely to run out of memory.
The issue has been resolved since Zabbix 5.0.1 for new installations with TimescaleDB, but if TimescaleDB was set up with Zabbix before that, please see ZBX-16347 for the migration notes.

Timescale DB 2.5.0: compression policy can fail on tables that contain integers
This issue manifests when TimescaleDB 2.5.0/2.5.1 is used. It has been resolved since TimescaleDB 2.5.2.
For more information, please see TimescaleDB Issue #3773.

Upgrade with MariaDB 10.2.1 and before
Upgrading Zabbix may fail if database tables were created with MariaDB 10.2.1 and before, because in those versions the default row format is compact. This can be fixed by changing the row format to dynamic (see also ZBX-17690).

Database TLS connection with MariaDB
Database TLS connection is not supported with the 'verify_ca' option for the DBTLSConnect parameter if MariaDB is used.

Global event correlation
Events may not get correlated correctly if the time interval between the first and second event is very small, i.e. half a second and less.

Numeric (float) data type range with PostgreSQL 11 and earlier
PostgreSQL 11 and earlier versions only support floating point value range of approximately \(-1.34\text{E}-154\) to \(1.34\text{E}+154\).

NetBSD 8.0 and newer

Various Zabbix processes may randomly crash on startup on the NetBSD versions 8.X and 9.X. That is due to the too small default stack size (4MB), which must be increased by running:

```
ulimit -s 10240
```

For more information, please see the related problem report: ZBX-18275.

IPMI checks

IPMI checks will not work with the standard OpenIPMI library package on Debian prior to 9 (stretch) and Ubuntu prior to 16.04 (xenial). To fix that, recompile OpenIPMI library with OpenSSL enabled as discussed in ZBX-6139.

SSH checks

Some Linux distributions like Debian, Ubuntu do not support encrypted private keys (with passphrase) if the libssh2 library is installed from packages. Please see ZBX-4850 for more details.

When using libssh 0.9.x on CentOS 8 with OpenSSH 8 SSH checks may occasionally report “Cannot read data from SSH server”. This is caused by a libssh issue (more detailed report). The error is expected to have been fixed by a stable libssh 0.9.5 release. See also ZBX-17756 for details.

ODBC checks

- MySQL unixODBC driver should not be used with Zabbix server or Zabbix proxy compiled against MariaDB connector library and vice versa, if possible it is also better to avoid using the same connector as the driver due to an upstream bug. Suggested setup:

  PostgreSQL, SQLite or Oracle connector → MariaDB or MySQL unixODBC driver
  MariaDB connector → MariaDB unixODBC driver
  MySQL connector → MySQL unixODBC driver

Please see ZBX-7665 for more information and available workarounds.

- XML data queried from Microsoft SQL Server may get truncated in various ways on Linux and UNIX systems.
- It has been observed that using ODBC checks for monitoring Oracle databases using various versions of Oracle Instant Client for Linux causes Zabbix server to crash. See also ZBX-18402, ZBX-20803.

Incorrect request method parameter in items

The request method parameter, used only in HTTP checks, may be incorrectly set to ‘1’, a non-default value for all items as a result of upgrade from a pre-4.0 Zabbix version. For details on how to fix this situation, see ZBX-19308.

Web monitoring and HTTP agent

Zabbix server leaks memory on CentOS 6, CentOS 7 and possibly other related Linux distributions due to an upstream bug when “SSL verify peer” is enabled in web scenarios or HTTP agent. Please see ZBX-10486 for more information and available workarounds.

Simple checks

There is a bug in fping versions earlier than v3.10 that mishandles duplicate echo replay packets. This may cause unexpected results for icmp<ping, icmp<pingloss, icmp<ping<sec items. It is recommended to use the latest version of fping. Please see ZBX-11726 for more details.

SNMP checks

If the OpenBSD operating system is used, a use-after-free bug in the Net-SNMP library up to the 5.7.3 version can cause a crash of Zabbix server if the SourceIP parameter is set in the Zabbix server configuration file. As a workaround, please do not set the SourceIP parameter. The same problem applies also for Linux, but it does not cause Zabbix server to stop working. A local patch for the net-snmp package on OpenBSD was applied and will be released with OpenBSD 6.3.

SNMP data spikes

Spikes in SNMP data have been observed that may be related to certain physical factors like voltage spikes in the mains. See ZBX-14318 more details.

SNMP traps

The “net-snmp-perl” package, needed for SNMP traps, has been removed in RHEL/CentOS 8.0-8.2; re-added in RHEL 8.3.

So if you are using RHEL 8.0-8.2, the best solution is to upgrade to RHEL 8.3; If you are using CentOS 8.0-8.2, you may wait for CentOS 8.3 or use a package from EPEL.
Please also see ZBX-17192 for more information.

Alerter process crash in Centos/RHEL 7

Instances of a Zabbix server alerter process crash have been encountered in Centos/RHEL 7. Please see ZBX-10461 for details.

Compiling Zabbix agent on HP-UX

If you install the PCRE library from a popular HP-UX package site http://hpux.connect.org.uk, for example from file pcre-8.42-ia64_64-11.3.55.0-depot, you get only the 64-bit version of the library installed in the /usr/local/lib/hpux64 directory.

In this case, for successful agent compilation customized options need to be used for the "configure" script, e.g.:

CFLAGS="-DD64" ./configure --enable-agent --with-libpcre-include=/usr/local/include --with-libpcre-lib=/usr/local/lib/hpux64

Flipping frontend locales

It has been observed that frontend locales may flip without apparent logic, i.e., some pages (or parts of pages) are displayed in one language while other pages (or parts of pages) in a different language. Typically the problem may appear when there are several users, some of whom use one locale, while others use another.

A known workaround to this is to disable multithreading in PHP and Apache.

The problem is related to how setting the locale works in PHP: locale information is maintained per process, not per thread. So in a multi-thread environment, when there are several projects run by same Apache process, it is possible that the locale gets changed in another thread and that changes how data can be processed in the Zabbix thread.

For more information, please see related problem reports:

- ZBX-10911 (Problem with flipping frontend locales)
- ZBX-16297 (Problem with number processing in graphs using the bcdiv function of BC Math functions)

PHP 7.3 opcache configuration

If "opcache" is enabled in the PHP 7.3 configuration, Zabbix frontend may show a blank screen when loaded for the first time. This is a registered PHP bug. To work around this, please set the "opcache.optimization_level" parameter to 0x7FFFBFDF in the PHP configuration (php.ini) file.

Graphs

Changes to Daylight Saving Time (DST) result in irregularities when displaying X axis labels (date duplication, date missing, etc).

Log file monitoring

log[] and logrt[] items repeatedly reread log file from the beginning if file system is 100% full and the log file is being appended (see ZBX-10884 for more information).

Slow MySQL queries

Zabbix server generates slow select queries in case of non-existing values for items. This is caused by a known issue in MySQL 5.6/5.7 versions. A workaround to this is disabling the index_condition_pushdown optimizer in MySQL. For an extended discussion, see ZBX-10652.

API login

A large number of open user sessions can be created when using custom scripts with the user/login method without a following user/logout.

IPv6 address issue in SNMPv3 traps

Due to a net-snmp bug, IPv6 address may not be correctly displayed when using SNMPv3 in SNMP traps. For more details and a possible workaround, see ZBX-14541.

Trimmed long IPv6 IP address in failed login information

A failed login attempt message will display only the first 39 characters of a stored IP address as that’s the character limit in the database field. That means that IPv6 IP addresses longer than 39 characters will be shown incompletely.

Zabbix agent checks on Windows

Non-existing DNS entries in a Server parameter of Zabbix agent configuration file (zabbix_agentd.conf) may increase Zabbix agent response time on Windows. This happens because Windows DNS caching daemon doesn’t cache negative responses for IPv4 addresses. However, for IPv6 addresses negative responses are cached, so a possible workaround to this is disabling IPv4 on the host.

YAML export/import

There are some known issues with YAML export/import:
Error messages are not translatable;
Valid JSON with a .yaml file extension sometimes cannot be imported;
Unquoted human-readable dates are automatically converted to Unix timestamps.

Setup wizard on SUSE with NGINX and php-fpm

Frontend setup wizard cannot save configuration file on SUSE with NGINX + php-fpm. This is caused by a setting in /usr/lib/systemd/system/php-fpm.service unit, which prevents Zabbix from writing to /etc. (introduced in PHP 7.4).

There are two workaround options available:

- Set the `ProtectSystem` option to 'true' instead of 'full' in the php-fpm systemd unit.
- Manually save /etc/zabbix/web/zabbix.conf.php file.

Chromium for Zabbix web service on Ubuntu 20

Though in most cases, Zabbix web service can run with Chromium, on Ubuntu 20.04 using Chromium causes the following error:

```
Cannot fetch data: chrome failed to start: cmd_run.go:994:
WARNING: cannot create user data directory: cannot create
"/var/lib/zabbix/snap/chromium/1564": mkdir /var/lib/zabbix: permission denied
Sorry, home directories outside of /home are not currently supported. See https://forum.snapcraft.io/t/11222/ for details.
```

This error occurs because /var/lib/zabbix is used as a home directory of user 'zabbix'.

MySQL custom error codes

If Zabbix is used with MySQL installation on Azure, an unclear error message [9002] Some errors occurred may appear in Zabbix logs. This generic error text is sent to Zabbix server or proxy by the database. To get more information about the cause of the error, check Azure logs.

Invalid regular expressions after switching to PCRE2

In Zabbix 6.0 support for PCRE2 has been added. Even though PCRE is still supported, Zabbix installation packages for RHEL/CentOS 7 and newer, SLES (all versions), Debian 9 and newer, Ubuntu 16.04 and newer have been updated to use PCRE2. While providing many benefits, switching to PCRE2 may cause certain existing PCRE regexp patterns becoming invalid or behaving differently. In particular, this affects the pattern `^\[\w\-\].` In order to make this regexp valid again without affecting semantics, change the expression to `^[\-\w\.]`. This happens due to the fact that PCRE2 treats the dash sign as a delimiter, creating a range inside a character class.

Geomap widget error

The maps in the Geomap widget may not load correctly, if you have upgraded from an older Zabbix version with NGINX and didn’t switch to the new NGINX configuration file during the upgrade.

To fix the issue, you can discard the old configuration file, use the configuration file from the current version package and reconfigure it as described in the download instructions in section e. Configure PHP for Zabbix frontend.

Alternatively, you can manually edit an existing NGINX configuration file (typically, /etc/zabbix/nginx.conf). To do so, open the file and locate the following block:

```
location ~ /(api\|\|conf[^\.]|include|locale|vendor) {
    deny all;
    return 404;
}
```

Then, replace this block with:

```
location ~ /(api\|\|conf[^\.]|include|locale) {
    deny all;
    return 404;
}
```

```
location /vendor {
    deny all;
    return 404;
}
```

9 Template changes
This page lists all changes to the stock templates that are shipped with Zabbix.

Note that upgrading to the latest Zabbix version will not automatically upgrade the templates used. It is suggested to modify the templates in existing installations by:

- Downloading the latest templates from the Zabbix Git repository;
- Then, while in Configuration → Templates you can import them manually into Zabbix. If templates with the same names already exist, the Delete missing options should be checked when importing to achieve a clean import. This way the old items that are no longer in the updated template will be removed (note that it will mean losing history of these old items).

CHANGES IN 6.2.0

New templates

See the list of new templates in Zabbix 6.2.0

CHANGES IN 6.2.1

A new template HPE Synergy by HTTP is available.

The templates HashiCorp Consul Node by HTTP and HashiCorp Consul Cluster by HTTP now support Consul namespaces.

10 Upgrade notes for 6.2.0

These notes are for upgrading from Zabbix 6.0.x to Zabbix 6.2.0. All notes are grouped into:

- Critical - the most critical information related to the upgrade process and the changes in Zabbix functionality
- Informational - all remaining information describing the changes in Zabbix functionality

It is possible to upgrade to Zabbix 6.2.0 from versions before Zabbix 6.0.0. See the upgrade procedure section for all relevant information about upgrading from previous Zabbix versions.

Critical

**Faster configuration sync**  Incremental configuration cache synchronization has been added for hosts, host tags, items, item tags, item preprocessing, triggers, trigger tags and functions to lessen synchronization time and database load when configuration is being updated on an already running Zabbix server or Zabbix proxy. As a result of this change, deterministic triggers need to be created during upgrade.

On MySQL and MariaDB, this requires `GLOBAL log_bin_trust_function_creators = 1` to be set if binary logging is enabled and there is no superuser privileges and `log_bin_trust_function_creators = 1` is not set in MySQL configuration file. To set the variable using MySQL console, run:

```
mysql> SET GLOBAL log_bin_trust_function_creators = 1;
```

Once the upgrade has been successfully completed, `log_bin_trust_function_creators` can be disabled:

```
mysql> SET GLOBAL log_bin_trust_function_creators = 0;
```

Triggers are also created for PostgreSQL and Oracle database.

**Minimum required PHP version**  The minimum required PHP version has been raised from 7.2.5 to 7.4.0.

**Internal items for history/trends removed**  The following internal items, deprecated since Zabbix 6.0, have now been removed:

- `zabbix[history]`
- `zabbix[history_log]`
- `zabbix[history_str]`
- `zabbix[history_text]`
- `zabbix[history_uint]`
- `zabbix[trends]`
- `zabbix[trends_uint]`

**History pollers removed from Zabbix proxy**  History pollers have been removed from Zabbix proxy.

Internal items that used to require a database connection (such as `zabbix[proxy,,lastaccass]`, `zabbix[proxy,,delay]` and `zabbix[proxy_history]`) and were polled by history pollers on Zabbix server or Zabbix proxy have been reconfigured to be polled by regular pollers and use data from the configuration cache instead.
Secure password hashing  In Zabbix 5.0 password hashing algorithm has been changed from MD5 to the more secure bcrypt. However, in Zabbix versions 5.0 - 6.0, MD5 hashing was still used upon the first user login after an upgrade - to convert passwords with hashes not exceeding 32 bytes from MD5 to bcrypt. Now support of MD5 cryptography has been dropped completely.

If you’re upgrading from Zabbix versions before 5.0, users with passwords hashed by MD5 won’t be able to log in. In this case, a Super administrator can change passwords of the affected users. If a Super administrator also cannot log in, run the following SQL query to apply the default password to the user (replace ‘Admin’ with the required username):

```sql
UPDATE users SET passwd = '$2a$10$8ZXIvHAEP22M.dLXTm6uPHOMVlARXX7cqjjbhm6Fm0cANzkCQBwMrS' WHERE username = 'Admin';
```

After running this query, the user’s password will be set to zabbix. Make sure to change the default password on the first login.

Storage of secrets  In addition to HashiCorp Vault, Zabbix now supports storage of secrets in CyberArk Vault. To distinguish between secret management platforms, a new parameter $DB['VAULT'] has been added to zabbix.conf.php.

If your Zabbix installation has been configured to work with HashiCorp Vault, after the upgrade you will need to manually update the configuration file. To continue using the HashiCorp integration, add to zabbix.conf.php the variable:

```php
$DB['VAULT'] = 'HashiCorp';
```

Additionally, the database credentials will no longer be cached by default. Instead, Zabbix will make a call to the vault API every time when establishing a database connection. To enable storing of retrieved credentials in a local cache, you now need to manually set the option $DB['VAULT_CACHE'] = true.

For more info, see Storage of secrets.

CurlHttpRequest removed  The CurlHttpRequest additional JavaScript object that was renamed to HttpRequest in Zabbix 5.4, and had been deprecated since, has now been removed.

API changes

See the list of API changes in Zabbix 6.2.0.

Informational

Immediate checks for new items  Previously, newly added items were first checked at a random time within their update interval. Now new items, web scenarios and discovery rules, will be executed within 60 seconds of their creation, unless they have Scheduling or Flexible update interval with the Update interval parameter set to 0.

Separate groups for templates  Host group functionality has been split into template groups, which may contain templates only and host groups, which may contain hosts only. During the upgrade all host groups that contained templates only will be automatically converted into template groups. Host and template groups will retain the same UUIDs.

If a host group contains both hosts and templates, such group will be split into two groups with the same name: a template group and a host group. If a group doesn’t contain hosts, but is referenced somewhere in the hosts-related configuration (for example, used by a host prototype or in an action operation), an empty host group will be created.

After the upgrade, all users whose role allows access to Configuration -> Host groups menu section will get access to the new Configuration -> Template groups section. For user groups, existing permission sets will be automatically split into host and template group permissions.

11 Upgrade notes for 6.2.1

Symlink name expansion  Symlink name and full path of the symlink are now returned in vfs.dir.get[ ] and vfs.file.get[ ] items, instead of resolving to the symlink target.

5. Quickstart

Please use the sidebar to access content in the Quickstart section.
1 Login and configuring user

Overview

In this section, you will learn how to log in and set up a system user in Zabbix.

Login

This is the Zabbix welcome screen. Enter the username **Admin** with password **zabbix** to log in as a Zabbix superuser. Access to Configuration and Administration menus will be granted.

Protection against brute force attacks

In case of five consecutive failed login attempts, Zabbix interface will pause for 30 seconds in order to prevent brute force and dictionary attacks.

The IP address of a failed login attempt will be displayed after a successful login.

Adding user

To view information about users, go to Administration → Users.

To add a new user, click on Create user.

In the new user form, make sure to add your user to one of the existing user groups, for example 'Zabbix administrators'.
All mandatory input fields are marked with a red asterisk.

By default, new users have no media (notification delivery methods) defined for them. To create one, go to the 'Media' tab and click on Add.

In this pop-up, enter an e-mail address for the user.

You can specify a time period when the medium will be active (see Time period specification page for a description of the format), by default a medium is always active. You can also customize trigger severity levels for which the medium will be active, but leave all of them enabled for now.

Click on Add to save the medium, then go to the Permissions tab.

Permissions tab has a mandatory field Role. The role determines which frontend elements the user can view and which actions he
is allowed to perform. Press Select and select one of the roles from the list. For example, select Admin role to allow access to all Zabbix frontend sections, except Administration. Later on, you can modify permissions or create more user roles. Upon selecting a role, permissions will appear in the same tab:

<table>
<thead>
<tr>
<th>Role</th>
<th>Admin role</th>
<th>Select</th>
</tr>
</thead>
</table>

User type: Admin

Permissions:

<table>
<thead>
<tr>
<th>Group</th>
<th>Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>All groups</td>
<td>Hosts</td>
<td>None</td>
</tr>
<tr>
<td>All groups</td>
<td>Templates</td>
<td>None</td>
</tr>
</tbody>
</table>

Permissions can be assigned for user groups only.

Access to UI elements

- **Monitoring**: Dashboard, Problems, Hosts, Latest data, Maps, Discovery
- **Services**: Services, Service actions, SLA, SLA report
- **Inventory**: Overview, Hosts
- **Reports**: Availability report, Triggers top 100, Notifications, Scheduled reports
- **Configuration**: Template groups, Host groups, Templates, Hosts, Maintenance, Actions, Discovery

Access to services

- Read-write access to services: All
- Read-only access to services: All

Access to modules

*No enabled modules found.*

Access to API

- **Enabled**

Access to actions

- Create and edit dashboards
- Create and edit maps
- Create and edit maintenance
- Add problem comments
- Change severity
- Acknowledge problems
- Close problems
- Execute scripts
- Manage API tokens
- Manage scheduled reports
- Manage SLA
- Invoke "Execute now" on read-only hosts

Add or Cancel

Click Add in the user properties form to save the user. The new user appears in the userlist.

Adding permissions

By default, a new user has no permissions to access hosts and templates. To grant the user rights, click on the group of the user in the Groups column (in this case - 'Zabbix administrators'). In the group properties form, go to the Host permissions tab to assign permissions to host groups.
This user is to have read-only access to Linux servers group, so click on Select next to the user group selection field.

In this pop-up, mark the checkbox next to ‘Linux servers’, then click Select. Linux servers should be displayed in the selection field. Click the ‘Read’ button to set the permission level and then Add to add the group to the list of permissions. In the user group properties form, click Update.

To grant permissions to templates, you will need to switch to the Template permissions tab and specify template groups.

In Zabbix, access rights to hosts and templates are assigned to user groups, not individual users.

Done! You may try to log in using the credentials of the new user.

2 New host

Overview

In this section you will learn how to set up a new host.

A host in Zabbix is a networked entity (physical, virtual) that you wish to monitor. The definition of what can be a “host” in Zabbix is quite flexible. It can be a physical server, a network switch, a virtual machine or some application.

Adding host

Information about configured hosts in Zabbix is available in Configuration → Hosts and Monitoring → Hosts. There is already one pre-defined host, called “Zabbix server”, but we want to learn adding another.

To add a new host, click on Create host. This will present us with a host configuration form.
All mandatory input fields are marked with a red asterisk.

The bare minimum to enter here is:

**Host name**

- Enter a host name. Alphanumerics, spaces, dots, dashes and underscores are allowed.

**Host groups**

- Select one or several existing groups by clicking Select button or enter a non-existing group name to create a new group.

All access permissions are assigned to host groups, not individual hosts. That is why a host must belong to at least one group.

**Interfaces: IP address**

- Although not a required field technically, you may want to enter the IP address of the host. Note that if this is the Zabbix server IP address, it must be specified in the Zabbix agent configuration file ‘Server’ directive.

**Other options** will suit us with their defaults for now.

When done, click Add. Your new host should be visible in the host list.

The Availability column contains indicators of host availability per each interface. We have defined a Zabbix agent interface, so we can use the agent availability icon (with ‘ZBX’ on it) to understand host availability:

- host status has not been established; no metric check has happened yet
- host is available, a metric check has been successful
- host is unavailable, a metric check has failed (move your mouse cursor over the icon to see the error message).

There might be some error with communication, possibly caused by incorrect interface credentials. Check that Zabbix server is running, and try refreshing the page later as well.

3 New item

**Overview**

In this section, you will learn how to set up an item.
Items are the basis of gathering data in Zabbix. Without items, there is no data - because only an item defines a single metric or what kind of data to collect from a host.

Adding item

All items are grouped around hosts. That is why to configure a sample item we go to Configuration → Hosts and find the "New host" we have created.

Click on the Items link in the row of "New host", and then click on Create item. This will present us with an item definition form.

All mandatory input fields are marked with a red asterisk.

For our sample item, the essential information to enter is:

**Name**
- Enter CPU load as the value. This will be the item name displayed in lists and elsewhere.

**Key**
- Manually enter system.cpu.load as the value. This is the technical name of an item that identifies the type of information that will be gathered. The particular key is just one of pre-defined keys that come with Zabbix agent.

**Type of information**
- This attribute defines the format of the expected data. For the system.cpu.load key, this field will be automatically set to Numeric (float).

You may also want to reduce the number of days item history will be kept, to 7 or 14. This is good practice to relieve the database from keeping lots of historical values.

**Other options** will suit us with their defaults for now.

When done, click Add. The new item should appear in the item list. Click on Details above the list to view what exactly was done.
Seeing data

With an item defined, you might be curious if it is actually gathering data. For that, go to Monitoring → Latest data, select ‘New host’ in the filter and click on Apply.

With that said, it may take up to 60 seconds for the first data to arrive. That, by default, is how often the server reads configuration changes and picks up new items to execute.

If you see no value in the ‘Change’ column, maybe only one value has been received so far. Wait 30 seconds for another value to arrive.

If you do not see information about the item as in the screenshot, make sure that:

- you have filled out the item ‘Key’ and ‘Type of information’ fields exactly as in the screenshot
- both the agent and the server are running
- host status is ‘Monitored’ and its availability icon is green
- a host is selected in the host dropdown, the item is active

Graphs

With the item working for a while, it might be time to see something visual. Simple graphs are available for any monitored numeric item without any additional configuration. These graphs are generated on runtime.

To view the graph, go to Monitoring → Latest data and click on the ‘Graph’ link next to the item.

4 New trigger

Overview

In this section you will learn how to set up a trigger.

Items only collect data. To automatically evaluate incoming data we need to define triggers. A trigger contains an expression that defines a threshold of what is an acceptable level for the data.

If that level is surpassed by the incoming data, a trigger will “fire” or go into a ‘Problem’ state - letting us know that something has happened that may require attention. If the level is acceptable again, trigger returns to an ‘Ok’ state.

Adding trigger

To configure a trigger for our item, go to Configuration → Hosts, find ‘New host’ and click on Triggers next to it and then on Create trigger. This presents us with a trigger definition form.
For our trigger, the essential information to enter here is:

Name

- Enter CPU load too high on 'New host' for 3 minutes as the value. This will be the trigger name displayed in lists and elsewhere.

Expression

- Enter: \texttt{avg(/New host/system.cpu.load,3m)>2}

This is the trigger expression. Make sure that the expression is entered right, down to the last symbol. The item key here (system.cpu.load) is used to refer to the item. This particular expression basically says that the problem threshold is exceeded when the CPU load average value for 3 minutes is over 2. You can learn more about the syntax of trigger expressions.

When done, click Add. The new trigger should appear in the trigger list.

Displaying trigger status

With a trigger defined, you might be interested to see its status.

If the CPU load has exceeded the threshold level you defined in the trigger, the problem will be displayed in Monitoring → Problems.
The flashing in the status column indicates a recent change of trigger status, one that has taken place in the last 30 minutes.

5 Receiving problem notification

Overview

In this section you will learn how to set up alerting in the form of notifications in Zabbix.

With items collecting data and triggers designed to “fire” upon problem situations, it would also be useful to have some alerting mechanism in place that would notify us about important events even when we are not directly looking at Zabbix frontend.

This is what notifications do. E-mail being the most popular delivery method for problem notifications, we will learn how to set up an e-mail notification.

E-mail settings

Initially there are several predefined notification delivery methods in Zabbix. E-mail is one of those.

To configure e-mail settings, go to Administration → Media types and click on Email in the list of pre-defined media types.

Media types

- **Email**: Email delivery method. Status is **Enabled**
  - **Details**: SMTP server: "mail.zabbix.com"

- **Mattermost**: Webhook delivery method. Status is **Enabled**

- **Opsgenie**: Webhook delivery method. Status is **Enabled**

This will present us with the e-mail settings definition form.
All mandatory input fields are marked with a red asterisk.

Set the values of SMTP server, SMTP helo and SMTP e-mail to the appropriate for your environment.

‘SMTP email’ will be used as the ‘From’ address for the notifications sent from Zabbix.

Press Update when ready.

Now you have configured ‘Email’ as a working media type. A media type must be linked to users by defining specific delivery addresses (like we did when configuring a new user), otherwise it will not be used.

New action

Delivering notifications is one of the things actions do in Zabbix. Therefore, to set up a notification, go to Configuration → Actions and click on Create action.
All mandatory input fields are marked with a red asterisk.

In this form, enter a name for the action.

In the most simple case, if we do not add any more specific conditions, the action will be taken upon any trigger change from ‘Ok’ to ‘Problem’.

We still should define what the action should do - and that is done in the Operations tab. Click on Add in the Operations block, which opens a new operation form.
All mandatory input fields are marked with a red asterisk.

Here, click on Add in the Send to Users block and select the user (‘user’) we have defined. Select ‘Email’ as the value of Send only to. When done with this, click on Add, and the operation should be added:

That is all for a simple action configuration, so click Add in the action form.

Receiving notification

Now, with delivering notifications configured it would be fun to actually receive one. To help with that, we might on purpose increase the load on our host - so that our trigger “fires” and we receive a problem notification.

Open the console on your host and run:
cat /dev/urandom | md5sum

You may run one or several of these processes.

Now go to Monitoring → Latest data and see how the values of ‘CPU Load’ have increased. Remember, for our trigger to fire, the ‘CPU Load’ value has to go over ‘2’ for 3 minutes running. Once it does:

- in Monitoring → Problems you should see the trigger with a flashing ‘Problem’ status
- you should receive a problem notification in your e-mail

If notifications do not work:

- verify once again that both the e-mail settings and the action have been configured properly
- make sure the user you created has at least read permissions on the host which generated the event, as noted in the Adding user step. The user, being part of the ‘Zabbix administrators’ user group must have at least read access to ‘Linux servers’ host group that our host belongs to.
- Additionally, you can check out the action log by going to Reports → Action log.

6 New template

Overview

In this section you will learn how to set up a template.

Previously we learned how to set up an item, a trigger and how to get a problem notification for the host.

While all of these steps offer a great deal of flexibility in themselves, it may appear like a lot of steps to take if needed for, say, a thousand hosts. Some automation would be handy.

This is where templates come to help. Templates allow to group useful items, triggers and other entities so that those can be reused again and again to hosts in a single step.

When a template is linked to a host, the host inherits all entities of the template. So, basically a pre-prepared bunch of checks can be applied very quickly.

Adding template

To start working with templates, we must first create one. To do that, in Configuration → Templates click on Create template. This will present us with a template configuration form.

<table>
<thead>
<tr>
<th>Templates</th>
<th>Tags</th>
<th>Macros</th>
<th>Value mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Template name</td>
<td>New template</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visible name</td>
<td>New template</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Templates</td>
<td></td>
<td>type here to search</td>
<td></td>
</tr>
<tr>
<td>* Template groups</td>
<td>Templates</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>type here to search</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All mandatory input fields are marked with a red asterisk.

The required parameters to enter here are:

Template name
• Enter a template name. Alpha-numericals, spaces and underscores are allowed.

**Template groups**

• Select one or several groups by clicking Select button. The template must belong to a group.

Access permissions to template groups are assigned in the user group configuration on the Template permissions tab in the same way as host permissions. All access permissions are assigned to groups, not individual templates, that’s why including the template into at least one group is mandatory.

When done, click Add. Your new template should be visible in the list of templates.

As you may see, the template is there, but it holds nothing in it - no items, triggers or other entities.

**Adding item to template**

To add an item to the template, go to the item list for 'New host'. In Configuration → Hosts click on Items next to 'New host'.

Then:

• mark the checkbox of the 'CPU Load' item in the list
• click on Copy below the list
• select the template to copy item to

All mandatory input fields are marked with a red asterisk.

• click on Copy

If you now go to Configuration → Templates, 'New template' should have one new item in it.

We will stop at one item only for now, but similarly you can add any other items, triggers or other entities to the template until it’s a fairly complete set of entities for given purpose (monitoring OS, monitoring single application).

**Linking template to host**

With a template ready, it only remains to add it to a host. For that, go to Configuration → Hosts, click on 'New host' to open its property form and find the Templates field.

Start typing New template in the Templates field. The name of template we have created should appear in the dropdown list. Scroll down to select. See that it appears in the Templates field.

Click Update in the form to save the changes. The template is now added to the host, with all entities that it holds.
This way it can be applied to any other host as well. Any changes to the items, triggers and other entities at the template level will propagate to the hosts the template is linked to.

Linking pre-defined templates to hosts
As you may have noticed, Zabbix comes with a set of predefined templates for various OS, devices and applications. To get started with monitoring very quickly, you may link the appropriate one of them to a host, but beware that these templates need to be fine-tuned for your environment. Some checks may not be needed, and polling intervals may be way too frequent.

More information about templates is available.

6. Zabbix appliance

Overview  As an alternative to setting up manually or reusing an existing server for Zabbix, users may download a Zabbix appliance or a Zabbix appliance installation CD image.

Zabbix appliance and installation CD versions are based on CentOS 8 (x86_64).

Zabbix appliance installation CD can be used for instant deployment of Zabbix server (MySQL).

You can use this Appliance to evaluate Zabbix. The Appliance is not intended for serious production use.

System requirements:

- RAM: 1.5 GB
- Disk space: at least 8 GB should be allocated for the virtual machine.

Zabbix installation CD/DVD boot menu:

Zabbix appliance contains a Zabbix server (configured and running on MySQL) and a frontend.

Zabbix virtual appliance is available in the following formats:

- VMWare (.vmx)
- Open virtualization format (.ovf)
- Microsoft Hyper-V 2012 (.vhdx)
- Microsoft Hyper-V 2008 (.vhd)
• KVM, Parallels, QEMU, USB stick, VirtualBox, Xen (.raw)
• KVM, QEMU (.qcow2)

To get started, boot the appliance and point a browser at the IP the appliance has received over DHCP.

DHCP must be enabled on the host.

To get the IP address from inside the virtual machine run:

```
  ip addr show
```

To access Zabbix frontend, go to `http://<host_ip>` (for access from the host’s browser bridged mode should be enabled in the VM network settings).

If the appliance fails to start up in Hyper-V, you may want to press Ctrl+Alt+F2 to switch tty sessions.

**1 Changes to CentOS 8 configuration**  The appliance is based on AlmaLinux 8.

**1.1 Repositories**

Official Zabbix repository has been added to `/etc/yum.repos.d`:

```
[zabbix]
name=Zabbix Official Repository - $basearch
baseurl=http://repo.zabbix.com/zabbix/6.2/rhel/8/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-ZABBIX-A14FE591
```

**1.2 Firewall configuration**

The appliance uses iptables firewall with predefined rules:

- Opened SSH port (22 TCP);
- Opened Zabbix agent (10050 TCP) and Zabbix trapper (10051 TCP) ports;
- Opened HTTP (80 TCP) and HTTPS (443 TCP) ports;
- Opened SNMP trap port (162 UDP);
- Opened outgoing connections to NTP port (53 UDP);
- ICMP packets limited to 5 packets per second;
- All other incoming connections are dropped.

**1.3 Using a static IP address**

By default the appliance uses DHCP to obtain the IP address. To specify a static IP address:

- Log in as root user;
- Open `/etc/sysconfig/network-scripts/ifcfg-eth0` file;
- Replace `BOOTPROTO=dhcp` with `BOOTPROTO=none`
- Add the following lines:
  - `IPADDR=<IP address of the appliance>`
  - `PREFIX=<CIDR prefix>`
  - `GATEWAY=<gateway IP address>`
  - `DNS1=<DNS server IP address>`
- Run `systemctl restart network` command.

Consult the official Red Hat documentation if needed.

**1.4 Changing timezone**

By default the appliance uses UTC for the system clock. To change the timezone, copy the appropriate file from `/usr/share/zoneinfo` to `/etc/localtime`, for example:

```
  cp /usr/share/zoneinfo/Europe/Riga /etc/localtime
```

**2 Zabbix configuration**  Zabbix appliance setup has the following passwords and configuration changes:

**2.1 Credentials (login:password)**

System:

- root:zabbix

Zabbix frontend:

- Admin:zabbix
Database:

- root:<random>
- zabbix:<random>

Database passwords are randomly generated during the installation process. Root password is stored inside the /root/.my.cnf file. It is not required to input a password under the “root” account.

To change the database user password, changes have to be made in the following locations:

- MySQL:
  - /etc/zabbix/zabbix_server.conf;
  - /etc/zabbix/web/zabbix.conf.php.

Separate users zabbix_srv and zabbix_web are defined for the server and the frontend respectively.

2.2 File locations

- Configuration files are located in /etc/zabbix.
- Zabbix server, proxy and agent logfiles are located in /var/log/zabbix.
- Zabbix frontend is located in /usr/share/zabbix.
- Home directory for the user zabbix is /var/lib/zabbix.

2.3 Changes to Zabbix configuration

- Frontend timezone is set to Europe/Riga (this can be modified in /etc/php-fpm.d/zabbix.conf);

3 Frontend access

By default, access to the frontend is allowed from anywhere.

The frontend can be accessed at http://<host>.

This can be customized in /etc/nginx/conf.d/zabbix.conf. Nginx has to be restarted after modifying this file. To do so, log in using SSH as root user and execute:

```
systemctl restart nginx
```

4 Firewall

By default, only the ports listed in the configuration changes above are open. To open additional ports, modify “/etc/sysconfig/iptables” file and reload firewall rules:

```
systemctl reload iptables
```

5 Upgrading

The Zabbix appliance packages may be upgraded. To do so, run:

```
dnf update zabbix*
```

6 System Services

Systemd services are available:

```
systemctl list-units zabbix*
```

7 Format-specific notes

7.1 VMware

The images in vmdk format are usable directly in VMware Player, Server and Workstation products. For use in ESX, ESXi and vSphere they must be converted using VMware converter.

7.2 HDD/flash image (raw)

```
dd if=./zabbix_appliance_5.2.0.raw of=/dev/sdc bs=4k conv=fdatasync
```

Replace /dev/sdc with your Flash/HDD disk device.

7. Configuration

Please use the sidebar to access content in the Configuration section.
1 Configuring a template

Overview

Configuring a template requires that you first create a template by defining its general parameters and then you add entities (items, triggers, graphs, etc.) to it.

Creating a template

To create a template, do the following:

- Go to Configuration → Templates
- Click on Create template
- Edit template attributes

The Templates tab contains general template attributes.

<table>
<thead>
<tr>
<th>Templates</th>
<th>Tags</th>
<th>Macros</th>
<th>Value mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Template name</td>
<td>Linux</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visible name</td>
<td>Linux</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Templates</td>
<td>type here to search</td>
<td>Select</td>
<td></td>
</tr>
<tr>
<td>* Template groups</td>
<td>Operating systems (new)</td>
<td>Select</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>type here to search</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All mandatory input fields are marked with a red asterisk.

Template attributes:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template name</td>
<td>Unique template name. Alphanumerics, spaces, dots, dashes, and underscores are allowed. However, leading and trailing spaces are disallowed.</td>
</tr>
<tr>
<td>Visible name</td>
<td>If you set this name, it will be the one visible in lists, maps, etc.</td>
</tr>
<tr>
<td>Templates</td>
<td>Link one or more “nested” templates to this template. All entities (items, triggers, graphs, etc.) will be inherited from the linked templates. To link a new template, start typing the template name in the Link new templates field. A list of matching templates will appear; scroll down to select. Alternatively, you may click on Select next to the field and select templates from the list in a popup window. The templates that are selected in the Link new templates field will be linked to the template when the template configuration form is saved or updated. To unlink a template, use one of the two options in the Linked templates block: Unlink - unlink the template, but preserve its items, triggers, and graphs Unlink and clear - unlink the template and remove all its items, triggers, and graphs</td>
</tr>
<tr>
<td>Template groups</td>
<td>Groups the template belongs to.</td>
</tr>
<tr>
<td>Description</td>
<td>Enter the template description.</td>
</tr>
</tbody>
</table>

The Tags tab allows you to define template-level tags. All problems of hosts linked to this template will be tagged with the values entered here.
User macros, \{INVENTORY.*\} macros, \{HOST.HOST\}, \{HOST.NAME\}, \{HOST.CONNECT\}, \{HOST.DNS\}, \{HOST.IP\}, \{HOST.PORT\} and \{HOST.ID\} macros are supported in tags.

The **Macros** tab allows you to define template-level user macros as a name-value pairs. Note that macro values can be kept as plain text, secret text, or Vault secret. Adding a description is also supported.

You may also view here macros from linked templates and global macros if you select the Inherited and template macros option. That is where all defined user macros for the template are displayed with the value they resolve to as well as their origin.
For convenience, links to respective templates and global macro configuration are provided. It is also possible to edit a nested template/global macro on the template level, effectively creating a copy of the macro on the template.

The **Value mapping** tab allows to configure human-friendly representation of item data in value mappings.

**Buttons:**

- **Add**
  - Add the template. The added template should appear in the list.

- **Update**
  - Update the properties of an existing template.

- **Clone**
  - Create another template based on the properties of the current template, including the entities (items, triggers, etc) inherited from linked templates.

- **Full clone**
  - Create another template based on the properties of the current template, including the entities (items, triggers, etc) both inherited from linked templates and directly attached to the current template.

- **Delete**
  - Delete the template; entities of the template (items, triggers, etc) remain with the linked hosts.

- **Clear history and trends**
  - Delete the template and all its entities from linked hosts.

- **Cancel**
  - Cancel the editing of template properties.

With a template created, it is time to add some entities to it.

**Items have to be added to a template first. Triggers and graphs cannot be added without the corresponding item.**

**Adding items, triggers, graphs**

**To add items to the template, do the following:**

- Go to Configuration → Hosts (or Templates)
- Click on items in the row of the required host/template
- Mark the checkboxes of items you want to add to the template
- Click on Copy below the item list
Select the template (or group of templates) the items should be copied to and click on Copy. All the selected items should be copied to the template.

Adding triggers and graphs is done in a similar fashion (from the list of triggers and graphs respectively), again, keeping in mind that they can only be added if the required items are added first.

Adding dashboards

To add dashboards to a template in Configuration → Templates, do the following:

- Click on Dashboards in the row of the template
- Configure a dashboard following the guidelines of configuring dashboards

The widgets that can be included in a template dashboard are: classic graph, graph prototype, clock, plain text, URL.

For details on accessing host dashboards that are created from template dashboards, see the host dashboard section.

Configuring low-level discovery rules

See the low-level discovery section of the manual.

Adding web scenarios

To add web scenarios to a template in Configuration → Templates, do the following:

- Click on Web in the row of the template
- Configure a web scenario following the usual method of configuring web scenarios

Creating a template group

Only Super Admin users can create template groups.

To create a template group in Zabbix frontend, do the following:

- Go to: Configuration → Template groups
- Click on Create template group in the upper right corner of the screen
- Enter the group name in the form

To create a nested template group, use the ‘/’ forward slash separator, for example Linux servers/Databases/MySQL. You can create this group even if none of the two parent template groups (Linux servers/Databases/) exist. In this case creating these parent template groups is up to the user; they will not be created automatically. Leading and trailing slashes, several slashes in a row are not allowed. Escaping of ‘/’ is not supported.

Once the group is created, you can click on the group name in the list to edit group name, clone the group or set additional option:

Apply permissions to all subgroups - mark this checkbox and click on Update to apply the same level of permissions to all nested template groups. For user groups that may have had differing permissions assigned to nested template groups, the permission level of the parent template group will be enforced on the nested groups. This is a one-time option that is not saved in the database.
Permissions to nested template groups

- When creating a child template group to an existing parent template group, user group permissions to the child are inherited from the parent (for example, when creating Databases/MySQL if Databases already exists)
- When creating a parent template group to an existing child template group, no permissions to the parent are set (for example, when creating Databases if Databases/MySQL already exists)

2 Linking/unlinking

Overview

Linking is a process whereby templates are applied to hosts, whereas unlinking removes the association with the template from a host.

Linking a template

To link a template to the host, do the following:

- Go to Configuration → Hosts
- Click on the required host
- Start typing the template name in the Templates field. A list of matching templates will appear; scroll down to select.
- Alternatively, you may click on Select next to the field and select one or several templates from the list in a popup window
- Click on Add/Update in the host attributes form

The host will now have all the entities (items, triggers, graphs etc.) of the template.

Linking multiple templates to the same host will fail if in those templates there are items with the same item key. And, as triggers and graphs use items, they cannot be linked to a single host from multiple templates either, if using identical item keys.

When entities (items, triggers, graphs etc.) are added from the template:

- previously existing identical entities on the host are updated as entities of the template
- entities from the template are added
- any directly linked entities that, prior to template linkage, existed only on the host remain untouched

In the lists, all entities from the template now are prefixed by the template name, indicating that these belong to the particular template. The template name itself (in gray text) is a link allowing to access the list of those entities on the template level.

If some entity (item, trigger, graph etc.) is not prefixed by the template name, it means that it existed on the host before and was not added by the template.

Entity uniqueness criteria

When adding entities (items, triggers, graphs etc.) from a template it is important to know what of those entities already exist on the host and need to be updated and what entities differ. The uniqueness criteria for deciding upon the sameness/difference are:

- for items - the item key
- for triggers - trigger name and expression
- for custom graphs - graph name and its items

Linking templates to several hosts

To update template linkage of many hosts, in Configuration → Hosts select some hosts by marking their checkboxes, then click on Mass update below the list and then select Link templates:
To link additional templates, start typing the template name in the auto-complete field until a dropdown appears offering the matching templates. Just scroll down to select the template to link.

The Replace option will allow to link a new template while unlinking any template that was linked to the hosts before. The Unlink option will allow to specify which templates to unlink. The Clear when unlinking option will allow to not only unlink any previously linked templates, but also remove all elements inherited from them (items, triggers, etc.).

Zabbix offers a sizable set of predefined templates. You can use these for reference, but beware of using them unchanged in production as they may contain too many items and poll for data too often. If you feel like using them, finetune them to fit your real needs.

Editing linked entities

If you try to edit an item or trigger that was linked from the template, you may realize that many key options are disabled for editing. This makes sense as the idea of templates is that things are edited in one-touch manner on the template level. However, you still can, for example, enable/disable an item on the individual host and set the update interval, history length and some other parameters.

If you want to edit the entity fully, you have to edit it on the template level (template level shortcut is displayed in the form name), keeping in mind that these changes will affect all hosts that have this template linked to them.

Unlinking a template

To unlink a template from a host, do the following:

- Go to Configuration → Hosts
- Click on the required host and find the Templates field
- Click on Unlink or Unlink and clear next to the template to unlink
- Click on Update in the host attributes form

Choosing the Unlink option will simply remove association with the template, while leaving all its entities (items, triggers, graphs etc.) with the host.

Choosing the Unlink and clear option will remove both the association with the template and all its entities (items, triggers, graphs etc.).

3 Nesting

Overview

Nesting is a way of one template encompassing one or more other templates. As it makes sense to separate out on individual templates entities for various services, applications etc. you may end up with quite a few templates all of which may need to be linked to quite a few hosts. To simplify the picture, it is possible to link some templates together, in one “nested” template.

The benefit of nesting is that then you have to link only the one template to the host and the host will inherit all entities of the linked templates automatically.

Configuring a nested template

If you want to link some templates, to begin with you can take an existing template or a new one, then:

- Open the template properties form
- Find the Templates field
- Click on Select to select templates in the popup window
- Click on Add to list selected templates
- Click on Add/Update in the template properties form

Thus, all entities of the template itself, as well as all linked templates’ entities (such as items, triggers, graphs etc.) will appear in the template configuration, except for the linked template dashboards, which will nevertheless be inherited by hosts.

To unlink any of the linked templates, in the same form use the Unlink or Unlink and clear buttons and click on Update.

Choosing the Unlink option will simply remove the association with the other template, while not removing all its entities (items, triggers, graphs etc.).

Choosing the Unlink and clear option will remove both the association with the other template and all its entities (items, triggers, graphs etc.).
4 Mass update

Overview

Sometimes you may want to change some attribute for a number of templates at once. Instead of opening each individual template for editing, you may use the mass update function for that.

Using mass update

To mass-update some templates, do the following:

- Mark the checkboxes before the templates you want to update in the template list
- Click on Mass update below the list
- Navigate to the tab with required attributes (Template, Tags, Macros or Value mapping)
- Mark the checkboxes of any attribute to update and enter a new value for them

The following options are available when selecting the respective button for template linkage update:

- Link - specify which additional templates to link
- Replace - specify which templates to link while unlinking any template that was linked to the templates before
- Unlink - specify which templates to unlink

To specify the templates to link/unlink start typing the template name in the auto-complete field until a dropdown appears offering the matching templates. Just scroll down to select the required template.

The Clear when unlinking option will allow to not only unlink any previously linked templates, but also remove all elements inherited from them (items, triggers, etc.).

The following options are available when selecting the respective button for template group update:

- Add - allows to specify additional template groups from the existing ones or enter completely new template groups for the templates
- Replace - will remove the template from any existing template groups and replace them with the one(s) specified in this field (existing or new template groups)
- Remove - will remove specific template groups from templates

These fields are auto-complete - starting to type in them offers a dropdown of matching template groups. If the template group is new, it also appears in the dropdown and it is indicated by (new) after the string. Just scroll down to select.
User macros, \{INVENTORY.*\} macros, \{HOST.HOST\}, \{HOST.NAME\}, \{HOST.CONN\}, \{HOST.DNS\}, \{HOST.IP\}, \{HOST.PORT\} and \{HOST.ID\} macros are supported in tags. Note, that tags with the same name, but different values are not considered ‘duplicates’ and can be added to the same template.

The following options are available when selecting the respective button for macros update:

- **Add** - allows to specify additional user macros for the templates. If Update existing checkbox is checked, value, type and description for the specified macro name will be updated. If unchecked, if a macro with that name already exist on the template(s), it will not be updated.
- **Update** - will replace values, types and descriptions of macros specified in this list. If Add missing checkbox is checked, macro that didn’t previously exist on a template will be added as new macro. If unchecked, only macros that already exist on a template will be updated.
- **Remove** - will remove specified macros from templates. If Except selected box is checked, all macros except specified in the list will be removed. If unchecked, only macros specified in the list will be removed.
- **Remove all** - will remove all user macros from templates. If I confirm to remove all macros checkbox is not checked, a new popup window will open asking to confirm removal of all macros.
Buttons with the following options are available for value map update:

- **Add** - add value maps to the templates. If you mark Update existing, all properties of the value map with this name will be updated. Otherwise, if a value map with that name already exists, it will not be updated.
- **Update** - update existing value maps. If you mark Add missing, a value map that didn’t previously exist on a template will be added as a new value map. Otherwise only the value maps that already exist on a template will be updated.
- **Rename** - give new name to an existing value map
- **Remove** - remove the specified value maps from the templates. If you mark Except selected, all value maps will be removed except the ones that are specified.
- **Remove all** - remove all value maps from the templates. If the I confirm to remove all value maps checkbox is not marked, a new popup window will open asking to confirm the removal.

When done with all required changes, click on Update. The attributes will be updated accordingly for all the selected templates.

1 Hosts and host groups

What is a "host"?

Typical Zabbix hosts are the devices you wish to monitor (servers, workstations, switches, etc).

Creating hosts is one of the first monitoring tasks in Zabbix. For example, if you want to monitor some parameters on a server "x", you must first create a host called, say, "Server X" and then you can look to add monitoring items to it.

Hosts are organized into host groups.

Proceed to creating and configuring a host.

1 Configuring a host

Overview

To configure a host in Zabbix frontend, do the following:

- Go to: Configuration → Hosts or Monitoring → Hosts
- Click on Create host to the right (or on the host name to edit an existing host)
- Enter parameters of the host in the form

You can also use the Clone and Full clone buttons in the form of an existing host to create a new host. Clicking on Clone will retain all host parameters and template linkage (keeping all entities from those templates). Full clone will additionally retain directly attached entities (applications, items, triggers, graphs, low-level discovery rules and web scenarios).

Note: When a host is cloned, it will retain all template entities as they are originally on the template. Any changes to those entities made on the existing host level (such as changed item interval, modified regular expression or added prototypes to the low-level discovery rule) will not be cloned to the new host; instead they will be as on the template.

Configuration

The **Host** tab contains general host attributes:
All mandatory input fields are marked with a red asterisk.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host name</td>
<td>Enter a unique host name. Alphanumerics, spaces, dots, dashes and underscores are allowed. However, leading and trailing spaces are disallowed. Note: With Zabbix agent running on the host you are configuring, the agent configuration file parameter Hostname must have the same value as the host name entered here. The name in the parameter is needed in the processing of active checks.</td>
</tr>
<tr>
<td>Visible name</td>
<td>Enter a unique visible name for the host. If you set this name, it will be the one visible in lists, maps, etc instead of the technical host name. This attribute has UTF-8 support.</td>
</tr>
</tbody>
</table>
Parameter Description

Templates Link templates to the host. All entities (items, triggers, graphs, etc) will be inherited from the template.
To link a new template, start typing the template name in the Link new templates field. A list of matching templates will appear; scroll down to select. Alternatively, you may click on Select next to the field and select templates from the list in a popup window. The templates that are selected in the Link new templates field will be linked to the host when the host configuration form is saved or updated.
To unlink a template, use one of the two options in the Linked templates block:
Unlink - unlink the template, but preserve its items, triggers and graphs
Unlink and clear - unlink the template and remove all its items, triggers and graphs
Listed template names are clickable links leading to the template configuration form.

Host groups
Select host groups the host belongs to. A host must belong to at least one host group. A new group can be created and linked to the host group by adding a non-existing group name.

Interfaces Several host interface types are supported for a host: Agent, SNMP, JMX and IPMI.
No interfaces are defined by default. To add a new interface, click on Add in the Interfaces block, select the interface type and enter IP/DNS, Connect to and Port info.
Note: Interfaces that are used in any items cannot be removed and link Remove is grayed out for them.
See Configuring SNMP monitoring for additional details on configuring an SNMP interface (v1, v2 and v3).

IP address Host IP address (optional).
DNS name Host DNS name (optional).
Connect to Clicking the respective button will tell Zabbix server what to use to retrieve data from agents:
IP - Connect to the host IP address (recommended)
DNS - Connect to the host DNS name
Port TCP/UDP port number. Default values are: 10050 for Zabbix agent, 161 for SNMP agent, 12345 for JMX and 623 for IPMI.
Default Check the radio button to set the default interface.
Description Enter the host description.
Monitored by The host can be monitored either by Zabbix server or one of Zabbix proxies:
(no proxy) - host is monitored by Zabbix server
Proxy name - host is monitored by Zabbix proxy "Proxy name"
Enabled Mark the checkbox to make the host active, ready to be monitored. If unchecked, the host is not active, thus not monitored.

The IPMI tab contains IPMI management attributes.

Parameter Description
Authentication algorithm Select the authentication algorithm.
Privilege level Select the privilege level.
Username User name for authentication. User macros may be used.
Password Password for authentication. User macros may be used.

The Tags tab allows you to define host-level tags. All problems of this host will be tagged with the values entered here.

User macros, {INVENTORY.*} macros, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros are supported in tags.

The Macros tab allows you to define host-level user macros as a name-value pairs. Note that macro values can be kept as plain text, secret text or Vault secret. Adding a description is also supported.
You may also view here template-level and global user macros if you select the Inherited and host macros option. That is where all defined user macros for the host are displayed with the value they resolve to as well as their origin.

For convenience, links to respective templates and global macro configuration are provided. It is also possible to edit a template/global macro on the host level, effectively creating a copy of the macro on the host.

The Host inventory tab allows you to manually enter inventory information for the host. You can also select to enable Automatic inventory population, or disable inventory population for this host.

If inventory is enabled (manual or automatic), a green dot is displayed with the tab name.

Encryption

The Encryption tab allows you to require encrypted connections with the host.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connections to host</td>
<td>How Zabbix server or proxy connects to Zabbix agent on a host: no encryption (default), using PSK (pre-shared key) or certificate.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Connections from host</td>
<td>Select what type of connections are allowed from the host (i.e. from Zabbix agent and Zabbix sender). Several connection types can be selected at the same time (useful for testing and switching to other connection type). Default is &quot;No encryption&quot;.</td>
</tr>
<tr>
<td>Issuer</td>
<td>Allowed issuer of certificate. Certificate is first validated with CA (certificate authority). If it is valid, signed by the CA, then the Issuer field can be used to further restrict allowed CA. This field is intended to be used if your Zabbix installation uses certificates from multiple CAs. If this field is empty then any CA is accepted.</td>
</tr>
<tr>
<td>Subject</td>
<td>Allowed subject of certificate. Certificate is first validated with CA. If it is valid, signed by the CA, then the Subject field can be used to allow only one value of Subject string. If this field is empty then any valid certificate signed by the configured CA is accepted.</td>
</tr>
<tr>
<td>PSK identity</td>
<td>Pre-shared key identity string. Do not put sensitive information in the PSK identity, it is transmitted unencrypted over the network to inform a receiver which PSK to use.</td>
</tr>
<tr>
<td>PSK</td>
<td>Pre-shared key (hex-string). Maximum length: 512 hex-digits (256-byte PSK) if Zabbix uses GnuTLS or OpenSSL library, 64 hex-digits (32-byte PSK) if Zabbix uses mbed TLS (PolarSSL) library. Example: 1f87b595725ac58dd977beeef14b97461a7c1045b9a1c963065002c5473194952</td>
</tr>
</tbody>
</table>

**Value mapping**

The **Value mapping** tab allows to configure human-friendly representation of item data in value mappings.

**Creating a host group**

Only Super Admin users can create host groups.

To create a host group in Zabbix frontend, do the following:

- Go to: Configuration → Host groups
- Click on Create host group in the upper right corner of the screen
- Enter the group name in the form

![New host group](image)

To create a nested host group, use the `/` forward slash separator, for example Europe/Latvia/Riga/Zabbix servers. You can create this group even if none of the three parent host groups (Europe/Latvia/Riga/) exist. In this case creating these parent host groups is up to the user; they will not be created automatically. Leading and trailing slashes, several slashes in a row are not allowed. Escaping of `/` is not supported.

Once the group is created, you can click on the group name in the list to edit group name, clone the group or set additional option:

![Host group](image)

Apply permissions and tag filters to all subgroups - mark this checkbox and click on Update to apply the same level of permissions/tag filters to all nested host groups. For user groups that may have had differing permissions assigned to nested host groups, the permission level of the parent host group will be enforced on the nested groups. This is a one-time option that is not saved in the database.
Permissions to nested host groups

- When creating a child host group to an existing parent host group, user group permissions to the child are inherited from the parent (for example, when creating Riga/Zabbix servers if Riga already exists).
- When creating a parent host group to an existing child host group, no permissions to the parent are set (for example, when creating Riga if Riga/Zabbix servers already exists).

2 Inventory

Overview

You can keep the inventory of networked devices in Zabbix. There is a special Inventory menu in the Zabbix frontend. However, you will not see any data there initially and it is not where you enter data. Building inventory data is done manually when configuring a host or automatically by using some automatic population options.

Building inventory

Manual mode

When configuring a host, in the Host inventory tab you can enter such details as the type of device, serial number, location, responsible person, etc - data that will populate inventory information.

If a URL is included in host inventory information and it starts with ‘http’ or ‘https’, it will result in a clickable link in the Inventory section.

Automatic mode

Host inventory can also be populated automatically. For that to work, when configuring a host the inventory mode in the Host inventory tab must be set to Automatic.

Then you can configure host items to populate any host inventory field with their value, indicating the destination field with the respective attribute (called Item will populate host inventory field) in item configuration.

Items that are especially useful for automated inventory data collection:

- system.hw.chassis[full|type|vendor|model|serial] - default is [full], root permissions needed
- system.hw.cpu[all|cpunum,full|maxfreq|vendor|model|curfreq] - default is [all,full]
- system.hw.devices[pci|usb] - default is [pci]
- system.hw.macaddr[interface,short|full] - default is [all,full], interface is regexp
- system.sw.arch
- system.sw.os[name|short|full] - default is [name]
- system.sw.packages[package,manager,short|full] - default is [all,all,full], package is regexp

Inventory mode selection

Inventory mode can be selected in the host configuration form.

Inventory mode by default for new hosts is selected based on the Default host inventory mode setting in Administration → General → Other.

For hosts added by network discovery or autoregistration actions, it is possible to define a Set host inventory mode operation selecting manual or automatic mode. This operation overrides the Default host inventory mode setting.

Inventory overview

The details of all existing inventory data are available in the Inventory menu.

In Inventory → Overview you can get a host count by various fields of the inventory.

In Inventory → Hosts you can see all hosts that have inventory information. Clicking on the host name will reveal the inventory details in a form.
The **Overview** tab shows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host name</td>
<td>Name of the host. Clicking on the name opens a menu with the scripts defined for the host. Host name is displayed with an orange icon, if the host is in maintenance.</td>
</tr>
<tr>
<td>Visible name</td>
<td>Visible name of the host (if defined).</td>
</tr>
<tr>
<td>Host (Agent, SNMP, JMX, IPMI)&lt;br&gt;interfaces</td>
<td>This block provides details of the interfaces configured for the host.</td>
</tr>
<tr>
<td>OS</td>
<td>Operating system inventory field of the host (if defined).</td>
</tr>
<tr>
<td>Hardware</td>
<td>Host hardware inventory field (if defined).</td>
</tr>
<tr>
<td>Software</td>
<td>Host software inventory field (if defined).</td>
</tr>
<tr>
<td>Description</td>
<td>Host description.</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Links to monitoring sections with data for this host: Web, Latest data, Problems, Graphs, Dashboards.</td>
</tr>
<tr>
<td>Configuration</td>
<td>Links to configuration sections for this host: Host, Applications, Items, Triggers, Graphs, Discovery, Web. The amount of configured entities is listed in parenthesis after each link.</td>
</tr>
</tbody>
</table>

The **Details** tab shows all inventory fields that are populated (are not empty).

**Inventory macros**

There are host inventory macros `{INVENTORY.*}` available for use in notifications, for example:

"Server in `{INVENTORY,LOCATION1}` has a problem, responsible person is `{INVENTORY,CONTACT1}`, phone number `{INVENTORY,POC,PRIMARY,PHONE,A1}`."

For more details, see the supported macro page.

**3 Mass update**

**Overview**

Sometimes you may want to change some attribute for a number of hosts at once. Instead of opening each individual host for editing, you may use the mass update function for that.

**Using mass update**

To mass-update some hosts, do the following:

- Mark the checkboxes before the hosts you want to update in the host list
- Click on Mass update below the list
Navigating to the tab with required attributes (Host, IPMI, Tags, Macros, Inventory, Encryption or Value mapping) and marking the checkboxes of any attribute to update and enter a new value for them.

The following options are available when selecting the respective button for template linkage update:

- **Link** - specify which additional templates to link
- **Replace** - specify which templates to link while unlinking any template that was linked to the hosts before
- **Unlink** - specify which templates to unlink

To specify the templates to link/unlink start typing the template name in the auto-complete field until a dropdown appears offering the matching templates. Just scroll down to select the required template.

The Clear when unlinking option will allow not only to unlink any previously linked templates, but also to remove all elements inherited from them (items, triggers, etc.).

The following options are available when selecting the respective button for host group update:

- **Add** - allows to specify additional host groups from the existing ones or enter completely new host groups for the hosts
- **Replace** - will remove the host from any existing host groups and replace them with the one(s) specified in this field (existing or new host groups)
- **Remove** - will remove specific host groups from hosts

These fields are auto-complete - starting to type in them offers a dropdown of matching host groups. If the host group is new, it also appears in the dropdown and is indicated by (new) after the string. Just scroll down to select.
User macros, \{INVENTORY.*\} macros, \{HOST.HOST\}, \{HOST.NAME\}, \{HOST.CONN\}, \{HOST.DNS\}, \{HOST.IP\}, \{HOST.PORT\} and \{HOST.ID\} macros are supported in tags. Note, that tags with the same name, but different values are not considered ‘duplicates’ and can be added to the same host.

The following options are available when selecting the respective button for macros update:

- **Add** - allows to specify additional user macros for the hosts. If Update existing checkbox is checked, value, type and description for the specified macro name will be updated. If unchecked, if a macro with that name already exist on the host(s), it will not be updated.
- **Update** - will replace values, types and descriptions of macros specified in this list. If Add missing checkbox is checked, macro that didn’t previously exist on a host will be added as new macro. If unchecked, only macros that already exist on a host will be updated.
- **Remove** - will remove specified macros from hosts. If Except selected box is checked, all macros except specified in the list
will be removed. If unchecked, only macros specified in the list will be removed.

- Remove all - will remove all user macros from hosts. If I confirm to remove all macros checkbox is not checked, a new popup window will open asking to confirm removal of all macros.

To be able to mass update inventory fields, the Inventory mode should be set to ‘Manual’ or ‘Automatic’.

Buttons with the following options are available for value map update:

- Add - add value maps to the hosts. If you mark Update existing, all properties of the value map with this name will be
updated. Otherwise, if a value map with that name already exists, it will not be updated.

- Update - update existing value maps. If you mark Add missing, a value map that didn’t previously exist on a host will be added as a new value map. Otherwise only the value maps that already exist on a host will be updated.
- Rename - give new name to an existing value map
- Remove - remove the specified value maps from the hosts. If you mark Except selected, all value maps will be removed except the ones that are specified.
- Remove all - remove all value maps from the hosts. If the I confirm to remove all value maps checkbox is not marked, a new popup window will open asking to confirm the removal.

When done with all required changes, click on Update. The attributes will be updated accordingly for all the selected hosts.

2 Items

Overview

Items are the ones that gather data from a host.

Once you have configured a host, you need to add some monitoring items to start getting actual data.

An item is an individual metric. One way of quickly adding many items is to attach one of the predefined templates to a host. For optimized system performance though, you may need to fine-tune the templates to have only as many items and as frequent monitoring as is really necessary.

In an individual item you specify what sort of data will be gathered from the host.

For that purpose you use the item key. Thus an item with the key name system.cpu.load will gather data of the processor load, while an item with the key name net.if.in will gather incoming traffic information.

To specify further parameters with the key, you include those in square brackets after the key name. Thus, system.cpu.load[avg5] will return processor load average for the last 5 minutes, while net.if.in[eth0] will show incoming traffic in the interface eth0.

For all supported item types and item keys, see individual sections of item types.

Proceed to creating and configuring an item.

1 Creating an item

Overview

To create an item in Zabbix frontend, do the following:

- Go to: Configuration → Hosts
- Click on Items in the row of the host
- Click on Create item in the upper right corner of the screen
- Enter parameters of the item in the form

You can also create an item by opening an existing one, pressing the Clone button and then saving under a different name.

Configuration

The Item tab contains general item attributes.
All mandatory input fields are marked with a red asterisk.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Item name.</td>
</tr>
<tr>
<td>Type</td>
<td>Item type. See individual item type sections.</td>
</tr>
<tr>
<td>Key</td>
<td>Item key (up to 2048 characters). The supported item keys can be found in individual item type sections. The key must be unique within a single host. If key type is 'Zabbix agent', 'Zabbix agent (active)' or 'Simple check', the key value must be supported by Zabbix agent or Zabbix server. See also: the correct key format.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Type of information</td>
<td>Type of data as stored in the database after performing conversions, if any. <strong>Numeric (unsigned)</strong> - 64bit unsigned integer&lt;br&gt;<strong>Numeric (float)</strong> - 64bit floating point number&lt;br&gt;This type will allow precision of approximately 15 digits and range from approximately -1.79E+308 to 1.79E+308 (with exception of PostgreSQL 11 and earlier versions).&lt;br&gt;Receiving values in scientific notation is also supported. E.g. 1.23E+7, 1e308, 1.1E-4. <strong>Character</strong> - short text data&lt;br&gt;<strong>Log</strong> - long text data with optional log related properties (timestamp, source, severity, logeventid)&lt;br&gt;<strong>Text</strong> - long text data. See also text data limits.&lt;br&gt;For item keys that return data only in one specific format, matching type of information is selected automatically.</td>
</tr>
<tr>
<td>Host interface</td>
<td>Select the host interface. This field is available when editing an item on the host level.</td>
</tr>
<tr>
<td>Units</td>
<td>If a unit symbol is set, Zabbix will add post processing to the received value and display it with the set unit postfix. By default, if the raw value exceeds 1000, it is divided by 1000 and displayed accordingly. For example, if you set bps and receive a value of 881764, it will be displayed as 881.76 Kbps. The JEDEC memory standard is used for processing <strong>B</strong> (byte), <strong>Bps</strong> (bytes per second) units, which are divided by 1024. Thus, if units are set to <strong>B</strong> or <strong>Bps</strong> Zabbix will display: 1 as 1B/1Bps&lt;br&gt;1024 as 1KB/1KBps&lt;br&gt;1536 as 1.5KB/1.5KBps&lt;br&gt;Special processing is used if the following time-related units are used: <strong>unixtime</strong> - translated to “yyyy:mma.dd hh:mm:ss”. To translate correctly, the received value must be a Numeric (unsigned) type of information.&lt;br&gt;<strong>uptime</strong> - translated to “hh:mm:ss” or “N days, hh:mm:ss”&lt;br&gt;For example, if you receive the value as 881764 (seconds), it will be displayed as “10 days, 04:56:04”&lt;br&gt;s - translated to “yyymmmddd hhmmm sss ms”; parameter is treated as number of seconds. For example, if you receive the value as 881764 (seconds), it will be displayed as “10d 4h 56m”&lt;br&gt;Only 3 upper major units are shown, like “1m 15d 5h” or “2h 4m 46s”. If there are no days to display, only two levels are displayed - “1m 5h” (no minutes, seconds or milliseconds are shown).&lt;br&gt;Will be translated to “&lt; 1 ms” if the value is less than 0.001.&lt;br&gt;Note that if a unit is prefixed with !, then no unit prefixes/processing is applied to item values. See unit conversion.</td>
</tr>
<tr>
<td>Update interval</td>
<td>Retrieve a new value for this item every N seconds. Maximum allowed update interval is 86400 seconds (1 day). <strong>Time suffixes</strong> are supported, e.g. 30s, 1m, 2h, 1d.&lt;br&gt;<strong>User macros</strong> are supported.&lt;br&gt;A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported.&lt;br&gt;Note: The update interval can only be set to ‘0’ if custom intervals exist with a non-zero value. If set to ‘0’, and a custom interval (flexible or scheduled) exists with a non-zero value, the item will be polled during the custom interval duration.&lt;br&gt;Note that the first item poll after the item became active or after update interval change might occur earlier than the configured value.&lt;br&gt;New items will be checked within 60 seconds of their creation, unless they have Scheduling or Flexible update interval and the Update interval is set to 0.&lt;br&gt;An existing passive item can be polled for value immediately by pushing the Execute now button.</td>
</tr>
<tr>
<td>Custom intervals</td>
<td>You can create custom rules for checking the item: <strong>Flexible</strong> - create an exception to the Update interval (interval with different frequency)&lt;br&gt;<strong>Scheduling</strong> - create a custom polling schedule.&lt;br&gt;For detailed information see Custom intervals.&lt;br&gt;<strong>Time suffixes</strong> are supported in the Interval field, e.g. 30s, 1m, 2h, 1d.&lt;br&gt;<strong>User macros</strong> are supported.&lt;br&gt;A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported.&lt;br&gt;Scheduling is supported since Zabbix 3.0.0.&lt;br&gt;Note: Not available for Zabbix agent active items.</td>
</tr>
</tbody>
</table>
### Parameter: Description

#### History storage period
Select either:
- **Do not keep history** - item history is not stored. Useful for master items if only dependent items need to keep history.

This setting cannot be overridden by global housekeeper settings.

- **Storage period** - specify the duration of keeping detailed history in the database (1 hour to 25 years). Older data will be removed by the housekeeper. Stored in seconds.

**Time suffixes** are supported, e.g. 2h, 1d. **User macros** are supported.

The Storage period value can be overridden globally in Administration → General → Housekeeper.

If a global overriding setting exists, a green info icon is displayed. If you position your mouse on it, a warning message is displayed, e.g. Overridden by global housekeeper settings (1d).

It is recommended to keep the recorded values for the smallest possible time to reduce the size of value history in the database. Instead of keeping a long history of values, you can keep longer data of trends.

See also History and trends.

#### Trend storage period
Select either:
- **Do not keep trends** - trends are not stored.

This setting cannot be overridden by global housekeeper settings.

- **Storage period** - specify the duration of keeping aggregated (hourly min, max, avg, count) history in the database (1 day to 25 years). Older data will be removed by the housekeeper. Stored in seconds.

**Time suffixes** are supported, e.g. 24h, 1d. **User macros** are supported.

The Storage period value can be overridden globally in Administration → General → Housekeeper.

If a global overriding setting exists, a green info icon is displayed. If you position your mouse on it, a warning message is displayed, e.g. Overridden by global housekeeper settings (7d).

Note: Keeping trends is not available for non-numeric data - character, log and text.

See also History and trends.

#### Value mapping
Apply value mapping to this item. **Value mapping** does not change received values, it is for displaying data only.

It works with Numeric(unsigned), Numeric(float) and Character items.

For example, "Windows service states".

#### Log time format
Available for items of type **Log** only. Supported placeholders:

- **y**: Year (1970-2038)
- **M**: Month (01-12)
- **d**: Day (01-31)
- **h**: Hour (00-23)
- **m**: Minute (00-59)
- **s**: Second (00-59)

If left blank the timestamp will not be parsed.

For example, consider the following line from the Zabbix agent log file:

```
```

It begins with six character positions for PID, followed by date, time, and the rest of the line.

Log time format for this line would be "pppppp:yyyyMMdd:hhmmss".

Note that "p" and ":" chars are just placeholders and can be anything but "yMdhms".

#### Populates host inventory field
You can select a host inventory field that the value of item will populate. This will work if automatic inventory population is enabled for the host.

This field is not available if Type of information is set to ‘Log’.

#### Description
Enter an item description.

#### Enabled
Mark the checkbox to enable the item so it will be processed.

#### Latest data
Click on the link to view the latest data for the item.

This link is only available when editing an already existing item.

---

Item type specific fields are described on corresponding pages.

When editing an existing template level item on a host level, a number of fields are read-only. You can use the link in the form header and go to the template level and edit them there, keeping in mind that the changes on a template level will change the item for all hosts that the template is linked to.

The Tags tab allows to define item-level tags.
Item value preprocessing

The **Preprocessing** tab allows to define *transformation rules* for the received values.

Testing

It is possible to test an item and, if configured correctly, get a real value in return. Testing can occur even before an item is saved.

Testing is available for host and template items, item prototypes and low-level discovery rules. Testing is not available for active items.

Item testing is available for the following passive item types:

- Zabbix agent
- SNMP agent (v1, v2, v3)
- IPMI agent
- SSH checks
- Telnet checks
- JMX agent
- Simple checks (except `icmpping*`, `vmware.*` items)
- Zabbix internal
- Calculated items
- External checks
- Database monitor
- HTTP agent
- Script

To test an item, click on the Test button at the bottom of the item configuration form. Note that the Test button will be disabled for items that cannot be tested (like active checks, excluded simple checks).

The item testing form has fields for the required host parameters (host address, port, proxy name/no proxy) and item-specific details (such as SNMPv2 community or SNMPv3 security credentials). These fields are context aware:

- The values are pre-filled when possible, i.e. for items requiring an agent, by taking the information from the selected agent interface of the host
- The values have to be filled manually for template items
- Plain-text macro values are resolved
- Fields where the value (or part of the value) is a secret or Vault macro are empty and have to be entered manually. If any item parameter contains a secret macro value, the following warning message is displayed: “Item contains user-defined macros with secret values. Values of these macros should be entered manually.”
- The fields are disabled when not needed in the context of the item type (e.g. the host address field and the proxy field are disabled for calculated items)
To test the item, click on Get value. If the value is retrieved successfully, it will fill the Value field, moving the current value (if any) to the Previous value field while also calculating the Prev. time field, i.e. the time difference between the two values (clicks) and trying to detect an EOL sequence and switch to CRLF if detecting "\n\r" in retrieved value.

If the configuration is incorrect, an error message is displayed describing the possible cause.

A successfully retrieved value from host can also be used to test preprocessing steps.

Form buttons

Buttons at the bottom of the form allow to perform several operations.

Add

Add an item. This button is only available for new items.

Update

Update the properties of an item.

Clone

Create another item based on the properties of the current item.
Execute a check for a new item value immediately. Supported for **passive** checks only (see more details).

Note that when checking for a value immediately, configuration cache is not updated, thus the value will not reflect very recent changes to item configuration.

Test if item configuration is correct by getting a value.

Delete the item history and trends.

Delete the item.

Cancel the editing of item properties.

---

**Text data limits**

Text data limits depend on the database backend. Before storing text values in the database they get truncated to match the database value type limit:

<table>
<thead>
<tr>
<th>Database</th>
<th>Type of information</th>
<th>Character</th>
<th>Log</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>Character</td>
<td>255</td>
<td>65536 bytes</td>
<td>65536 bytes</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>Character</td>
<td>255</td>
<td>65536 characters</td>
<td>65536 characters</td>
</tr>
<tr>
<td>Oracle</td>
<td>Character</td>
<td>255</td>
<td>65536 characters</td>
<td>65536 characters</td>
</tr>
</tbody>
</table>

**Unit conversion**

By default, specifying a unit for an item results in a multiplier prefix being added - for example, an incoming value ‘2048’ with unit ‘B’ would be displayed as ‘2KB’.

To prevent a unit from conversion, use the `!` prefix, for example, `!B`. To better understand how the conversion works with and without the exclamation mark, see the following examples of values and units:

- `1024 !B → 1024 B`
- `1024 B → 1 KB`
- `61 !s → 61 s`
- `61 s → 1m 1s`
- `0 !uptime → 0 uptime`
- `0 uptime → 00:00:00`
- `0 !! → 0 !`
- `0 ! → 0`

Before Zabbix 4.0, there was a hardcoded unit stoplist consisting of `ms`, `rpm`, `RPM`, `%`. This stoplist has been deprecated, thus the correct way to prevent converting such units is `!ms`, `!rpm`, `!RPM`, `!%`.

**Custom script limit**

Available custom script length depends on the database used:

<table>
<thead>
<tr>
<th>Database</th>
<th>Limit in characters</th>
<th>Limit in bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>65535</td>
<td>65535</td>
</tr>
<tr>
<td>Oracle Database</td>
<td>2048</td>
<td>4000</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>65535</td>
<td>not limited</td>
</tr>
<tr>
<td>SQLite (only Zabbix proxy)</td>
<td>65535</td>
<td>not limited</td>
</tr>
</tbody>
</table>

**Unsupported items**

An item can become unsupported if its value cannot be retrieved for some reason. Such items are still rechecked at their standard **Update interval**.

Unsupported items are reported as having a NOT SUPPORTED state.
1 Item key format

Item key format, including key parameters, must follow syntax rules. The following illustrations depict the supported syntax. Allowed elements and characters at each point can be determined by following the arrows - if some block can be reached through the line, it is allowed, if not - it is not allowed.

To construct a valid item key, one starts with specifying the key name, then there's a choice to either have parameters or not - as depicted by the two lines that could be followed.

Key name
The key name itself has a limited range of allowed characters, which just follow each other. Allowed characters are: 0-9a-zA-Z_-.
Which means:
• all numbers;
• all lowercase letters;
• all uppercase letters;
• underscore;
• dash;
• dot.

Key parameters
An item key can have multiple parameters that are comma separated.

Each key parameter can be either a quoted string, an unquoted string or an array.
The parameter can also be left empty, thus using the default value. In that case, the appropriate number of commas must be added if any further parameters are specified. For example, item key `icmping[,,200,,500]` would specify that the interval between individual pings is 200 milliseconds, timeout - 500 milliseconds, and all other parameters are left at their defaults.

Parameter - quoted string

If the key parameter is a quoted string, any Unicode character is allowed.

If the key parameter string contains comma, this parameter has to be quoted.

If the key parameter string contains quotation mark, this parameter has to be quoted and each quotation mark which is a part of the parameter string has to be escaped with a backslash (\) character.

To quote item key parameters, use double quotes only. Single quotes are not supported.

Parameter - unquoted string

If the key parameter is an unquoted string, any Unicode character is allowed except comma and right square bracket (]). Unquoted parameter cannot start with left square bracket ([).

Parameter - array

If the key parameter is an array, it is again enclosed in square brackets, where individual parameters come in line with the rules and syntax of specifying multiple parameters.
Multi-level parameter arrays, e.g. \([a, [b, [c,d]], e]\), are not allowed.

2 Custom intervals

Overview

It is possible to create custom rules regarding the times when an item is checked. The two methods for that are Flexible intervals, which allow to redefine the default update interval, and Scheduling, whereby an item check can be executed at a specific time or sequence of times.

Flexible intervals

Flexible intervals allow to redefine the default update interval for specific time periods. A flexible interval is defined with Interval and Period where:

- Interval – the update interval for the specified time period
- Period – the time period when the flexible interval is active (see the time periods for detailed description of the Period format)

Up to seven flexible intervals can be defined. If multiple flexible intervals overlap, the smallest Interval value is used for the overlapping period. Note that if the smallest value of overlapping flexible intervals is ‘0’, no polling will take place. Outside the flexible intervals the default update interval is used.

Note that if the flexible interval equals the length of the period, the item will be checked exactly once. If the flexible interval is greater than the period, the item might be checked once or it might not be checked at all (thus such configuration is not advisable). If the flexible interval is less than the period, the item will be checked at least once.

If the flexible interval is set to ‘0’, the item is not polled during the flexible interval period and resumes polling according to the default Update interval once the period is over. Examples:

<table>
<thead>
<tr>
<th>Interval</th>
<th>Period</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1-5,09:00-18:00</td>
<td>Item will be checked every 10 seconds during working hours.</td>
</tr>
<tr>
<td>0</td>
<td>1-7,00:00-7:00</td>
<td>Item will not be checked during the night.</td>
</tr>
<tr>
<td>0</td>
<td>7-7,00:00-24:00</td>
<td>Item will not be checked on Sundays.</td>
</tr>
<tr>
<td>60</td>
<td>1-7,12:00-12:01</td>
<td>Item will be checked at 12:00 every day. Note that this was used as a workaround for scheduled checks and starting with Zabbix 3.0 it is recommended to use scheduling intervals for such checks.</td>
</tr>
</tbody>
</table>

Scheduling intervals

Scheduling intervals are used to check items at specific times. While flexible intervals are designed to redefine the default item update interval, the scheduling intervals are used to specify an independent checking schedule, which is executed in parallel.

A scheduling interval is defined as: \(md<\text{filter}>wd<\text{filter}>h<\text{filter}>m<\text{filter}>s<\text{filter}>\) where:

- \(md\) - month days
- \(wd\) - week days
- \(h\) - hours
- \(m\) - minutes
- \(s\) - seconds

\(<\text{filter}>\) is used to specify values for its prefix (days, hours, minutes, seconds) and is defined as: \([<\text{from}>[-<\text{to}>]]/[<\text{step}>][,<\text{filter}>]\) where:

- \(<\text{from}>\) and \(<\text{to}>\) define the range of matching values (included). If \(<\text{to}>\) is omitted then the filter matches a \(<\text{from}>\) – \(<\text{from}>\) range. If \(<\text{from}>\) is also omitted then the filter matches all possible values.
- \(<\text{step}>\) defines the skips of the number value through the range. By default \(<\text{step}>\) has the value of 1, which means that all values of the defined range are matched.

While the filter definitions are optional, at least one filter must be used. A filter must either have a range or the \(<\text{step}>\) value defined.
An empty filter matches either ‘0’ if no lower-level filter is defined or all possible values otherwise. For example, if the hour filter is omitted then only ‘0’ hour will match, provided minute and seconds filters are omitted too, otherwise an empty hour filter will match all hour values.

Valid `<from>` and `<to>` values for their respective filter prefix are:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Description</th>
<th><code>&lt;from&gt;</code></th>
<th><code>&lt;to&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>md</td>
<td>Month days</td>
<td>1-31</td>
<td>1-31</td>
</tr>
<tr>
<td>wd</td>
<td>Week days</td>
<td>1-7</td>
<td>1-7</td>
</tr>
<tr>
<td>h</td>
<td>Hours</td>
<td>0-23</td>
<td>0-23</td>
</tr>
<tr>
<td>m</td>
<td>Minutes</td>
<td>0-59</td>
<td>0-59</td>
</tr>
<tr>
<td>s</td>
<td>Seconds</td>
<td>0-59</td>
<td>0-59</td>
</tr>
</tbody>
</table>

The `<from>` value must be less or equal to `<to>` value. The `<step>` value must be greater or equal to 1 and less or equal to `<to>` - `<from>`.

Single digit month days, hours, minutes and seconds values can be prefixed with 0. For example `md01-31` and `h/02` are valid intervals, but `md01-031` and `wd01-07` are not.

In Zabbix frontend, multiple scheduling intervals are entered in separate rows. In Zabbix API, they are concatenated into a single string with a semicolon ; as a separator.

If a time is matched by several intervals it is executed only once. For example, `wd1h9;h9` will be executed only once on Monday at 9am.

Examples:

<table>
<thead>
<tr>
<th>Interval</th>
<th>Will be executed</th>
</tr>
</thead>
<tbody>
<tr>
<td>m0-59</td>
<td>every minute</td>
</tr>
<tr>
<td>h9-17/2</td>
<td>every 2 hours starting with 9:00 (9:00, 11:00 ...)</td>
</tr>
<tr>
<td>m0,30 or m/30</td>
<td>hourly at hh:00 and hh:30</td>
</tr>
<tr>
<td>m0,5,10,15,20,25,30,35,40,45,50,55 or m/5</td>
<td>every five minutes</td>
</tr>
<tr>
<td>wd1-5h9</td>
<td>every Monday till Friday at 9:00</td>
</tr>
<tr>
<td>wd1-5h9-18</td>
<td>every Monday till Friday at 9:00, 10:00, ..., 18:00</td>
</tr>
<tr>
<td>h9,10,11 or h9-11</td>
<td>every day at 9:00, 10:00 and 11:00</td>
</tr>
<tr>
<td>md1h9m30</td>
<td>every 1st day of each month at 9:30</td>
</tr>
<tr>
<td>md1wd1h9m30</td>
<td>every 1st day of each month at 9:30 if it is Monday</td>
</tr>
<tr>
<td>h9m/30</td>
<td>every day at 9:00, 9:30</td>
</tr>
<tr>
<td>h9m0-59/30</td>
<td>every day at 9:00, 9:30</td>
</tr>
<tr>
<td>h9,10m30</td>
<td>every day at 9:00, 9:30, 10:00, 10:30</td>
</tr>
<tr>
<td>h9-10m30</td>
<td>every day at 9:30, 10:30</td>
</tr>
<tr>
<td>h9m10-40/30</td>
<td>every day at 9:10, 9:40</td>
</tr>
<tr>
<td>h9,10m10-40/30</td>
<td>every day at 9:10, 9:40, 10:10, 10:40</td>
</tr>
<tr>
<td>h9-10m10-40/30</td>
<td>every day at 9:10, 9:40, 10:10, 10:40</td>
</tr>
<tr>
<td>h9m10-40</td>
<td>every day at 9:10, 9:11, 9:12, ..., 9:40</td>
</tr>
<tr>
<td>h9m10-40/1</td>
<td>every day at 9:10, 9:11, 9:12, ..., 9:40</td>
</tr>
<tr>
<td>h9-12,15</td>
<td>every day at 9:00, 10:00, 11:00, 12:00, 15:00</td>
</tr>
<tr>
<td>h9-12,15m0</td>
<td>every day at 9:00, 10:00, 11:00, 12:00, 15:00</td>
</tr>
<tr>
<td>h9-12,15m0s30</td>
<td>every day at 9:00:30, 10:00:30, 11:00:30, 12:00:30, 15:00:30</td>
</tr>
<tr>
<td>h9-12s30</td>
<td>every day at 9:00:30, 9:01:30, 9:02:30 ... 12:58:30, 12:59:30</td>
</tr>
<tr>
<td>h9m30;h10 (API-specific syntax)</td>
<td>every day at 9:00, 9:30, 10:00</td>
</tr>
<tr>
<td>h9m30</td>
<td>every day at 9:00, 9:30, 10:00</td>
</tr>
</tbody>
</table>

2 Item value preprocessing

Overview

Preprocessing allows to define transformation rules for the received item values. One or several transformations are possible before saving to the database.
Transformations are executed in the order in which they are defined. Preprocessing is done by Zabbix server or proxy (if items are monitored by proxy).

Note that all values passed to preprocessing are of the string type, conversion to desired value type (as defined in item configuration) is performed at the end of the preprocessing pipeline; conversions, however, may also take place if required by the corresponding preprocessing step. See preprocessing details for more technical information.

See also: Usage examples

Configuration

Preprocessing rules are defined in the Preprocessing tab of the item configuration form.

Items

An item will become unsupported if any of the preprocessing steps fails, unless custom error handling has been specified using a Custom on fail option for supported transformations.

For log items, log metadata (without value) will always reset item unsupported state and make item supported again, even if the initial error occurred after receiving a log value from agent.

User macros and user macros with context are supported in item value preprocessing parameters, including JavaScript code.

Context is ignored when a macro is replaced with its value. Macro value is inserted in the code as is, it is not possible to add additional escaping before placing the value in the JavaScript code. Please be advised, that this can cause JavaScript errors in some cases.

<table>
<thead>
<tr>
<th>Type</th>
<th>Transformation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Regular expression</td>
<td>Match the value to the &lt;pattern&gt; regular expression and replace value with &lt;output&gt;. The regular expression supports extraction of maximum 10 captured groups with the \N sequence. Failure to match the input value will make the item unsupported. Parameters: <strong>pattern</strong> - regular expression <strong>output</strong> - output formatting template. An \N (where N=1…9) escape sequence is replaced with the Nth matched group. A \0 escape sequence is replaced with the matched text. Please refer to regular expressions section for some existing examples. If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.</td>
</tr>
<tr>
<td></td>
<td>Replace</td>
<td>Find the search string and replace it with another (or nothing). All occurrences of the search string will be replaced. Parameters: <strong>search string</strong> - the string to find and replace, case-sensitive (required) <strong>replacement</strong> - the string to replace the search string with. The replacement string may also be empty effectively allowing to delete the search string when found. It is possible to use escape sequences to search for or replace line breaks, carriage return, tabs and spaces “\n \r \t \s”: backslash can be escaped as ”\“ and escape sequences can be escaped as ”\n”. Escaping of line breaks, carriage return, tabs is automatically done during low-level discovery.</td>
</tr>
<tr>
<td></td>
<td>Trim</td>
<td>Remove specified characters from the beginning and end of the value.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Right trim</td>
<td>Remove specified characters from the end of the value.</td>
<td></td>
</tr>
<tr>
<td>Left trim</td>
<td>Remove specified characters from the beginning of the value.</td>
<td></td>
</tr>
<tr>
<td>Structured data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XML XPath</td>
<td>Extract value or fragment from XML data using XPath functionality.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For this option to work, Zabbix server must be compiled with libxml support.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Examples:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>number(/document/item/value) will extract 10 from</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;document&gt;&lt;item&gt;&lt;value&gt;10&lt;/value&gt;&lt;/item&gt;&lt;/document&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>number(/document/item/@attribute) will extract 10 from</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;document&gt;&lt;item attribute=&quot;10&quot;&gt;&lt;/item&gt;&lt;/document&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/document/item will extract &lt;item&gt;&lt;value&gt;10&lt;/value&gt;&lt;/item&gt; from &lt;document&gt;&lt;item&gt;&lt;value&gt;10&lt;/value&gt;&lt;/item&gt;&lt;/document&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note that namespaces are not supported.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>preprocessing step and it is possible to specify custom error-handling options: either to discard the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>value, set a specified value or set a specified error message.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSON Path</td>
<td>Extract value or fragment from JSON data using JSONPath functionality.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>preprocessing step and it is possible to specify custom error-handling options: either to discard the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>value, set a specified value or set a specified error message.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSV to JSON</td>
<td>Convert CSV file data into JSON format.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information, see: CSV to JSON preprocessing.</td>
<td></td>
</tr>
<tr>
<td>XML to JSON</td>
<td>Convert data in XML format to JSON.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information, see: Serialization rules.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>preprocessing step and it is possible to specify custom error-handling options: either to discard the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>value, set a specified value or set a specified error message.</td>
<td></td>
</tr>
<tr>
<td>Arithmetic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Custom multiplier</td>
<td>Multiply the value by the specified integer or floating-point value.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use this option to convert values received in KB, MBps, etc into B, Bps. Otherwise Zabbix cannot</td>
<td></td>
</tr>
<tr>
<td></td>
<td>correctly set prefixes (K, M, G etc).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note that if the item type of information is Numeric (unsigned), incoming values with a fractional</td>
<td></td>
</tr>
<tr>
<td></td>
<td>part will be trimmed (i.e. ‘0.9’ will become ‘0’) before the custom multiplier is applied.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Supported: scientific notation, for example, 1e+70 (since version 2.2); user macros and LLD macros</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(since version 4.0); strings that include macros, for example, {#MACRO}e+10,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$MACRO1}e+{$MACRO2} (since version 5.2.3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The macros must resolve to an integer or a floating-point number.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>preprocessing step and it is possible to specify custom error handling options: either to discard the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>value, set a specified value, or set a specified error message.</td>
<td></td>
</tr>
<tr>
<td>Change</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple change</td>
<td>Calculate the difference between the current and previous value.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Evaluated as value-prev_value, where</td>
<td></td>
</tr>
<tr>
<td></td>
<td>value - current value; prev_value - previously received value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This setting can be useful to measure a constantly growing value. If the current value is smaller</td>
<td></td>
</tr>
<tr>
<td></td>
<td>than the previous value, Zabbix discards that difference (stores nothing) and waits for another value.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Only one change operation per item is allowed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>preprocessing step and it is possible to specify custom error handling options: either to discard the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>value, set a specified value, or set a specified error message.</td>
<td></td>
</tr>
</tbody>
</table>
### Type

**Change per second**

Calculate the value change (difference between the current and previous value) speed per second.

Evaluated as \((\text{value} - \text{prev\_value})/(\text{time} - \text{prev\_time})\), where

- \(\text{value}\) - current value;
- \(\text{prev\_value}\) - previously received value;
- \(\text{time}\) - current timestamp;
- \(\text{prev\_time}\) - timestamp of previous value.

This setting is extremely useful to get speed per second for a constantly growing value. If the current value is smaller than the previous value, Zabbix discards that difference (stores nothing) and waits for another value. This helps to work correctly with, for instance, a wrapping (overflow) of 32-bit SNMP counters.

Note: As this calculation may produce floating-point numbers, it is recommended to set the ‘Type of information’ to Numeric (float), even if the incoming raw values are integers. This is especially relevant for small numbers where the decimal part matters. If the floating-point values are large and may exceed the ‘float’ field length in which case the entire value may be lost, it is actually suggested to use Numeric (unsigned) and thus trim only the decimal part.

Only one change operation per item is allowed.

If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.

### Numeral systems

**Boolean to decimal**

Convert the value from boolean format to decimal. The textual representation is translated into either 0 or 1. Thus, ‘TRUE’ is stored as 1 and ‘FALSE’ is stored as 0. All values are matched in a case-insensitive way. Currently recognized values are, for:

- TRUE - true, t, yes, y, on, up, running, enabled, available, ok, master
- FALSE - false, f, no, n, off, down, unused, disabled, unavailable, err, slave

Additionally, any non-zero numeric value is considered to be TRUE and zero is considered to be FALSE.

If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.

**Octal to decimal**

Convert the value from octal format to decimal.

If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.

**Hexadecimal to decimal**

Convert the value from hexadecimal format to decimal.

If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.

### Custom scripts

**JavaScript**

Enter JavaScript code in the block that appears when clicking in the parameter field or on a pencil icon.

Note that available JavaScript length depends on the database used.

For more information, see: [Javascript preprocessing](#).

### Validation

**In range**

Define a range that a value should be in by specifying minimum/maximum values (inclusive).

Numeric values are accepted (including any number of digits, optional decimal part and optional exponential part, negative values). User macros and low-level discovery macros can be used. The minimum value should be less than the maximum.

At least one value must exist.

If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.

**Matches regular expression**

Specify a regular expression that a value must match.

If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.

**Does not match regular expression**

Specify a regular expression that a value must not match.

If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check for error in JSON</td>
<td>Check for an application-level error message located at JSON path. Stop processing if succeeded and the message is not empty; otherwise, continue processing with the value that was before this preprocessing step. Note that these external service errors are reported to the user as is, without adding preprocessing step information. No error will be reported in case of failing to parse invalid JSON. If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.</td>
<td></td>
</tr>
<tr>
<td>Check for error in XML</td>
<td>Check for an application-level error message located at XPath. Stop processing if succeeded and the message is not empty; otherwise, continue processing with the value that was before this preprocessing step. Note that these external service errors are reported to the user as is, without adding preprocessing step information. No error will be reported in case of failing to parse invalid XML. If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.</td>
<td></td>
</tr>
<tr>
<td>Check for error using a regular expression</td>
<td>Check for an application-level error message using a regular expression. Stop processing if succeeded and the message is not empty; otherwise, continue processing with the value that was before this preprocessing step. Note that these external service errors are reported to the user as is, without adding preprocessing step information. Parameters: pattern - regular expression output - output formatting template. An \N (where N=1…9) escape sequence is replaced with the Nth matched group. A \0 escape sequence is replaced with the matched text.</td>
<td></td>
</tr>
<tr>
<td>Check for not supported value</td>
<td>Check if there was an error in retrieving item value. Normally that would lead to the item turning unsupported, but you may modify that behavior by specifying the Custom on fail error-handling options: to discard the value, to set a specified value (in this case the item will stay supported and the value can be used in triggers) or set a specified error message. Note that for this preprocessing step, the Custom on fail checkbox is grayed out and always marked. This step is always executed as the first preprocessing step and is placed above all others after saving changes to the item. It can be used only once.</td>
<td></td>
</tr>
</tbody>
</table>

**Throttling**

| Discard unchanged                                                   | Discard a value if it has not changed. If a value is discarded, it is not saved in the database and Zabbix server has no knowledge that this value was received. No trigger expressions will be evaluated, as a result, no problems for related triggers will be created/resolved. Functions will work only based on data that is actually saved in the database. As trends are built based on data in the database, if there is no value saved for an hour then there will also be no trends data for that hour. Only one throttling option can be specified for an item. Note that it is possible for items monitored by Zabbix proxy that very small value differences (less than 0.000001) are correctly not discarded by proxy, but are stored in the history as the same value if the Zabbix server database has not been upgraded. |
| Discard unchanged with heartbeat                                   | Discard a value if it has not changed within the defined time period (in seconds). Positive integer values are supported to specify the seconds (minimum -1 second). Time suffixes can be used in this field (e.g. 30s, 1m, 2h, 1d). User macros and low-level discovery macros can be used in this field. If a value is discarded, it is not saved in the database and Zabbix server has no knowledge that this value was received. No trigger expressions will be evaluated, as a result, no problems for related triggers will be created/resolved. Functions will work only based on data that is actually saved in the database. As trends are built based on data in the database, if there is no value saved for an hour then there will also be no trends data for that hour. Only one throttling option can be specified for an item. Note that it is possible for items monitored by Zabbix proxy that very small value differences (less than 0.000001) are correctly not discarded by proxy, but are stored in the history as the same value if the Zabbix server database has not been upgraded. |
For change and throttling preprocessing steps Zabbix has to remember the last value to calculate/compare the new value as required. These previous values are handled by the preprocessing manager. If Zabbix server or proxy is restarted or there is any change made to preprocessing steps the last value of the corresponding item is reset, resulting in:

- for Simple change, Change per second steps - the next value will be ignored because there is no previous value to calculated change from;
- for Discard unchanged. Discard unchanged with heartbeat steps - the next value will never be discarded, even if it should have been because of discarding rules.

Item’s Type of information parameter is displayed at the bottom of the tab when at least one preprocessing step is defined. If required, it is possible to change the type of information without leaving the Preprocessing tab. See Creating an item for the detailed parameter description.

If you use a custom multiplier or store value as Change per second for items with the type of information set to Numeric (unsigned) and the resulting calculated value is actually a float number, the calculated value is still accepted as a correct one by trimming the decimal part and storing the value as an integer.

Testing

Testing preprocessing steps is useful to make sure that complex preprocessing pipelines yield the results that are expected from them, without waiting for the item value to be received and preprocessed.

It is possible to test:

- against a hypothetical value
- against a real value from a host

Each preprocessing step can be tested individually as well as all steps can be tested together. When you click on the Test or Test all steps button respectively in the Actions block, a testing window is opened.

Testing hypothetical value
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get value from host</td>
<td>If you want to test a hypothetical value, leave this checkbox unmarked. See also: Testing real value.</td>
</tr>
<tr>
<td>Value</td>
<td>Enter the input value to test. Clicking in the parameter field or on the view/edit button will open a text area window for entering the value or code block.</td>
</tr>
<tr>
<td>Not supported</td>
<td>Mark this checkbox to test an unsupported value.</td>
</tr>
<tr>
<td>Time</td>
<td>Time of the input value is displayed: now (read-only).</td>
</tr>
<tr>
<td>Previous value</td>
<td>Enter a previous input value to compare to.</td>
</tr>
<tr>
<td>Previous time</td>
<td>Enter the previous input value time to compare to.</td>
</tr>
<tr>
<td>Macros</td>
<td>If any macros are used, they are listed along with their values. The values are editable for testing purposes, but the changes will only be saved within the testing context.</td>
</tr>
<tr>
<td>End of line sequence</td>
<td>Select the end of line sequence for multiline input values: LF - LF (line feed) sequence CRLF - CRLF (carriage-return line-feed) sequence.</td>
</tr>
<tr>
<td>Preprocessing steps</td>
<td>Preprocessing steps are listed; the testing result is displayed for each step after the Test button is clicked.</td>
</tr>
<tr>
<td>Result</td>
<td>The final result of testing preprocessing steps is displayed in all cases when all steps are tested together (when you click on the Test all steps button).</td>
</tr>
</tbody>
</table>

Click on Test to see the result after each preprocessing step.

Test values are stored between test sessions for either individual steps or all steps, allowing the user to change preprocessing steps or item configuration and then return to the testing window without having to re-enter information. Values are lost on a page refresh though.

The testing is done by Zabbix server. The frontend sends a corresponding request to the server and waits for the result. The request contains the input value and preprocessing steps (with expanded user macros). For Change and Throttling steps, an optional previous value and time can be specified. The server responds with results for each preprocessing step.

All technical errors or input validation errors are displayed in the error box at the top of the testing window.
Testing real value

To test preprocessing against a real value:

- Mark the Get value from host checkbox
- Enter or verify host parameters (host address, port, proxy name/no proxy) and item-specific details (such as SNMPv2 community or SNMPv3 security credentials). These fields are context-aware:
  - The values are pre-filled when possible, i.e. for items requiring an agent, by taking the information from the selected agent interface of the host
  - The values have to be filled manually for template items
  - Plain-text macro values are resolved
  - Fields where the value (or part of the value) is a secret or Vault macro are empty and have to be entered manually.
    If any item parameter contains a secret macro value, the following warning message is displayed: “Item contains user-defined macros with secret values. Values of these macros should be entered manually.”
  - The fields are disabled when not needed in the context of the item type (e.g. the host address and the proxy fields are disabled for calculated items)

- Click on Get value and test to test the preprocessing

If you have specified a value mapping in the item configuration form ('Show value' field), the item test dialog will show another line after the final result, named 'Result with value map applied'.

Parameters that are specific to getting a real value from a host:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get value from host</td>
<td>Mark this checkbox to get a real value from the host. Enter the host address. This field is automatically filled by the address of the item host interface.</td>
</tr>
<tr>
<td>Host address</td>
<td>Enter the host port. This field is automatically filled by the port of item host interface.</td>
</tr>
<tr>
<td>Port</td>
<td></td>
</tr>
<tr>
<td>Additional fields for SNMP</td>
<td>See Configuring SNMP monitoring for additional details on configuring an SNMP interface (v1, v2 and v3).</td>
</tr>
<tr>
<td>interfaces&lt;br&gt;(SNMP version, SNMP community, Context name, etc)</td>
<td>These fields are automatically filled from the item host interface.</td>
</tr>
<tr>
<td>Proxy</td>
<td>Specify the proxy if the host is monitored by a proxy. This field is automatically filled by the proxy of the host (if any).</td>
</tr>
</tbody>
</table>

For the rest of the parameters, see Testing hypothetical value above.

1 Usage examples
Overview

This section presents examples of using preprocessing steps to accomplish some practical tasks.

Filtering VMWare event log records

Using a regular expression preprocessing to filter unnecessary events of the VMWare event log.

1. On a working VMWare Hypervisor host check that the event log item `vmware.eventlog[<url>,<mode>]` is present and working properly. Note that the event log item could already be present on the hypervisor if the Template VM VMWare template has been linked during the host creation.

2. On the VMWare Hypervisor host create a dependent item of ‘Log’ type and set the event log item as its master.

In the “Preprocessing” tab of the dependent item select the “Matches regular expression” validation option and fill pattern, for example:

```
".* logged in .*" - filters all logging events in the event log
"\bUser\s+\K\S+" - filter only lines with usernames from the event log
```

If the regular expression is not matched then the dependent item becomes unsupported with a corresponding error message. To avoid this mark the “Custom on fail” checkbox and select to discard unmatched value, for example.

Another approach that allows using matching groups and output control is to select “Regular expression” option in the “Preprocessing” tab and fill parameters, for example:

```
pattern: ".* logged in.*", output: "\0" - filters all logging events in the event log
pattern "User (.+?)\(?=(?= . )\)", output: "\1" - filter only usernames from the event log
```

2 Preprocessing details

Overview

This section provides item value preprocessing details. Item value preprocessing allows to define and execute transformation rules for the received item values.

Preprocessing is managed by a preprocessing manager process, which was added in Zabbix 3.4, along with preprocessing workers that perform the preprocessing steps. All values (with or without preprocessing) from different data gatherers pass through the preprocessing manager before being added to the history cache. Socket-based IPC communication is used between data gatherers (pollers, trappers, etc) and the preprocessing process. Either Zabbix server or Zabbix proxy (for items monitored by the proxy) is performing preprocessing steps.

Item value processing

To visualize the data flow from data source to the Zabbix database, we can use the following simplified diagram:
1. Raw data

2. **Data gathering**
   Data gathering is performed by: poller, trapper, and other processes.

3. **Configuration cache**
   Configuration cache acts as a buffer between Zabbix processes and Zabbix DB (where configuration is stored).

4. **Data preprocessing**
   Data preprocessing is performed by: preprocessing manager and preprocessing worker processes.

5. **Storing data in history cache**

6. **History cache**
   History cache acts as a buffer between data source (from where data is gathered) and Zabbix DB (where data is stored).

7. **Storing data in database**

8. **Process data**
   Data processing starts with data synchronization (transferring data from history cache to Zabbix database) and proceeds with trigger processing, item configuration updates, etc.

9. **Zabbix database**
   Data processing is performed by: history syncer processes.
The diagram above shows only processes, objects and actions related to item value processing in a simplified form. The diagram does not show conditional direction changes, error handling or loops. Local data cache of preprocessing manager is not shown either because it doesn’t affect data flow directly. The aim of this diagram is to show processes involved in item value processing and the way they interact.

- Data gathering starts with raw data from a data source. At this point, data contains only ID, timestamp and value (can be multiple values as well)
- No matter what type of data gatherer is used, the idea is the same for active or passive checks, for trapper items and etc, as it only changes the data format and the communication starter (either data gatherer is waiting for a connection and data, or data gatherer initiates the communication and requests the data). Raw data is validated, item configuration is retrieved from configuration cache (data is enriched with the configuration data).
- Socket-based IPC mechanism is used to pass data from data gatherers to preprocessing manager. At this point data gatherer continue to gather data without waiting for the response from preprocessing manager.
- Data preprocessing is performed. This includes execution of preprocessing steps and dependent item processing.

Item can change its state to NOT SUPPORTED while preprocessing is performed if any of preprocessing steps fail.

- History data from local data cache of preprocessing manager is being flushed into history cache.
- At this point data flow stops until the next synchronization of history cache (when history syncer process performs data synchronization).
- Synchronization process starts with data normalization storing data in Zabbix database. Data normalization performs conversions to desired item type (type defined in item configuration), including truncation of textual data based on pre-defined sizes allowed for those types (HISTORY_STR_VALUE_LEN for string, HISTORY_TEXT_VALUE_LEN for text and HISTORY_LOG_VALUE_LEN for log values). Data is being sent to Zabbix database after normalization is done.

Item can change its state to NOT SUPPORTED if data normalization fails (for example, when textual value cannot be converted to number).

- Gathered data is being processed - triggers are checked, item configuration is updated if item becomes NOT SUPPORTED, etc.
- This is considered the end of data flow from the point of view of item value processing.

Item value preprocessing

To visualize the data preprocessing process, we can use the following simplified diagram:
The diagram above shows only processes, objects and main actions related to item value preprocessing in a simplified form. The diagram does not show conditional direction changes, error handling or loops. Only one preprocessing worker is shown on this diagram (multiple preprocessing workers can be used in real-life scenarios), only one item value is being processed and we assume that this item requires to execute at least one preprocessing step. The aim of this diagram is to show the idea behind item value preprocessing pipeline.

- Item data and item value is passed to preprocessing manager using socket-based IPC mechanism.
- Item is placed in the preprocessing queue.
- Item can be placed at the end or at the beginning of the preprocessing queue. Zabbix internal items are always placed at the beginning of preprocessing queue, while other item types are enqueued at the end.

  - At this point data flow stops until there is at least one unoccupied (that is not executing any tasks) preprocessing worker.
  - When preprocessing worker is available, preprocessing task is being sent to it.
  - After preprocessing is done (both failed and successful execution of preprocessing steps), preprocessed value is being passed back to preprocessing manager.
  - Preprocessing manager converts result to desired format (defined by item value type) and places result in preprocessing queue. If there are dependent items for current item, then dependent items are added to preprocessing queue as well. Dependent items are enqueued in preprocessing queue right after the master item, but only for master items with value set and not in NOT SUPPORTED state.

Value processing pipeline

Item value processing is executed in multiple steps (or phases) by multiple processes. This can cause:

- Dependent item can receive values, while THE master value cannot. This can be achieved by using the following use case:
  - Master item has value type UINT, (trapper item can be used), dependent item has value type TEXT.
  - No preprocessing steps are required for both master and dependent items.
- Textual value (like, “abc”) should be passed to master item.  
- As there are no preprocessing steps to execute, preprocessing manager checks if master item is not in NOT SUPPORTED state and if value is set (both are true) and enqueues dependent item with the same value as master item (as there are no preprocessing steps).  
- When both master and dependent items reach history synchronization phase, master item becomes NOT SUPPORTED, because of the value conversion error (textual data cannot be converted to unsigned integer).

As a result, dependent item receives a value, while master item changes its state to NOT SUPPORTED.

- Dependent item receives value that is not present in master item history. The use case is very similar to the previous one, except for the master item type. For example, if CHAR type is used for master item, then master item value will be truncated at the history synchronization phase, while dependent items will receive their value from the initial (not truncated) value of master item.

Preprocessing queue

Preprocessing queue is a FIFO data structure that stores values preserving the order in which values are reviewed by preprocessing manager. There are multiple exceptions to FIFO logic:

- Internal items are enqueued at the beginning of the queue  
- Dependent items are always enqueued after the master item

To visualize the logic of preprocessing queue, we can use the following diagram:

![Preprocessing queue diagram](image)

Values from the preprocessing queue are flushed from the beginning of the queue to the first unprocessed value. So, for example, preprocessing manager will flush values 1, 2 and 3, but will not flush value 5 as value 4 is not processed yet:

![Preprocessing queue diagram](image)

Only two values will be left in queue (4 and 5) after flushing, values are added into local data cache of preprocessing manager and then values are transferred from local cache into history cache. Preprocessing manager can flush values from local data cache in single item mode or in bulk mode (used for dependent items and values received in bulk).  

Preprocessing workers
Zabbix server configuration file allows users to set count of preprocessing worker processes. StartPreprocessors configuration parameter should be used to set number of pre-forked instances of preprocessing workers. Optimal number of preprocessing workers can be determined by many factors, including the count of “preprocessable” items (items that require to execute any preprocessing steps), count of data gathering processes, average step count for item preprocessing, etc.

But assuming that there is no heavy preprocessing operations like parsing of large XML / JSON chunks, number of preprocessing workers can match total number of data gatherers. This way, there will mostly (except for the cases when data from gatherer comes in bulk) be at least one unoccupied preprocessing worker for collected data.

Too many data gathering processes (pollers, unreachable pollers, ODBC pollers, HTTP pollers, Java pollers, pingers, trappers, proxypollers) together with IPMI manager, SNMP trapper and preprocessing workers can exhaust the per-process file descriptor limit for the preprocessing manager. This will cause Zabbix server to stop (usually shortly after the start, but sometimes it can take more time). The configuration file should be revised or the limit should be raised to avoid this situation.

3 JSONPath functionality

Overview

This section provides details of supported JSONPath functionality in item value preprocessing steps.

JSONPath consists of segments separated with dots. A segment can be either a simple word like a JSON value name, * or a more complex construct enclosed within square brackets [ ]. The separating dot before bracket segment is optional and can be omitted.

For example:

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$.object.name</td>
<td>Return the object.name contents.</td>
</tr>
<tr>
<td>$.object['name']</td>
<td>Return the object.name contents.</td>
</tr>
<tr>
<td>$['object']['name']</td>
<td>Return the object.name contents.</td>
</tr>
<tr>
<td>$.['object'].['name']</td>
<td>Return the object.name contents.</td>
</tr>
<tr>
<td>$.object.history.length()</td>
<td>Return the number of object.history array elements.</td>
</tr>
<tr>
<td>$([@.name == 'Object'].price.first()</td>
<td>Return the price field of the first object with name 'Object'.</td>
</tr>
<tr>
<td>'Object').history.first().length()</td>
<td>Return the number of history array elements of the first object with name 'Object'.</td>
</tr>
<tr>
<td>$([@.price &gt; 10]).length()</td>
<td>Return the number of objects with price being greater than 10.</td>
</tr>
</tbody>
</table>

See also: Escaping special characters from LLD macro values in JSONPath.

Supported segments

<table>
<thead>
<tr>
<th>Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;name&gt;</td>
<td>Match object property by name.</td>
</tr>
<tr>
<td>*</td>
<td>Match all object properties.</td>
</tr>
<tr>
<td>['&lt;name&gt;']</td>
<td>Match object property by name.</td>
</tr>
<tr>
<td>['&lt;name&gt;',... ]</td>
<td>Match object property by any of the listed names.</td>
</tr>
<tr>
<td>['&lt;name&gt;',... ]</td>
<td>Match array element by the index.</td>
</tr>
<tr>
<td>[&lt;number&gt;,... ]</td>
<td>Match array element by any of the listed indexes.</td>
</tr>
<tr>
<td>[*]</td>
<td>Match all object properties or array elements.</td>
</tr>
<tr>
<td>[&lt;start&gt;:&lt;end&gt;]</td>
<td>Match array elements by the defined range: &lt;start&gt; - the first index to match (including). If not specified matches all array elements from the beginning. If negative specifies starting offset from the end of array. &lt;end&gt; - the last index to match (excluding). If not specified matches all array elements to the end. If negative specifies starting offset from the end of array.</td>
</tr>
<tr>
<td>[?(&lt;expression&gt;)]</td>
<td>Match objects/array elements by applying filter expression.</td>
</tr>
</tbody>
</table>

To find a matching segment ignoring its ancestry (detached segment) it must be prefixed with `.`, for example $.name or $..['name'] return values of all ‘name’ properties.
Matched element names can be extracted by adding a `-` suffix to the JSONPath. It returns the name of the matched object or an index in string format of the matched array item. The output format follows the same rules as other JSONPath queries - definite path results are returned 'as is' and indefinite path results are returned in array. However there is not much point of extracting the name of an element matching a definite path - it’s already known.

Filter expression

Filter expression is a arithmetical expression in infix notation.

Supported operands:

<table>
<thead>
<tr>
<th>Operand</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;&lt;text&gt;&quot;</td>
<td>Text constant.</td>
<td>'value: &quot;1&quot;'</td>
</tr>
<tr>
<td>'&lt;text&gt;'</td>
<td></td>
<td>'value: '1''</td>
</tr>
<tr>
<td>&lt;number&gt;</td>
<td>Numeric constant supporting scientific notation.</td>
<td>123</td>
</tr>
<tr>
<td>&lt;jsonpath starting with $&gt;</td>
<td>Value referred to by the JSONPath from the input document root node; only definite paths are supported.</td>
<td>$.object.name</td>
</tr>
<tr>
<td>&lt;jsonpath starting with @&gt;</td>
<td>Value referred to by the JSONPath from the current object/element; only definite paths are supported.</td>
<td>@.name</td>
</tr>
</tbody>
</table>

Supported operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Type</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>binary</td>
<td>Subtraction.</td>
<td>Number.</td>
</tr>
<tr>
<td>+</td>
<td>binary</td>
<td>Addition.</td>
<td>Number.</td>
</tr>
<tr>
<td>/</td>
<td>binary</td>
<td>Division.</td>
<td>Number.</td>
</tr>
<tr>
<td>*</td>
<td>binary</td>
<td>Multiplication.</td>
<td>Number.</td>
</tr>
<tr>
<td>==</td>
<td>binary</td>
<td>Is equal to.</td>
<td>Boolean (1 or 0).</td>
</tr>
<tr>
<td>!=</td>
<td>binary</td>
<td>Is not equal to.</td>
<td>Boolean (1 or 0).</td>
</tr>
<tr>
<td>&lt;</td>
<td>binary</td>
<td>Is less than.</td>
<td>Boolean (1 or 0).</td>
</tr>
<tr>
<td>&lt;=</td>
<td>binary</td>
<td>Is less than or equal to.</td>
<td>Boolean (1 or 0).</td>
</tr>
<tr>
<td>&gt;</td>
<td>binary</td>
<td>Is greater than.</td>
<td>Boolean (1 or 0).</td>
</tr>
<tr>
<td>&gt;=</td>
<td>binary</td>
<td>Is greater than or equal to.</td>
<td>Boolean (1 or 0).</td>
</tr>
<tr>
<td>=~</td>
<td>binary</td>
<td>Matches regular expression.</td>
<td>Boolean (1 or 0).</td>
</tr>
<tr>
<td>!</td>
<td>unary</td>
<td>Boolean not.</td>
<td>Boolean (1 or 0).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>binary</td>
</tr>
<tr>
<td>&amp; &amp;</td>
<td>binary</td>
<td>Boolean and.</td>
<td>Boolean (1 or 0).</td>
</tr>
</tbody>
</table>

Functions

Functions can be used at the end of JSONPath. Multiple functions can be chained if the preceding function returns value that is accepted by the following function.

Supported functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg</td>
<td>Average value of numbers in input array.</td>
<td>Array of numbers.</td>
<td>Number.</td>
</tr>
<tr>
<td>min</td>
<td>Minimum value of numbers in input array.</td>
<td>Array of numbers.</td>
<td>Number.</td>
</tr>
<tr>
<td>max</td>
<td>Maximum value of numbers in input array.</td>
<td>Array of numbers.</td>
<td>Number.</td>
</tr>
<tr>
<td>sum</td>
<td>Sum of numbers in input array.</td>
<td>Array of numbers.</td>
<td>Number.</td>
</tr>
<tr>
<td>length</td>
<td>Number of elements in input array.</td>
<td>Array.</td>
<td>A JSON construct (object, array, value) depending on input array contents.</td>
</tr>
<tr>
<td>first</td>
<td>The first array element.</td>
<td>Array.</td>
<td></td>
</tr>
</tbody>
</table>

Quoted numeric values are accepted by the JSONPath aggregate functions. It means that the values are converted from string type to numeric if aggregation is required.

Incompatible input will cause the function to generate error.
Output value

JSONPaths can be divided into definite and indefinite paths. A definite path can return only null or a single match. An indefinite path can return multiple matches, basically JSONPaths with detached, multiple name/index list, array slice or expression segments. However, when a function is used the JSONPath becomes definite, as functions always output single value.

A definite path returns the object/array/value it’s referencing, while indefinite path returns an array of the matched objects/arrays/values.

Whitespace

Whitespace (space, tab characters) can be freely used in bracket notation segments and expressions, for example, $[ 'a' ][ 0 ][ ?( $.b == 'c' ) ][ : -1 ].first() .

Strings

Strings should be enclosed with single ’ or double ” quotes. Inside the strings, single or double quotes (depending on which are used to enclose it) and backslashes \ are escaped with the backslash \ character.

Examples

Input data

```json
{
    "books": [
        {
            "category": "reference",
            "author": "Nigel Rees",
            "title": "Sayings of the Century",
            "price": 8.95,
            "id": 1
        },
        {
            "category": "fiction",
            "author": "Evelyn Waugh",
            "title": "Sword of Honour",
            "price": 12.99,
            "id": 2
        },
        {
            "category": "fiction",
            "author": "Herman Melville",
            "title": "Moby Dick",
            "isbn": "0-553-21311-3",
            "price": 8.99,
            "id": 3
        },
        {
            "category": "fiction",
            "author": "J. R. R. Tolkien",
            "title": "The Lord of the Rings",
            "isbn": "0-395-19395-8",
            "price": 22.99,
            "id": 4
        }
    ],
    "services": {
        "delivery": {
            "servicegroup": 1000,
            "description": "Next day delivery in local town",
            "active": true,
            "price": 5
        },
        "bookbinding": {
            "servicegroup": 1001,
            "description": "Printing and assembling book in A5 format",
            "active": true,
```
"price": 154.99
},
"restoration": {
  "servicegroup": 1002,
  "description": "Various restoration methods",
  "active": false,
  "methods": [
    {
      "description": "Chemical cleaning",
      "price": 46
    },
    {
      "description": "Pressing pages damaged by moisture",
      "price": 24.5
    },
    {
      "description": "Rebinding torn book",
      "price": 99.49
    }
  ]
},
"filters": {
  "price": 10,
  "category": "fiction",
  "no filters": "no \"filters\"
},
"closed message": "Store is closed",
"tags": [
  "a",
  "b",
  "c",
  "d",
  "e"
]

<table>
<thead>
<tr>
<th>JSONPath</th>
<th>Type</th>
<th>Result</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$.filters.price</td>
<td>definite</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>$.filters.category</td>
<td>definite</td>
<td>fiction</td>
<td></td>
</tr>
<tr>
<td>$.filters['no filters']</td>
<td>undefined</td>
<td>no &quot;filters&quot;</td>
<td></td>
</tr>
<tr>
<td>$.filters</td>
<td>definite</td>
<td>{</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;price&quot;: 10,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;category&quot;: &quot;fiction&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;no filters&quot;: &quot;no &quot;filters&quot;&quot;</td>
<td></td>
</tr>
<tr>
<td>$.books[1].title</td>
<td>definite</td>
<td>Sword of Honour</td>
<td></td>
</tr>
<tr>
<td>$.books[-1].author</td>
<td>definite</td>
<td>J. R. R. Tolkien</td>
<td></td>
</tr>
<tr>
<td>$.books.length</td>
<td>definite</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$.tags[: ]</td>
<td>indefinite</td>
<td>[&quot;a&quot;, &quot;b&quot;, &quot;c&quot;, &quot;d&quot;, &quot;e&quot;]</td>
<td></td>
</tr>
<tr>
<td>$.tags[2:]</td>
<td>indefinite</td>
<td>[&quot;c&quot;, &quot;d&quot;, &quot;e&quot;]</td>
<td></td>
</tr>
<tr>
<td>$.tags[3:]</td>
<td>indefinite</td>
<td>[&quot;a&quot;, &quot;b&quot;, &quot;c&quot;]</td>
<td></td>
</tr>
<tr>
<td>$.tags[1:4]</td>
<td>indefinite</td>
<td>[&quot;b&quot;, &quot;c&quot;, &quot;d&quot;]</td>
<td></td>
</tr>
<tr>
<td>$.tags[2:]</td>
<td>indefinite</td>
<td>[&quot;d&quot;, &quot;e&quot;]</td>
<td></td>
</tr>
<tr>
<td>$.tags[3:]</td>
<td>indefinite</td>
<td>[&quot;a&quot;, &quot;b&quot;]</td>
<td></td>
</tr>
<tr>
<td>$.tags[3:].length()</td>
<td>definite</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$.books[0, 2].title</td>
<td>indefinite</td>
<td>[&quot;Sayings of the Century&quot;, &quot;Moby Dick&quot;]</td>
<td></td>
</tr>
<tr>
<td>$.books[1]['author', 'title']</td>
<td>indefinite</td>
<td>[&quot;Evelyn Waugh&quot;, &quot;Sword of Honour&quot;]</td>
<td></td>
</tr>
<tr>
<td>$.id</td>
<td>indefinite</td>
<td>[1, 2, 3, 4]</td>
<td></td>
</tr>
<tr>
<td>JSONPath</td>
<td>Type</td>
<td>Result</td>
<td>Comments</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>$.services..price</td>
<td>indefinite</td>
<td>[5, 154.99, 46, 24.5, 99.49]</td>
<td>This query shows that arithmetical operations can be used in queries. Of course this query can be simplified to <code>$.services[?(@.id == 2)].price</code></td>
</tr>
<tr>
<td>$.books[?(@.id == 2 - 0.4) * 5].title</td>
<td>indefinite</td>
<td>&quot;Sword of Honour&quot;, &quot;The Lord of the Rings&quot;</td>
<td></td>
</tr>
<tr>
<td>$.books[?(@.id == 2</td>
<td></td>
<td>@.id == 4)].title</td>
<td>indefinite</td>
</tr>
<tr>
<td>$.books[?(!(@.id == 2)).title]</td>
<td>indefinite</td>
<td>&quot;SayingsoftheCentury&quot;, &quot;MobyDick&quot;, &quot;TheLordoftheRings&quot;</td>
<td></td>
</tr>
<tr>
<td>$.books[?(@.id != 2)].title</td>
<td>indefinite</td>
<td>&quot;SayingsoftheCentury&quot;, &quot;Sword of Honour&quot;, &quot;The Lord of the Rings&quot;</td>
<td></td>
</tr>
<tr>
<td>$.books[?(@.price &gt; $.filters.price)].title</td>
<td>indefinite</td>
<td>&quot;Sword of Honour&quot;, &quot;The Lord of the Rings&quot;</td>
<td></td>
</tr>
<tr>
<td>$.books[?(@.category == $.filters.category)].title</td>
<td>indefinite</td>
<td>&quot;Sword of Honour&quot;, &quot;MobyDick&quot;, &quot;The Lord of the Rings&quot;</td>
<td></td>
</tr>
</tbody>
</table>

```
[...
{
"category": "reference",
"author": "Nigel Rees",
"title": "SayingsoftheCentury",
"price": 8.95,
"id": 1
},
{
"category": "fiction",
"author": "Evelyn Waugh",
"title": "SwordofHonour",
"price": 12.99,
"id": 2
},
{
"category": "fiction",
"author": "Herman Melville",
"title": "MobyDick",
"isbn": "0-553-21311-3",
"price": 8.99,
"id": 3
},
{
"category": "fiction",
"author": "J. R. R. Tolkien",
"title": "TheLordoftheRings",
"isbn": "0-395-19395-8",
"price": 22.99,
"id": 4
}
]```
Escaping special characters from LLD macro values in JSONPath

When low-level discovery macros are used in JSONPath preprocessing and their values are resolved, the following rules of escaping special characters are applied:

- only backslash (\) and double quote ("") characters are considered for escaping;
- if the resolved macro value contains these characters, each of them is escaped with a backslash;
- if they are already escaped with a backslash, it is not considered as escaping and both the backslash and the following special characters are escaped once again.

For example:

<table>
<thead>
<tr>
<th>JSONPath</th>
<th>LLD macro value</th>
<th>After substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$.[@.value == &quot;{#MACRO}&quot;]</td>
<td>special &quot;value&quot;</td>
<td>$.[@.value == &quot;special &quot;value&quot;&quot;]</td>
</tr>
<tr>
<td>c:\temp</td>
<td>$.[@.value == &quot;c:\temp&quot;]</td>
<td>$.[@.value == &quot;a\b&quot;]</td>
</tr>
<tr>
<td>a\b</td>
<td>$.[@.value == &quot;a\b&quot;]</td>
<td></td>
</tr>
</tbody>
</table>

When used in the expression the macro that may have special characters should be enclosed in double quotes:

<table>
<thead>
<tr>
<th>JSONPath</th>
<th>LLD macro value</th>
<th>After substitution</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$.[@.value == &quot;{#MACRO}&quot;]</td>
<td>special &quot;value&quot;</td>
<td>$.[@.value == &quot;special &quot;value&quot;]</td>
<td>OK</td>
</tr>
<tr>
<td>$.[@.value == {#MACRO}]</td>
<td>special &quot;value&quot;</td>
<td>$.[@.value == special &quot;value&quot;]</td>
<td>Bad JSONPath expression</td>
</tr>
</tbody>
</table>

When used in the path the macro that may have special characters should be enclosed in square brackets and double quotes:

<table>
<thead>
<tr>
<th>JSONPath</th>
<th>LLD macro value</th>
<th>After substitution</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$.&quot;{#MACRO}&quot;.value</td>
<td>c:\temp</td>
<td>$.[c:\temp].value</td>
<td>OK</td>
</tr>
</tbody>
</table>
4 JavaScript preprocessing

Overview

This section provides details of preprocessing by JavaScript.

JavaScript preprocessing

JavaScript preprocessing is done by invoking JavaScript function with a single parameter ‘value’ and user provided function body. The preprocessing step result is the value returned from this function, for example, to perform Fahrenheit to Celsius conversion user must enter:

```
return (value - 32) * 5 / 9
```

in JavaScript preprocessing parameters, which will be wrapped into a JavaScript function by server:

```
function (value)
{
    return (value - 32) * 5 / 9
}
```

The input parameter ‘value’ is always passed as a string. The return value is automatically coerced to string via ToString() method (if it fails then the error is returned as string value), with a few exceptions:

- returning undefined value will result in an error
- returning null value will cause the input value to be discarded, much like ‘Discard value’ preprocessing on ‘Custom on fail’ action.

Errors can be returned by throwing values/objects (normally either strings or Error objects).

For example:

```
if (value == 0)
    throw "Zero input value"
return 1/value
```

Each script has a 10 second execution timeout (depending on the script it might take longer for the timeout to trigger); exceeding it will return error. A 64 megabyte heap limit is enforced.

The JavaScript preprocessing step bytecode is cached and reused when the step is applied next time. Any changes to the item’s preprocessing steps will cause the cached script to be reset and recompiled later.

Consecutive runtime failures (3 in a row) will cause the engine to be reinitialized to mitigate the possibility of one script breaking the execution environment for the next scripts (this action is logged with DebugLevel 4 and higher).

JavaScript preprocessing is implemented with Duktape (https://duktape.org/) JavaScript engine.

See also: Additional JavaScript objects and global functions

Using macros in scripts

It is possible to use user macros in JavaScript code. If a script contains user macros, these macros are resolved by server/proxy before executing specific preprocessing steps. Note, that when testing preprocessing steps in the frontend, macro values will not be pulled and need to be entered manually.

Context is ignored when a macro is replaced with its value. Macro value is inserted in the code as is, it is not possible to add additional escaping before placing the value in the JavaScript code. Please be advised, that this can cause JavaScript errors in some cases.

In an example below, if received value exceeds a {$THRESHOLD} macro value, the threshold value (if present) will be returned instead:

```
var threshold = '{$THRESHOLD}';
return (!isNaN(threshold) && value > threshold) ? threshold : value;
```

Additional JavaScript objects
Overview

This section describes Zabbix additions to the JavaScript language implemented with Duktape and supported global JavaScript functions.

Built-in objects

Zabbix

The Zabbix object provides interaction with the internal Zabbix functionality.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>log(loglevel, message)</td>
<td>Writes &lt;message&gt; into Zabbix log using &lt;loglevel&gt; log level (see configuration file DebugLevel parameter).</td>
</tr>
</tbody>
</table>

Example:

Zabbix.log(3, "this is a log entry written with 'Warning' log level")

You may use the following aliases:

<table>
<thead>
<tr>
<th>Alias</th>
<th>Alias to</th>
</tr>
</thead>
<tbody>
<tr>
<td>console.log(object)</td>
<td>Zabbix.log(4, JSON.stringify(object))</td>
</tr>
<tr>
<td>console.warn(object)</td>
<td>Zabbix.log(3, JSON.stringify(object))</td>
</tr>
<tr>
<td>console.error(object)</td>
<td>Zabbix.log(2, JSON.stringify(object))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sleep(delay)</td>
<td>Delay JavaScript execution by delay milliseconds.</td>
</tr>
</tbody>
</table>

Example (delay execution by 15 seconds):

Zabbix.sleep(15000)

HttpRequest

This object encapsulates cURL handle allowing to make simple HTTP requests. Errors are thrown as exceptions.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addHeader(name, value)</td>
<td>Adds HTTP header field. This field is used for all following requests until cleared with the clearHeader() method.</td>
</tr>
<tr>
<td>clearHeader()</td>
<td>Clears HTTP header. If no header fields are set, HttpRequest will set Content-Type to application/json if the data being posted is JSON-formatted; text/plain otherwise.</td>
</tr>
<tr>
<td>connect(url)</td>
<td>Sends HTTP CONNECT request to the URL and returns the response.</td>
</tr>
<tr>
<td>customRequest(method, url, data)</td>
<td>Allows to specify any HTTP method in the first parameter. Sends the method request to the URL with optional data payload and returns the response.</td>
</tr>
<tr>
<td>delete(url, data)</td>
<td>Sends HTTP DELETE request to the URL with optional data payload and returns the response.</td>
</tr>
<tr>
<td>get(url, data)</td>
<td>Sends HTTP GET request to the URL with optional data payload and returns the response.</td>
</tr>
<tr>
<td>head(url)</td>
<td>Sends HTTP HEAD request to the URL and returns the response.</td>
</tr>
<tr>
<td>options(url)</td>
<td>Sends HTTP OPTIONS request to the URL and returns the response.</td>
</tr>
<tr>
<td>patch(url, data)</td>
<td>Sends HTTP PATCH request to the URL with optional data payload and returns the response.</td>
</tr>
<tr>
<td>put(url, data)</td>
<td>Sends HTTP PUT request to the URL with optional data payload and returns the response.</td>
</tr>
<tr>
<td>post(url, data)</td>
<td>Sends HTTP POST request to the URL with optional data payload and returns the response.</td>
</tr>
<tr>
<td>getHeaders(&lt;asArray&gt;)</td>
<td>Returns the object of received HTTP header fields.</td>
</tr>
<tr>
<td></td>
<td>The asArray parameter may be set to “true” (e.g. getHeaders(true)), “false” or be undefined. If set to “true” the received HTTP header field values will be returned as arrays; this should be used to retrieve the field values of multiple same-name headers.</td>
</tr>
<tr>
<td></td>
<td>If not set or set to “false”, the received HTTP header field values will be returned as strings.</td>
</tr>
<tr>
<td>getStatus()</td>
<td>Returns the status code of the last HTTP request.</td>
</tr>
<tr>
<td>setProxy(proxy)</td>
<td>Sets HTTP proxy to “proxy” value. If this parameter is empty then no proxy is used.</td>
</tr>
</tbody>
</table>
### Method Description

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>setHttpAuth(bitmask, username, password)</strong></td>
<td>Sets enabled HTTP authentication methods (HTTPAUTH_BASIC, HTTPAUTH_DIGEST, HTTPAUTH_NEGOTIATE, HTTPAUTH_NTLM, HTTPAUTH_NONE) in the 'bitmask' parameter. The HTTPAUTH_NONE flag allows to disable HTTP authentication. Examples: request.setHttpAuth(HTTPAUTH_NTLM</td>
</tr>
<tr>
<td><strong>trace(url, data)</strong></td>
<td>Sends HTTP TRACE request to the URL with optional data payload and returns the response.</td>
</tr>
</tbody>
</table>

**Example:**

```javascript
try {
    Zabbix.log(4, 'jira webhook script value='+value);

    var result = {
        'tags': {
            'endpoint': 'jira'
        }
    },
    params = JSON.parse(value),
    req = new HttpRequest(),
    fields = {},
    resp;

    req.setHeader('Content-Type: application/json');
    req.setHeader('Authorization: Basic '+params.authentication);

    fields.summary = params.summary;
    fields.description = params.description;
    fields.project = {'key': params.project_key};
    fields.issuetype = {'id': params.issue_id};
    resp = req.post('https://tsupport.zabbix.lan/rest/api/2/issue/',
        JSON.stringify({"fields": fields})
    );

    if (req.getStatus() != 201) {
        throw 'Response code: '+req.getStatus();
    }

    resp = JSON.parse(resp);
    result.tags.issue_id = resp.id;
    result.tags.issue_key = resp.key;
} catch (error) {
    Zabbix.log(4, 'jira issue creation failed json : '+JSON.stringify({"fields": fields}));
    Zabbix.log(4, 'jira issue creation failed : '+error);

    result = {};
}
return JSON.stringify(result);
```

**XML**

The XML object allows the processing of XML data in the item and low-level discovery preprocessing and webhooks.

In order to use XML object, server/proxy must be compiled with libxml2 support.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XML.query(data, expression)</strong></td>
<td>Retrieves node content using XPath. Returns null if node is not found. <strong>expression</strong> - an XPath expression; <strong>data</strong> - XML data as a string.</td>
</tr>
<tr>
<td><strong>XML.toJson(data)</strong></td>
<td>Converts data in XML format to JSON.</td>
</tr>
</tbody>
</table>
**Method**

**Description**

XML.fromJson(object) Converts data in JSON format to XML.

---

**Example:**

**Input:**

```xml
<menu>
  <food type = "breakfast">
    <name>Chocolate</name>
    <price>$5.95</price>
    <description></description>
    <calories>650</calories>
  </food>
</menu>
```

**Output:**

```json
{
  "menu": {
    "food": {
      "@type": "breakfast",
      "name": "Chocolate",
      "price": "$5.95",
      "description": null,
      "calories": "650"
    }
  }
}
```

**Serialization rules**

XML to JSON conversion will be processed according to the following rules (for JSON to XML conversions reversed rules are applied):

1. **XML attributes will be converted to keys that have their names prepended with '@'.**

   **Example:**

   **Input:**

   ```xml
   <xml foo="FOO">
     <bar>
       <baz>BAZ</baz>
     </bar>
   </xml>
   ```

   **Output:**

   ```json
   {
     "xml": {
       "@foo": "FOO",
       "bar": {
         "baz": "BAZ"
       }
     }
   }
   ```

2. **Self-closing elements (<foo/>) will be converted as having 'null' value.**

   **Example:**

   **Input:**

   ```xml
   <xml>
     <foo/>
   </xml>
   ```

   **Output:**

   ```json
   {
   }
   ```
3. Empty attributes (with "" value) will be converted as having empty string ("") value.

Example:
Input:
```xml
<xml>
  <foo bar="" />
</xml>
```

Output:
```json
{
  "xml": {
    "foo": null
  }
}
```

4. Multiple child nodes with the same element name will be converted to a single key that has an array of values as its value.

Example:
Input:
```xml
<xml>
  <foo>BAR</foo>
  <foo>BAZ</foo>
  <foo>QUX</foo>
</xml>
```

Output:
```json
{
  "xml": {
    "foo": ["BAR", "BAZ", "QUX"]
  }
}
```

5. If a text element has no attributes and no children, it will be converted as a string.

Example:
Input:
```xml
<xml>
  <foo>BAZ</foo>
</xml>
```

Output:
```json
{
  "xml": {
    "foo": "BAZ"
  }
}
```

6. If a text element has no children, but has attributes: text content will be converted to an element with the key `#text` and content as a value; attributes will be converted as described in the serialization rule 1.

Example:
Input:
```xml
<xml>
  <foo bar="BAR">170
</xml>
```

Output:
```json
{
  "xml": {
    "foo": {
      "@bar": "BAR",
      "#text": "170"
    }
  }
}
```
Global JavaScript functions

Additional global JavaScript functions have been implemented with Duktape:

- `btoa(string)` - encodes string to base64 string
- `atob(base64_string)` - decodes base64 string

```
try {
  b64 = btoa("utf8 string");
  utf8 = atob(b64);
}
```

- `md5(string)` - calculates the MD5 hash of a string
- `sha256(string)` - calculates the SHA256 hash of a string
- `hmac(<hash type>,key,string)` - returns HMAC hash as hex formatted string. MD5 and SHA256 hash types are supported. Key and string parameters support binary data. E. g.:
  - `hmac('md5',key,string)`
  - `hmac('sha256',key,string)`

5 CSV to JSON preprocessing

Overview

In this preprocessing step it is possible to convert CSV file data into JSON format. It’s supported in:

- items (item prototypes)
- low-level discovery rules

Configuration

To configure a CSV to JSON preprocessing step:

- Go to the Preprocessing tab in item/discovery rule configuration
- Click on Add
- Select the CSV to JSON option

<table>
<thead>
<tr>
<th>Item</th>
<th>Tags</th>
<th>Preprocessing</th>
<th>Name</th>
<th>Parameters</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessing steps</td>
<td></td>
<td></td>
<td>CSV to JSON</td>
<td></td>
<td>With header</td>
</tr>
<tr>
<td>Custom on fail</td>
<td></td>
<td></td>
<td>Discard value</td>
<td></td>
<td>Set value to</td>
</tr>
</tbody>
</table>
The first parameter allows to set a custom delimiter. Note that if the first line of CSV input starts with "Sep=" and is followed by a single UTF-8 character then that character will be used as the delimiter in case the first parameter is not set. If the first parameter is not set and a delimiter is not retrieved from the "Sep=" line, then a comma is used as a separator.

The second optional parameter allows to set a quotation symbol.

If the With header row checkbox is marked, the header line values will be interpreted as column names (see Header processing for more information).

If the Custom on fail checkbox is marked, the item will not become unsupported in case of a failed preprocessing step. Additionally custom error handling options may be set: discard the value, set a specified value or set a specified error message.

Header processing

The CSV file header line can be processed in two different ways:

- If the With header row checkbox is marked - header line values are interpreted as column names. In this case the column names must be unique and the data row should not contain more columns than the header row;
- If the With header row checkbox is not marked - the header line is interpreted as data. Column names are generated automatically (1,2,3,4...)

CSV file example:

<table>
<thead>
<tr>
<th>Nr, Item name, Key, Qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, active agent item, agent.hostname, 33</td>
</tr>
<tr>
<td>&quot;2&quot;, &quot;passive agent item&quot;, &quot;agent.version&quot;, &quot;44&quot;</td>
</tr>
<tr>
<td>3, &quot;active, passive agent items&quot;, agent.ping, 55</td>
</tr>
</tbody>
</table>

A quotation character within a quoted field in the input must be escaped by preceding it with another quotation character.

Processing header line

JSON output when a header line is expected:

```
[  
  {  
    "Nr": "1",  
    "Item name": "active agent item",  
    "Key": "agent.hostname",  
    "Qty": "33"  
  },  
  {  
    "Nr": "2",  
    "Item name": "passive agent item",  
    "Key": "agent.version",  
    "Qty": "44"  
  },  
  {  
    "Nr": "3",  
    "Item name": "active, passive agent items",  
    "Key": "agent.ping",  
    "Qty": "55"  
  }  
]
```

No header line processing

JSON output when a header line is not expected:

```
[  
  {  
    "1": "Nr",  
    "2": "Item name",  
    "3": "Key",  
    "4": "Qty"  
  },  
  {  
    "1": "1",  
    "2": "active agent item",  
    "3": "agent.hostname"  
  }  
]```
3 Item types

Overview

Item types cover various methods of acquiring data from your system. Each item type comes with its own set of supported item keys and required parameters.

The following item types are currently offered by Zabbix:

- Zabbix agent checks
- SNMP agent checks
- SNMP traps
- IPMI checks
- Simple checks
  - VMware monitoring
- Log file monitoring
- Calculated items
  - Aggregate calculations
- Zabbix internal checks
- SSH checks
- Telnet checks
- External checks
- Trapper items
- JMX monitoring
- ODBC checks
- Dependent items
- HTTP checks
- Prometheus checks
- Script items

Details for all item types are included in the subpages of this section. Even though item types offer a lot of options for data gathering, there are further options through user parameters or loadable modules.

Some checks are performed by Zabbix server alone (as agent-less monitoring) while others require Zabbix agent or even Zabbix Java gateway (with JMX monitoring).

If a particular item type requires a particular interface (like an IPMI check needs an IPMI interface on the host) that interface must exist in the host definition.

Multiple interfaces can be set in the host definition: Zabbix agent, SNMP agent, JMX and IPMI. If an item can use more than one interface, it will search the available host interfaces (in the order: Agent→SNMP→JMX→IPMI) for the first appropriate one to be linked with.

All items that return text (character, log, text types of information) can return whitespace only as well (where applicable) setting the return value to an empty string (supported since 2.0).

1 Zabbix agent

Overview
This section provides details on the item keys that use communication with Zabbix agent for data gathering.

There are passive and active agent checks. When configuring an item, you can select the required type:

- Zabbix agent - for passive checks
- Zabbix agent (active) - for active checks

Note that all item keys supported by Zabbix agent on Windows are also supported by the new generation Zabbix agent 2. See the additional item keys that you can use with the agent 2 only.

Supported platforms

Except where specified differently in the item comments, the agent items (and all parameters) are supported on:

- Linux
- FreeBSD
- Solaris
- HP-UX
- AIX
- Tru64
- MacOS X
- OpenBSD
- NetBSD

Many agent items are also supported on Windows. See the Windows agent item page for details.

Supported item keys

The item keys that you can use with Zabbix agent are listed below. The items are grouped in tables by item family.

Parameters without angle brackets are mandatory. Parameters marked with angle brackets < > are optional.

Kernel data

<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel.maxfiles</td>
<td>Maximum number of opened files</td>
<td>Integer</td>
<td></td>
<td>Supported platforms: Linux, FreeBSD, MacOS X, OpenBSD, NetBSD.</td>
</tr>
<tr>
<td>kernel.maxproc</td>
<td>Maximum number of processes</td>
<td>Integer</td>
<td></td>
<td>Supported platforms: Linux 2.6 and later, FreeBSD, Solaris, MacOS X, OpenBSD, NetBSD.</td>
</tr>
<tr>
<td>kernel.openfiles</td>
<td>Return the number of currently open file descriptors.</td>
<td>Integer</td>
<td></td>
<td>Supported platforms: Linux. The item may work on other UNIX-like platforms. This item is supported since Zabbix 6.0.</td>
</tr>
</tbody>
</table>

Log data

See additional information on log monitoring.

<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>log[filename]</td>
<td>log data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item key</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monitoring of a log file.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>log - full path and name of log file</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>regexp - regular expression describing the required pattern</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>encoding - code page identifier</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxlines - maximum number of new lines per second the agent will send to Zabbix server or proxy. This parameter overrides the value of 'MaxLinesPerSecond' in zabbix_agentd.conf</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mode (since version 2.0)- possible values: all (default), skip - skip processing of older data (affects only newly created items).</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>output (since version 2.2) - an optional output formatting template. The \0 escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an \N (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxdelay (since version 3.2) - maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; &gt; 0.0 - ignore older lines in order to get the most recent lines analyzed within &quot;maxdelay&quot; seconds. Read the maxdelay notes before using it!</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>options (since version 4.4.7) - additional options: mtime-noread - non-unique records, reread only if the file size changes (ignore modification time change). (This parameter is deprecated since 5.0.2, because now mtime is ignored.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>persistent_dir (since versions 5.0.18, 5.4.9, only in zabbix_agentd on Unix systems; not supported in Agent2) - absolute pathname of directory where to store persistent files. See also additional notes on persistent files.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See supported platforms.

The item must be configured as an active check.

If file is missing or permissions do not allow access, item turns unsupported.

If output is left empty - the whole line containing the matched text is returned. Note that all global regular expression types except 'Result is TRUE' always return the whole matched line and the output parameter is ignored.

Content extraction using the output parameter takes place on the agent.

Examples:

```plaintext
=> log[/var/log/syslog]
=> log[/var/log/syslog,error]
=> log[/home/zabbix/logs/logfile",100]
```

Using output parameter for extracting a number from log record:

```plaintext
=> log[/app1/app.log,"taskrun[0-9.]+ sec, processed [0-9]+ records, [0-9]+ errors""\1"] → will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors" and send only '6080' to server. Because a numeric value is being sent, the "Type of information" for this item can be set to "Numeric (unsigned)" and the value can be used in graphs, triggers etc.
```

Using output parameter for rewriting log record before sending to server:

```plaintext
=> log[/app1/app.log,"\{0-9\}+ task run \{0-9\}+ sec, processed \{0-9\}+ records, \{0-9\}+ errors""\3, ERRORS: \4, DURATION: \2"] → will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors" and send a modified record "2015-11-13 10:08:26 RECORDS: 6080, ERRORS: 0, DURATION: 6:08" to server.
```

**log.count**(file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>,<persistent_dir>)
<table>
<thead>
<tr>
<th>Item key</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count of</td>
<td>Integer</td>
<td>Integer file - full path and name of log file</td>
</tr>
<tr>
<td>matched lines</td>
<td></td>
<td>regexp - regular expression describing the required pattern</td>
</tr>
<tr>
<td>in a monitored</td>
<td></td>
<td>encoding - code page identifier</td>
</tr>
<tr>
<td>log file</td>
<td></td>
<td>maxproclines - maximum number of new lines per second the agent will analyze (cannot exceed 10000). Default value is 10^&quot;MaxLinesPerSecond&quot; in zabbix_agentd.conf.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mode - possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>all (default), skip - skip processing of older data (affects only newly created items).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>maxdelay - maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; &gt; 0.0 - ignore older lines in order to get the most recent lines analyzed within &quot;maxdelay&quot; seconds.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Read the maxdelay notes before using it!</td>
</tr>
<tr>
<td></td>
<td></td>
<td>options (since version 4.4.7) - additional options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mtime-noreread - non-unique records, reread only if the file size changes (ignore modification time change). (This parameter is deprecated since 5.0.2, because now mtime is ignored.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>persistent_dir (since versions 5.0.18, 5.4.9, only in zabbix_agentd on Unix systems; not supported in Agent2) - absolute pathname of directory where to store persistent files. See also additional notes on persistent files.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>logging:///file_regexp,&lt;regexp&gt;,&lt;encoding&gt;,&lt;maxlines&gt;,&lt;mode&gt;,&lt;output&gt;,&lt;maxdelay&gt;,&lt;options&gt;,&lt;persistent_dir&gt; ]</td>
</tr>
</tbody>
</table>

See supported platforms.

The item must be configured as an active check.

Matching lines are counted in the new lines since the last log check by the agent, and thus depend on the item update interval.

If the file is missing or permissions do not allow access, item turns unsupported.

Supported since Zabbix 3.2.0.
Monitoring of a log file that is rotated.

**Log file**

**file regexp** - absolute path to file and the file name described by a regular expression. Note that only the file name is a regular expression.

**regexp** - regular expression describing the required content pattern.

**encoding** - code page identifier.

**maxlines** - maximum number of new lines per second the agent will send to Zabbix server or proxy. This parameter overrides the value of ‘MaxLinesPerSecond’ in zabbix_agentd.conf.

**mode** (since version 2.0) - possible values:

- all (default), skip - skip processing of older data (affects only newly created items).

**output** (since version 2.2) - an optional output formatting template. The \0 escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an \N (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).

**maxdelay** (since version 3.2) - maximum delay in seconds. Type: float. Values: 0 - (default) ignore log file lines; > 0.0 - ignore older lines in order to get the most recent lines analyzed within “maxdelay” seconds. Read the maxdelay notes before using it.

**options** (since version 4.0; mtime-reread, mtime-noreread options since 4.4.7) - type of log file rotation and other options.

Possible values:

- rotate (default), copytruncate - note that copytruncate cannot be used together with maxdelay. In this case maxdelay must be 0 or not specified; see copytruncate notes, mttime-reread - non-unique records, reread if modification time or size changes (default), mttime-noreread - non-unique records, reread only if the size changes (ignore modification time change).

**persistent_dir** (since versions 5.0.18, 5.4.9, only in zabbix_agentd on Unix systems; not supported in Agent2) - absolute pathname of directory where to store persistent files. See also additional notes on persistent files.

See supported platforms.

The item must be configured as an active check.

Log rotation is based on the last modification time of files.

Note that logrt is designed to work with one currently active log file, with several other matching inactive files rotated. If, for example, a directory has many active log files, a separate logrt item should be created for each one. Otherwise if one logrt item picks up too many files it may lead to exhausted memory and a crash of monitoring.

If output is left empty - the whole line containing the matched text is returned. Note that all global regular expression types except ‘Result is TRUE’ always return the whole matched line and the output parameter is ignored.

Content extraction using the output parameter takes place on the agent.

Examples:

=> logrt["/home/zabbix/logs/^logfile[0-9]{1,3}$",100] → will match a file like "logfile1" (will not match "logfile1")

=> logrt["/home/user/^logfile_.[0-9]{1,3}$","pattern_to_match","UTF-8",100] → will collect data from files such "logfile_abc_1" or "logfile_001".

Using output parameter for extracting a number from log record:

=> logrt[/app1/^test.*log$,"task run [0-9.]+ sec, processed ([0-9]+) records, [0-9]+ errors",,"\1"] → will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors" and send only ‘6080’ to server.

Because a numeric value is being sent, the “Type of information” for this item can be set to “Numeric (unsigned)” and the value can be used in graphs, triggers etc.

Using output parameter for rewriting log record before sending to server:

=> logrt[/app1/^test.*log$,"(\[0-9-:]+) task run ([0-9-:]+) sec, processed ([0-9]+) records, ([0-9]+) errors",,"\1 RECORDS: \3, ERRORS: \4, DURATION: \2"] → will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors" and send a modified record "2015-11-13 10:08:26 RECORDS: 6080, ERRORS: 0, DURATION: 6.08" to server.

logrt.count[file regexp,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>,<persistent_dir>]
Count of matched lines in a monitored log file that is rotated.

<table>
<thead>
<tr>
<th>Item key</th>
<th>Integer</th>
<th>Description</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_regexp</td>
<td>-</td>
<td>absolute path to file and regular expression describing the file name pattern</td>
<td></td>
<td>See supported platforms.</td>
</tr>
<tr>
<td>regexp</td>
<td>-</td>
<td>regular expression describing the required content pattern</td>
<td></td>
<td>The item must be configured as an active check.</td>
</tr>
<tr>
<td>encoding</td>
<td>-</td>
<td>code page identifier</td>
<td></td>
<td>Matching lines are counted in the new lines since the last log check by the agent, and thus depend on the item update interval.</td>
</tr>
<tr>
<td>maxproclines</td>
<td>-</td>
<td>maximum number of new lines per second the agent will analyze (cannot exceed 10000). Default value is 10^&quot;MaxLinesPerSecond&quot; in zabbix_agentd.conf.</td>
<td></td>
<td>Log rotation is based on the last modification time of files.</td>
</tr>
<tr>
<td>mode</td>
<td>-</td>
<td>possible values: all (default), skip - skip processing of older data (affects only newly created items).</td>
<td></td>
<td>Supported since Zabbix 3.2.0.</td>
</tr>
<tr>
<td>maxdelay</td>
<td>-</td>
<td>maximum delay in seconds.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Type: float</td>
<td>Values: 0 - (default) never ignore log file lines; &gt; 0.0 - ignore older lines in order to get the most recent lines analyzed within &quot;maxdelay&quot; seconds.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Read the maxdelay notes before using it!</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>options (since version 4.0; mtime-reread, mtime-noreread options since 4.4.7) - type of log file rotation and other options.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: rotate (default), copytruncate - note that copytruncate cannot be used together with maxdelay.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>In this case maxdelay must be 0 or not specified; see copytruncate notes, mtime-reread - non-unique records, reread if modification time or size changes (default), mtime-noreread - non-unique records, reread only if the size changes (ignore modification time change).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>persistent_dir</td>
<td>-</td>
<td>absolute pathname of directory where to store persistent files. See also additional notes on persistent files.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Modbus data

<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
</table>
| modbus.get| [endpoint, <slave id>, <function>, <address>, <count>, <type>, <endianness>, <offset>] | }
### Item key

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>net.dns</strong></td>
<td>ip - IP address of DNS server (leave empty for the default DNS server, ignored on Windows unless using Zabbix agent 2)</td>
<td>ip - IP address of DNS server (leave empty for the default DNS server, ignored on Windows unless using Zabbix agent 2)</td>
<td>See supported platforms.</td>
</tr>
<tr>
<td>Checks if DNS service is up.</td>
<td></td>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>name - DNS name to query</td>
<td>name - DNS name to query</td>
<td>=&gt; net.dns[8.8.8.8, example.com, MX, 2, 1]</td>
</tr>
<tr>
<td></td>
<td>type - record type to be queried (default is SOA)</td>
<td>type - record type to be queried (default is SOA)</td>
<td>The possible values for type are:</td>
</tr>
<tr>
<td></td>
<td>timeout (ignored on Windows unless using Zabbix agent 2) - timeout for the request in seconds (default is 1 second)</td>
<td>timeout (ignored on Windows unless using Zabbix agent 2) - timeout for the request in seconds (default is 1 second)</td>
<td>ANY, A, NS, CNAME, MB, MG, MR, PTR, MD, MF, MX, SOA, NULL, WKS (not supported for Zabbix agent on Windows, Zabbix agent 2 on all OS), HINFO, MINFO, TXT, SRV</td>
</tr>
<tr>
<td></td>
<td>count (ignored on Windows unless using Zabbix agent 2) - number of tries for the request (default is 2)</td>
<td>count (ignored on Windows unless using Zabbix agent 2) - number of tries for the request (default is 2)</td>
<td>Internationalized domain names are not supported, please use IDNA encoded names instead.</td>
</tr>
<tr>
<td></td>
<td>protocol (since version 3.0) - the protocol used to perform DNS queries: udp (default) or tcp</td>
<td>protocol (since version 3.0) - the protocol used to perform DNS queries: udp (default) or tcp</td>
<td>Naming before Zabbix 2.0 (still supported): net.tcp.dns</td>
</tr>
</tbody>
</table>

### Network data

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>net.dns.record</strong></td>
<td>ip - IP address of DNS server (leave empty for the default DNS server, ignored on Windows unless using Zabbix agent 2)</td>
<td>ip - IP address of DNS server (leave empty for the default DNS server, ignored on Windows unless using Zabbix agent 2)</td>
<td>See supported platforms.</td>
</tr>
<tr>
<td>Performs a DNS query.</td>
<td></td>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td>name - DNS name to query</td>
<td>name - DNS name to query</td>
<td>=&gt; net.dns.record[8.8.8.8, example.com, MX, 2, 1]</td>
</tr>
<tr>
<td></td>
<td>type - record type to be queried (default is SOA)</td>
<td>type - record type to be queried (default is SOA)</td>
<td>The possible values for type are:</td>
</tr>
<tr>
<td></td>
<td>timeout (ignored on Windows unless using Zabbix agent 2) - timeout for the request in seconds (default is 1 second)</td>
<td>timeout (ignored on Windows unless using Zabbix agent 2) - timeout for the request in seconds (default is 1 second)</td>
<td>ANY, A, NS, CNAME, MB, MG, MR, PTR, MD, MF, MX, SOA, NULL, WKS (not supported for Zabbix agent on Windows, Zabbix agent 2 on all OS), HINFO, MINFO, TXT, SRV</td>
</tr>
<tr>
<td></td>
<td>count (ignored on Windows unless using Zabbix agent 2) - number of tries for the request (default is 2)</td>
<td>count (ignored on Windows unless using Zabbix agent 2) - number of tries for the request (default is 2)</td>
<td>Internationalized domain names are not supported, please use IDNA encoded names instead.</td>
</tr>
<tr>
<td></td>
<td>protocol (since version 3.0) - the protocol used to perform DNS queries: udp (default) or tcp</td>
<td>protocol (since version 3.0) - the protocol used to perform DNS queries: udp (default) or tcp</td>
<td>Naming before Zabbix 2.0 (still supported): net.tcp.dns.query</td>
</tr>
</tbody>
</table>

**net.if.collisions**

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
</table>

**Supported platforms:**

- Linux.
- Supported since Zabbix 5.2.0.
**net.if.discovery**

List of JSON object network interfaces. Used for low-level discovery.

**net.if.in[if, <mode>]**

Incoming traffic statistics on network interface.

- **if** - network interface name (Unix);
  network interface full description or IPv4 address; or, if in braces, network interface GUID (Windows)
- **mode** - possible values:
  - bytes - number of bytes (default)
  - packets - number of packets
  - errors - number of errors
  - dropped - number of dropped packets
  - overruns (fifo) - the number of FIFO buffer errors
  - frame - the number of packet framing errors
  - compressed - the number of compressed packets transmitted or received by the device driver
  - multicast - the number of multicast frames received by the device driver

**Supported platforms:**
Linux, FreeBSD, Solaris, HP-UX, AIX, MacOS X, OpenBSD, NetBSD. Root privileges are required on NetBSD.

**Examples:**
```
=> net.if.in[eth0, errors]
=> net.if.in[eth0]
```

You may use this key with the Change per second preprocessing step in order to get bytes per second statistics.

**net.if.out[if, <mode>]**

Outgoing traffic statistics on network interface.

- **if** - network interface name (Unix);
  network interface full description or IPv4 address; or, if in braces, network interface GUID (Windows)
- **mode** - possible values:
  - bytes - number of bytes (default)
  - packets - number of packets
  - errors - number of errors
  - dropped - number of dropped packets
  - overruns (fifo) - the number of FIFO buffer errors
  - collisions (colls) - the number of collisions detected on the interface
  - carrier - the number of carrier losses detected by the device driver
  - compressed - the number of compressed packets transmitted by the device driver

**Supported platforms:**
Linux, FreeBSD, Solaris, HP-UX, AIX, MacOS X, OpenBSD, NetBSD.

Root privileges are required on NetBSD.

**Examples:**
```
=> net.if.out[eth0, errors]
=> net.if.out[eth0]
```

You may use this key with the Change per second preprocessing step in order to get bytes per second statistics.

**net.if.total[if, <mode>]**

180
## Item key

<table>
<thead>
<tr>
<th>Item key</th>
<th>Type</th>
<th>Description</th>
<th>Supported platforms</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum of incoming and outgoing traffic statistic on network interface.</td>
<td>Integer</td>
<td>if - network interface name (Unix); network interface full description or IPv4 address; or, if in braces, network interface GUID (Windows)   mode - possible values: bytes - number of bytes (default) packets - number of packets errors - number of errors dropped - number of dropped packets overruns (fifo) - the number of FIFO buffer errors compressed - the number of compressed packets transmitted or received by the device driver.</td>
<td>Linux, FreeBSD, Solaris, HP-UX, AIX, MacOS X, OpenBSD, NetBSD. Root privileges are required on NetBSD. The dropped mode is supported only on Linux, HP-UX. The overruns, collision, compressed modes are supported only on Linux. On HP-UX this item does not provide details on loopback interfaces (e.g. lo0).</td>
<td>You may use this key with the Change per second preprocessing step in order to get bytes per second statistics. Note that dropped packets are supported only if both net.if.in and net.if.out work for dropped packetson your platform.</td>
</tr>
<tr>
<td><strong>net.tcp.listen</strong>[port]</td>
<td></td>
<td>Checks if this TCP port is in LISTEN state. port - TCP port number</td>
<td>Linux, FreeBSD, Solaris, MacOS X.</td>
<td>Example: =&gt; net.tcp.listen[80] On Linux supported since Zabbix 1.8.4 Since Zabbix 3.0.0, on Linux kernels 2.6.14 and above, information about listening TCP sockets is obtained from the kernel’s NETLINK interface, if possible. Otherwise, the information is retrieved from /proc/net/tcp and /proc/net/tcp6 files.</td>
</tr>
<tr>
<td><strong>net.tcp.port</strong>[&lt;ip&gt;,&lt;port&gt;]</td>
<td></td>
<td>Checks if it is possible to make TCP connection to specified port. ip - IP or DNS name (default is 127.0.0.1) port - port number</td>
<td>See supported platforms.</td>
<td>Example: =&gt; net.tcp.port[80] can be used to test availability of web server running on port 80. For simple TCP performance testing use net.tcp.service.perf[tcp,&lt;ip&gt;,&lt;port&gt;] Note that these checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually).</td>
</tr>
<tr>
<td><strong>net.tcp.service</strong>[service,&lt;ip&gt;,&lt;port&gt;]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Checks if service is running and accepting TCP connections.

<table>
<thead>
<tr>
<th>checks</th>
<th>service</th>
<th>ip</th>
<th>port</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - service is down</td>
<td>service - either of: ssh, ldap, smtp, ftp, http, pop, nntp, imap, tcp, https, telnet (see details)</td>
<td>ip - IP address (default is 127.0.0.1)</td>
<td>port - port number (by default standard service port number is used)</td>
</tr>
<tr>
<td>1 - service is running</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See supported platforms.

Example:

```
=> net.tcp.service[ftp,45] → can be used to test the availability of FTP server on TCP port 45.
```

Note that these checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually).

Checking of encrypted protocols (like IMAP on port 993 or POP on port 995) is currently not supported. As a workaround, please use net.tcp.port for checks like these.

Checking of LDAP and HTTPS on Windows is only supported by Zabbix agent 2.

Note that the telnet check looks for a login prompt (':' at the end).

See also known issues of checking HTTPS service.

https and telnet services are supported since Zabbix 2.0.

```
net.tcp.service.perf[service,<ip>,<port>]
```

Checks performance of TCP service.

<table>
<thead>
<tr>
<th>checks</th>
<th>service</th>
<th>ip</th>
<th>port</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - service is down</td>
<td>service - either of: ssh, ldap, smtp, ftp, http, pop, nntp, imap, tcp, https, telnet (see details)</td>
<td>ip - IP address (default is 127.0.0.1)</td>
<td>port - port number (by default standard service port number is used)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See supported platforms.

Example:

```
=> net.tcp.service.perf[ssh] → can be used to test the speed of initial response from SSH server.
```

Checking of encrypted protocols (like IMAP on port 993 or POP on port 995) is currently not supported. As a workaround, please use net.tcp.service.perf[tcp,<ip>,<port>] for checks like these.

Note that the telnet check looks for a login prompt (':' at the end).

See also known issues of checking HTTPS service.

https and telnet services are supported since Zabbix 2.0.

```
net.tcp.socket.count[laddr],[lport],[raddr],[rport],[state]
```

182
### Item key

<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
</table>
| Return the number of TCP sockets that match parameters. | Integer | laddr - local IPv4/6 address or CIDR subnet | lport - local port number or service name | Supported platforms: Linux. | Example: 
=> net.tcp.socket.count["80","established"]
→ check if local TCP port 80 is in "established" state |
| net.udp.listen[port] | Checks if UDP port is in LISTEN state. | port - UDP port number | Supported platforms: Linux, FreeBSD, Solaris, MacOS X. | Example: 
=> net.udp.listen[68] |
| net.udp.service[service,<ip>,<port>] | Checks if service is running and responding to UDP requests. | service - ntp (see details) | ip - IP address (default is 127.0.0.1) | See supported platforms. | Example: 
=> net.udp.service[ntp,45] → can be used to test the availability of NTP service on UDP port 45. |
| net.udp.socket.count[laddr],[lport],[raddr],[rport],[state] | Return the number of TCP sockets that match parameters. | Integer | laddr - local IPv4/6 address or CIDR subnet | Supported platforms: Linux. | Example: 
=> net.udp.socket.count[",listening"] → check if any UDP socket is in "listening" state |

### Process data

<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>proc.cpu.util[name&gt;,&lt;user&gt;,&lt;type&gt;,&lt;cmdline&gt;,&lt;mode&gt;,&lt;zone&gt;]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Item key**

| Process CPU utilization percentage. | Float | name - process name (default is all processes) | user - user name (default is all users) | type - CPU utilization type: total (default), user, system | cmdline - filter by command line (it is a regular expression) | mode - data gathering mode: avg1 (default), avg5, avg15 | zone - target zone: current (default), all. This parameter is supported on Solaris only. |

**Supported platforms:**
Linux, Solaris®.

Examples:
=> proc.cpu.util[,root] → CPU utilization of all processes running under the “root” user
=> proc.cpu.util[zabbix_server,zabbix] → CPU utilization of all zabbix_server processes running under the zabbix user

The returned value is based on single CPU core utilization percentage. For example CPU utilization of a process fully using two cores is 200%.

The process CPU utilization data is gathered by a collector which supports the maximum of 1024 unique (by name, user and command line) queries. Queries not accessed during the last 24 hours are removed from the collector.

Note that when setting the zone parameter to current (or default) in case the agent has been compiled on a Solaris without zone support, but running on a newer Solaris where zones are supported, then the agent will return NOT_SUPPORTED (the agent cannot limit results to only the current zone). However, all is supported in this case.

**Notes on selecting processes with name and cmdline parameters (Linux-specific).**

**List of process parameters returned for each mode and OS.**

**proc.get[<name>,<user>,<cmdline>,<mode>]**

List of OS processes and their parameters. Can be used for low-level discovery.

**proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]**

| Supported platforms: | Linux, FreeBSD, Windows, OpenBSD, NetBSD. |

If a value cannot be retrieved, for example, because of an error (process already died, lack of permissions, system call failure), -1 will be returned.

Examples:
=> proc.get[zabbix,,process] → list of all Zabbix processes, returns one entry per PID
=> proc.get[java,,thread] → list of all Java processes, returns one entry per thread
=> proc.get[zabbix,,summary] → combined data for Zabbix processes of each type, returns one entry per process name.

See also:
- List of process parameters returned for each mode and OS.
<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
<th>Supported platforms:</th>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory used by process in bytes.</td>
<td>Integer - with mode as max, min, sum</td>
<td>Linux, FreeBSD, Solaris, AIX, Tru64, OpenBSD, NetBSD.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Float - with mode as avg</td>
<td>The memtype parameter is supported only on Linux, FreeBSD, Solaris®, AIX.</td>
<td></td>
</tr>
<tr>
<td>name - process name (default is all processes)</td>
<td></td>
<td>Examples:</td>
<td></td>
</tr>
<tr>
<td>user - user name (default is all users)</td>
<td></td>
<td>=&gt; proc.mem[,root] → memory used by all processes running under the &quot;root&quot; user</td>
<td></td>
</tr>
<tr>
<td>mode - possible values: avg, max, min, sum (default)</td>
<td></td>
<td>=&gt; proc.mem[zabbix_server,zabbix] → memory used by all zabbix_server processes running under the zabbix user</td>
<td></td>
</tr>
<tr>
<td>cmdline - filter by command line (it is a regular expression)</td>
<td></td>
<td>=&gt; proc.mem[,oracle,max,oracleZABBIX] → memory used by the most memory-hungry process running under oracle having oracleZABBIX in its command line</td>
<td></td>
</tr>
<tr>
<td>memtype - type of memory used by process</td>
<td></td>
<td>Note: When several processes use shared memory, the sum of memory used by processes may result in large, unrealistic values.</td>
<td></td>
</tr>
</tbody>
</table>

**Supported platforms:**

- Linux, FreeBSD, Solaris, AIX, Tru64, OpenBSD, NetBSD.

Examples:

- => proc.mem[,root] → memory used by all processes running under the "root" user
- => proc.mem[zabbix_server,zabbix] → memory used by all zabbix_server processes running under the zabbix user
- => proc.mem[,oracle,max,oracleZABBIX] → memory used by the most memory-hungry process running under oracle having oracleZABBIX in its command line

Note: When several processes use shared memory, the sum of memory used by processes may result in large, unrealistic values.

See notes on selecting processes with name and cmdline parameters (Linux-specific).

When this item is invoked from the command line and contains a command line parameter (e.g. using the agent test mode: `zabbix_agentd -t proc.mem[,,,apache2]`), one extra process will be counted, as the agent will count itself.

**proc.num[<name>,<user>,<state>,<cmdline>,<zone>]**
### Item key

| The number of processes. | Integer | **name** - process name (default is all processes) | **user** - user name (default is all users) | **state** - possible values: all (default), disk - uninterruptible sleep, run - running, sleep - interruptible sleep, trace - stopped, zomb - zombie | **cmdline** - filter by command line (it is a regular expression) | **zone** - target zone: current (default), all. This parameter is supported on Solaris only. | **Supported platforms:** Linux, FreeBSD, Solaris, HP-UX, AIX, Tru64, OpenBSD, NetBSD. | The disk and trace state parameters are supported only on Linux, FreeBSD, OpenBSD, NetBSD. | Examples: => proc.num[,mysql] → number of processes running under the mysql user => proc.num[apache2,www-data] → number of apache2 processes running under the www-data user => proc.num[,oracle,sleep,oracleZABBIX] → number of processes in sleep state running under oracle having oracleZABBIX in its command line | See notes on selecting processes with name and cmdline parameters (Linux-specific). |

When this item is invoked from the command line and contains a command line parameter (e.g. using the agent test mode: `zabbix_agentd -t proc.num[,,,apache2]`), one extra process will be counted, as the agent will count itself.

Note that when setting the zone parameter to current (or default) in case the agent has been compiled on a Solaris without zone support, but running on a newer Solaris where zones are supported, then the agent will return NOTSUPPORTED (the agent cannot limit results to only the current zone). However, all is supported in this case.

### Sensor data

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sensor</strong>[device,sensor,&lt;mode&gt;]</td>
<td>Hardware sensor reading.</td>
<td><strong>device</strong> - device name&lt;br&gt;<strong>sensor</strong> - sensor name&lt;br&gt;<strong>mode</strong> - possible values: avg, max, min (if this parameter is omitted, device and sensor are treated verbatim).</td>
<td><strong>Supported platforms:</strong> Linux, OpenBSD.</td>
</tr>
</tbody>
</table>
Reads the hw.sensors MIB on OpenBSD.

Examples:
=> sensor[cpu0,temp0] → temperature of one CPU
=> sensor["cpu[0-2]",temp,avg] → average temperature of the first three CPU’s

Supported on OpenBSD since Zabbix 1.8.4.

System data

<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>system.boottime</strong></td>
<td>System boot time.</td>
<td>Integer (Unix timestamp)</td>
<td></td>
<td>Supported platforms: Linux, FreeBSD, Solaris, MacOS X, OpenBSD, NetBSD.</td>
</tr>
<tr>
<td><strong>system.cpu.discovery</strong></td>
<td>List of detected CPUs/CPU cores. Used for low-level discovery.</td>
<td>JSON object</td>
<td></td>
<td>See supported platforms.</td>
</tr>
<tr>
<td><strong>system.cpu.intr</strong></td>
<td>Device interrupts.</td>
<td>Integer</td>
<td></td>
<td>Supported platforms: Linux, FreeBSD, Solaris, AIX, OpenBSD, NetBSD.</td>
</tr>
<tr>
<td><strong>system.cpu.load[&lt;cpu&gt;,&lt;mode&gt;]</strong></td>
<td>CPU load.</td>
<td>Float</td>
<td>cpu - possible values: all (default), percpu (total load divided by online CPU count) mode - possible values: avg1 (one-minute average, default), avg5, avg15</td>
<td>See supported platforms. The percpu parameter is not supported on Tru64. Example: =&gt; system.cpu.load[ avg5 ]. Supported platforms: Linux, FreeBSD, Solaris, HP-UX, AIX, MacOS X, OpenBSD, NetBSD. The max type parameter is supported only on Linux, FreeBSD, Solaris, MacOS X. Example: =&gt; system.cpu.num Supported platforms: Linux, FreeBSD, Solaris, AIX, OpenBSD, NetBSD.</td>
</tr>
<tr>
<td><strong>system.cpu.num[&lt;type&gt;]</strong></td>
<td>Number of CPUs.</td>
<td>Integer</td>
<td>type - possible values: online (default), max</td>
<td></td>
</tr>
<tr>
<td><strong>system.cpu.switches</strong></td>
<td>Count of context switches.</td>
<td>Integer</td>
<td></td>
<td>Supported platforms: Linux, FreeBSD, Solaris, AIX, OpenBSD, NetBSD.</td>
</tr>
<tr>
<td><strong>system.cpu.util[&lt;cpu&gt;,&lt;type&gt;,&lt;mode&gt;,&lt;logical_or_physical&gt;]</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

187
<table>
<thead>
<tr>
<th>Item key</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| CPU utilization percentage. | Float | **cpu** - `<CPU number>` or all (default)  
**type** - possible values: user (default), idle, nice, system, iowait, interrupt, softirq, steal, guest (on Linux kernels 2.6.24 and above), guest_nice (on Linux kernels 2.6.33 and above).  
**mode** - possible values: avg1 (one-minute average, default), avg5, avg15  
**logical_or_physical** - possible values: logical (default), physical. This parameter is supported on AIX only. |
| system.hostname[<type>,<transform>] | String | System host name.  
**type** (before version 5.4.7 supported on Windows only) - possible values: netbios (default on Windows), host (default on Linux), shorthost (since version 5.4.7; returns part of the hostname before the first dot, a full string for names without dots).  
**transform** (since version 5.4.7) - possible values: none (default), lower (convert to lowercase) |
| system.hw.chassis[<info>] | String | Chassis information.  
**info** - one of full (default), model, serial, type or vendor |
| system.hw.cpu[<cpu>,<info>] | | |

**Supported platforms:**  
Linux, FreeBSD, Solaris, HP-UX, AIX, Tru64, OpenBSD, NetBSD.  

The **nice** type parameter is supported only on Linux, FreeBSD, HP-UX, Tru64, OpenBSD, NetBSD.  
The **iowait** type parameter is supported only on Linux 2.6 and later, Solaris, AIX.  
The **interrupt** type parameter is supported only on Linux 2.6 and later, FreeBSD, OpenBSD.  
The **softirq**, steal, guest, guest_nice type parameters are supported only on Linux 2.6 and later.  
The avg5 and avg15 mode parameters are supported on Linux, FreeBSD, Solaris, HP-UX, AIX, OpenBSD, NetBSD.  

Example:  
`=> system.cpu.util[0,user,avg5]`

Old naming:  
`system.cpu.idleX, system.cpu.niceX, system.cpu.systemX, system.cpu.userX`

---

**system.hostname**

**type** - String  
Possible values: netbios (default on Windows), host (default on Linux), shorthost (since version 5.4.7; returns part of the hostname before the first dot, a full string for names without dots).  
**transform** (since version 5.4.7) - possible values: none (default), lower (convert to lowercase)  

See **supported platforms**.  

The value is acquired by taking **hostname** from the **uname** system API output.  

Examples of returned values:  
`=> system.hostname → linux-w7x1`
`=> system.hostname → example.com`
`=> system.hostname[shorthost] → example`

**Supported platforms:**  
Linux.  
Hewlett-Packard HP Pro 3010 Small Form Factor PC CZXXXXXXX Desktop}

This key depends on the availability of the SMI BIOS table.  
Will try to read the DMI table from sysfs, if sysfs access fails then try reading directly from memory.  

**Root permissions** are required because the value is acquired by reading from sysfs or memory.  

Supported since Zabbix 2.0.
### Item key

| **CPU information.** | **String or integer** | **cpu** - `<CPU number>` or all (default)  
**info** - possible values: full (default), curfreq, maxfreq, model or vendor | **Supported platforms:**  
Linux.  
Example:  
=> system.hw.cpu[0,vendor] → AuthenticAMD  
Gathers info from `/proc/cpuinfo` and `/sys/devices/system/cpu/[cpunum]/cpufreq/cpuinfo_max_freq`.  
If a CPU number and curfreq or maxfreq is specified, a numeric value is returned (Hz).  
Supported since Zabbix 2.0. |
| --- | --- | --- | --- |
| **system.hw.devices[]** | **Text** | **type** (since version 2.0) - pci (default) or usb | **Supported platforms:**  
Linux.  
Example:  
=> system.hw.devices[pci] → 00:00.0  
Host bridge: Advanced Micro Devices [AMD] RS780 Host Bridge  
[...]  
Returns the output of either `lspci` or `lsusb` utility (executed without any parameters).  
Supported since Zabbix 2.0. |
| **system.hw.macaddr[]** | **String** | **interface** - all (default) or a regular expression  
**format** - full (default) or short | **Supported platforms:**  
Linux.  
Lists MAC addresses of the interfaces whose name matches the given interface regular expression (all lists for all interfaces).  
Example:  
=> system.hw.macaddr["eth0",full] →  
eth0 00:11:22:33:44:55  
If `format` is specified as short, interface names and identical MAC addresses are not listed.  
Supported since Zabbix 2.0.  
See supported platforms.  
Must be used as a passive check only.  
Example:  
=> system.localtime[local] → create an item using this key and then use it to display host time in the Clock dashboard widget. |
| **system.localtime[]** | **Integer** - with  
**type as utc** - (default) the time since the Epoch (00:00:00 UTC, January 1, 1970), measured in seconds.  
**String** - with  
**type as local** - the time in the 'yyyy-mm-dd,hh:mm:ss.nnn,+hh:mm' format | **type** - possible values: utc - (default) the time since the Epoch (00:00:00 UTC, January 1, 1970), measured in seconds.  
local - the time in the 'yyyy-mm-dd,hh:mm:ss.nnn,+hh:mm' format | **Supported platforms:**  
See supported platforms.  
Must be used as a passive check only.  
Example:  
=> system.localtime[local] → create an item using this key and then use it to display host time in the Clock dashboard widget. |
| **system.run[]** | **Text** | **command** - the command to be executed |  |
**Item key**

<table>
<thead>
<tr>
<th>Run specified command on the host.</th>
<th>Text result of the command</th>
<th>command - command for execution mode - possible values: wait - wait end of execution (default), nowait - do not wait</th>
<th>See supported platforms.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - with mode as nowait (regardless of command result)</td>
<td></td>
<td></td>
<td>Up to 512KB of data can be returned, including trailing whitespace that is truncated. To be processed correctly, the output of the command must be text. Example: =&gt; system.run[ls -l] → detailed file list of root directory. Note: system.run items are disabled by default. Learn how to enable them.</td>
</tr>
</tbody>
</table>

The return value of the item is standard output together with standard error produced by command. The exit code is not checked.

Empty result is allowed starting with Zabbix 2.4.0. See also: Command execution.

**system.stat**[resource,<type>]
### Item key

| System statistics. | Integer or float | **ent** - number of processor units this partition is entitled to receive (float)  
**kthr,**<type> - information about kernel thread states:  
r - average number of runnable kernel threads (float)  
b - average number of kernel threads placed in the Virtual Memory Manager wait queue (float)  
**memory,**<type> - information about the usage of virtual and real memory:  
avm - active virtual pages (integer)  
fre - size of the free list (integer)  
**page,**<type> - information about page faults and paging activity:  
fi - file page-ins per second (float)  
fo - file page-outs per second (float)  
pi - pages paged in from paging space (float)  
po - pages paged out to paging space (float)  
fr - pages freed (page replacement) (float)  
sr - pages scanned by page-replacement algorithm (float)  
**faults,**<type> - trap and interrupt rate:  
in - device interrupts (float)  
sy - system calls (float)  
cs - kernel thread context switches (float)  
**cpu,**<type> - breakdown of percentage usage of processor time:  
us - user time (float)  
sy - system time (float)  
id - idle time (float)  
wa - idle time during which the system had outstanding disk/NFS I/O request(s) (float)  
**pc** - number of physical processors consumed (float)  
**ec** - the percentage of entitled capacity consumed (float)  
llbusy - indicates the percentage of logical processor(s) utilization that occurred while executing at the user and system level (float)  
app - indicates the available physical processors in the shared pool (float)  
**disk,**<type> - disk statistics:  
bps - indicates the amount of data transferred (read or written) to the drive in bytes per second (integer)  
tps - indicates the number of transfers per second that were issued to the physical disk/tape (float) |

| **system.sw.arch** | String | See supported platforms.  
Example:  
=> system.sw.arch → i686  
Info is acquired from uname() function.  
Supported since Zabbix 2.0. |
**Item key**

- **system.sw.os[<info>]**
  - Operating system information.
  - **info** - possible values: full (default), short or name
  - **Supported platforms:** Linux.
  - **Example:**
    
    $=> system.sw.os[short] → Ubuntu 2.6.35-28.50-generic 2.6.35.11$

  - Info is acquired from (note that not all files and options are present in all distributions):
    - /proc/version (full)
    - /proc/version_signature (short)
    - PRETTY_NAME parameter from /etc/os-release on systems supporting it, or /etc/issue.net (name)

  - Supported since Zabbix 2.0.

- **system.sw.packages[<package>,<manager>,<format>]**
  - Listing of installed packages.
  - **package** - all (default) or a regular expression
  - **manager** - all (default) or a package manager
  - **format** - full (default) or short
  - **Supported platforms:** Linux.

  - Lists (alphabetically) installed packages whose name matches the given package regular expression (all lists them all).

  - **Example:**
    
    $=> system.sw.packages[mini,dpkg,short] → python-minimal, python2.6-minimal, ubuntu-minimal$

  - Supported package managers (executed command):
    - dpkg (dpkg --get-selections)
    - pkgtool (ls /var/log/packages)
    - rpm (rpm -qa)
    - pacman (pacman -Q)

  - If format is specified as full, packages are grouped by package managers (each manager on a separate line beginning with its name in square brackets).
  - If format is specified as short, packages are not grouped and are listed on a single line.

  - Supported since Zabbix 2.0.

- **system.swap.in[<device>,<type>]**
  - Swap in (from device into memory) statistics.
  - **device** - specify device used for swapping (Linux only) or all (default)
  - **type** - possible values:
    - count (number of swapins, default on non-Linux platforms), sectors (sectors swapped in), pages (pages swapped in, default on Linux).
  - Note that pages will only work if device was not specified.

  - **Supported platforms:**
    - Linux, FreeBSD, OpenBSD.

  - The sectors type parameter is supported only on Linux.

  - **Example:**
    
    $=> system.swap.in[.pages]$

  - The source of this information is:
    - /proc/swaps, /proc/partitions, /proc/stat (Linux 2.4)
    - /proc/swaps, /proc/diskstats, /proc/vmstat (Linux 2.6)
Item key

**system.swap.out**<device>,<type>

Swap out (from memory onto device) statistics.

**device** - specify device used for swapping (Linux only) or all (default)

**type** - possible values:
- count (number of swapouts, default on non-Linux platforms), sectors (sectors swapped out), pages (pages swapped out, default on Linux).

Note that pages will only work if device was not specified.

**Supported platforms:**
Linux, FreeBSD, OpenBSD.

Example:
=> system.swap.out[,pages]

The source of this information is:
/proc/swaps, /proc/partitions, /proc/stat (Linux 2.4)
/proc/swaps, /proc/diskstats, /proc/vmstat (Linux 2.6)

**system.swap.size**<device>,<type>

Swap space size in bytes or in percentage from total.

**device** - specify device used for swapping (FreeBSD only) or all (default)

**type** - possible values:
- free (free swap space, default), pfree (free swap space, in percent), pused (used swap space, in percent), total (total swap space), used (used swap space)

Note that pfree, pused are not supported on Windows if swap size is 0.

**Supported platforms:**
Linux, FreeBSD, Solaris, AIX, Tru64, OpenBSD.

Example:
=> system.swap.size[,pfree] → free swap space percentage

If device is not specified Zabbix agent will only take into account swap devices (files), physical memory will be ignored. For example, on Solaris systems swap -s command includes a portion of physical memory and swap devices (unlike swap -l).

See supported platforms.

Example of returned value (Unix):
FreeBSD localhost 4.2-RELEASE FreeBSD 4.2-RELEASE #0: Mon Nov i386

On Unix since Zabbix 2.2.0 the value for this item is obtained with uname() system call. Previously it was obtained by invoking “uname -a”. The value of this item might differ from the output of “uname -a” and does not include additional information that “uname -a” prints based on other sources.

Note that on Windows the item returns OS architecture, whereas on Unix it returns CPU architecture.

**Supported platforms:**
Linux, FreeBSD, Solaris, AIX, MacOS X, OpenBSD, NetBSD.

Support on Tru64 is unknown.

In item configuration, use s or uptime units to get readable values.

**system.uptime**

System uptime in seconds.

**Supported platforms:**
Linux, FreeBSD, Solaris, AIX, MacOS X, OpenBSD, NetBSD.

Support on Tru64 is unknown.

In item configuration, use s or uptime units to get readable values.

**system.users.num**

Number of users.

**Supported platforms:**
All platforms.

Example:
=> system.users.num
## Virtual file system data

### Item key

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>vfs.dev.discovery</td>
<td>JSON object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List of block devices and their type. Used for low-level discovery.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**vfs.dev.read**

Disk read statistics.

- **device** - disk device (default is all 3)
- **type** - possible values: sectors, operations, bytes, sps, ops, bps
- **mode** - possible values: avg1 (one-minute average, default), avg5, avg15

Note: if using an update interval of three hours or more, will always return '0'.

*Supported platforms:*

- Linux, FreeBSD, Solaris, AIX, OpenBSD.

You may use relative device names (for example, sda) as well as an optional /dev/ prefix (for example, /dev/sda).

LVM logical volumes are supported.

Default values of 'type' parameter for different OSES:
- AIX - operations
- FreeBSD - bps
- Linux - sps
- OpenBSD - operations
- Solaris - bytes

**Example:**

```bash
=> vfs.dev.read[.operations]
```

sps, ops and bps on supported platforms is limited to 1024 devices (1023 individual and one for all).
### Disk write statistics.

<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>device</strong></td>
<td>disk device (default is all 3)</td>
</tr>
<tr>
<td><strong>type</strong></td>
<td>possible values: sectors, operations, bytes, sps, ops, bps</td>
</tr>
<tr>
<td><strong>mode</strong></td>
<td>possible values: avg1 (one-minute average, default), avg5, avg15</td>
</tr>
</tbody>
</table>

Note that 'type' parameter support and defaults depend on the platform.

- sps, ops, bps stand for: sectors, operations, bytes per second, respectively.
- This parameter is supported only with `type in: sps, ops, bps`.

**Supported platforms:**

- Linux, FreeBSD, Solaris, AIX, OpenBSD.

- The sectors and sps type parameters are supported only on Linux.
- The ops type parameter is supported only on Linux, FreeBSD.
- The bps type parameter is supported only on FreeBSD.
- The bytes type parameter is supported only on FreeBSD, Solaris, AIX, OpenBSD.
- The mode parameter is supported only on Linux, FreeBSD.

You may use relative device names (for example, sda) as well as an optional /dev/ prefix (for example, /dev/sda).

LVM logical volumes are supported.

Default values of 'type' parameter for different OSes:

- AIX - operations
- FreeBSD - bps
- Linux - sps
- OpenBSD - operations
- Solaris - bytes

Example:

```perl
e => vfs.dev.write[.operations]
```

sps, ops and bps on supported platforms is limited to 1024 devices (1023 individual and one for all).

---

### vfs.dir.count

```perl
vfs.dir.count(dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age>,<max_age>)
```

Example:

```perl
e => vfs.dir.count()
```
| **Item key** | **Integer** | **dir** - absolute path to directory  
**regex_incl** - regular expression describing the name pattern of the entity (file, directory, symbolic link) to include; include all if empty (default value)  
**regex_excl** - regular expression describing the name pattern of the entity (file, directory, symbolic link) to exclude; don’t exclude any if empty (default value)  
**types_incl** - directory entry types to count, possible values:  
file - regular file, dir - subdirectory, sym - symbolic link, sock - socket, bdev - block device, cdev - character device, fifo - FIFO, dev - synonymous with “bdev,cdev”, all - all types (default), i.e. “file,dir,sym,sock,bdev,cdev,fifo”. Multiple types must be separated with comma and quoted.  
**types_excl** - directory entry types (see <types_incl>) to NOT count. If some entry type is in both <types_incl> and <types_excl>, directory entries of this type are NOT counted.  
**max_depth** - maximum depth of subdirectories to traverse. -1 (default) - unlimited, 0 - no descending into subdirectories.  
**min_size** - minimum size (in bytes) for file to be counted. Smaller files will not be counted. Memory suffixes can be used.  
**max_size** - maximum size (in bytes) for file to be counted. Larger files will not be counted. Memory suffixes can be used.  
**min_age** - minimum age (in seconds) of directory entry to be counted. More recent entries will not be counted. Time suffixes can be used.  
**max_age** - maximum age (in seconds) of directory entry to be counted. Entries so old and older will not be counted (modification time). Time suffixes can be used.  
**regex_excl_dir** - regular expression describing the name pattern of the directory to exclude. All content of the directory will be excluded (in contrast to regex_excl) | **See supported platforms.**  
Environment variables, e.g. %APP_HOME%, $HOME and %TEMP% are not supported.  
Pseudo-directories “.” and “..” are never counted.  
Symbolic links are never followed for directory traversal.  
Both regex_incl and regex_excl are being applied to files and directories when calculating entry size, but are ignored when picking subdirectories to traverse (if regex_incl is “(?i)^.+\.zip$” and max_depth is not set, then all subdirectories will be traversed, but only files of type zip will be counted).  
Execution time is limited by the default timeout value in agent configuration (3 sec). Since large directory traversal may take longer than that, no data will be returned and the item will turn unsupported. Partial count will not be returned.  
When filtering by size, only regular files have meaningful sizes. Under Linux and BSD, directories also have non-zero sizes (a few Kb typically). Devices have zero sizes, e.g. the size of /dev/sdal does not reflect the respective partition size. Therefore, when using <min_size> and <max_size>, it is advisable to specify <types_incl> as “file”, to avoid surprises.  
Examples:  
⇒ vfs.dir.count[/dev] - monitors number of devices in /dev (Linux)  
Supported since Zabbix 4.0.0. |
| **vfs.dir.get**[dir,regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age>,<max_age>] |
Item key

<table>
<thead>
<tr>
<th>Directory entry list</th>
<th>JSON</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dir</strong> - absolute path to directory</td>
<td></td>
<td>describing the name pattern of the entity (file, directory, symbolic link) to include; include all if empty (default value)</td>
</tr>
<tr>
<td><strong>regex_incl</strong> - regular expression</td>
<td></td>
<td>describing the name pattern of the entity (file, directory, symbolic link) to include; include all if empty (default value)</td>
</tr>
<tr>
<td><strong>regex_excl</strong> - regular expression</td>
<td></td>
<td>describing the name pattern of the entity (file, directory, symbolic link) to exclude; don’t exclude any if empty (default value)</td>
</tr>
<tr>
<td><strong>types_incl</strong> - directory entry types to list, possible values:</td>
<td></td>
<td>file - regular file, dir - subdirectory, sym - symbolic link, sock - socket, bdev - block device, cdev - character device, fifo - FIFO, dev - synonymous with &quot;bdev,cdev&quot;, all - all types (default), i.e. “file,dir,sym,sock,bdev,cdev,fifo”. Multiple types must be separated with comma and quoted.</td>
</tr>
<tr>
<td><strong>types_excl</strong> - directory entry types (see &lt;types_incl&gt;) to NOT list. If some entry type is in both &lt;types_incl&gt; and &lt;types_excl&gt;, directory entries of this type are NOT listed.</td>
<td></td>
<td>See supported platforms.</td>
</tr>
<tr>
<td><strong>max_depth</strong> - maximum depth of subdirectories to traverse.</td>
<td></td>
<td>-1 (default) - unlimited, 0 - no descending into subdirectories.</td>
</tr>
<tr>
<td><strong>min_size</strong> - minimum size (in bytes) for file to be listed. Smaller files will not be listed.</td>
<td></td>
<td>Memory suffixes can be used.</td>
</tr>
<tr>
<td><strong>max_size</strong> - maximum size (in bytes) for file to be listed. Larger files will not be counted.</td>
<td></td>
<td>Memory suffixes can be used.</td>
</tr>
<tr>
<td><strong>min_age</strong> - minimum age (in seconds) of directory entry to be listed. More recent entries will not be listed.</td>
<td></td>
<td>Time suffixes can be used.</td>
</tr>
<tr>
<td><strong>max_age</strong> - maximum age (in seconds) of directory entry to be listed. Entries so old and older will not be listed (modification time).</td>
<td></td>
<td>Time suffixes can be used.</td>
</tr>
<tr>
<td><strong>regex_excl_dir</strong> - regular expression</td>
<td></td>
<td>describing the name pattern of the directory to exclude. All content of the directory will be excluded (in contrast to regex_excl)</td>
</tr>
</tbody>
</table>

**Examples:**

- `vfs.dir.size(dir,<regex_incl>,<regex_excl>,<mode>,<max_depth>,<regex_excl_dir>)`
### vfs.file.cksum

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>file</code></td>
<td>Full path to file</td>
</tr>
<tr>
<td><code>&lt;mode&gt;</code></td>
<td><code>crc32</code> (default), <code>md5</code>, <code>sha256</code></td>
</tr>
</tbody>
</table>

**Example:**

```
=> vfs.file.cksum[/etc/passwd]
```

**Example of returned values**

```
675436101
9845acf68b73991eb7fd7ee0ded23c44
ae67546e4aac995e5c921042d0cf01f7147703aa42bfbf
```

The file size limit depends on `large file support`.  

The `mode` parameter is supported since Zabbix 6.0.

See supported platforms.

### vfs.file.contents

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>file</code></td>
<td>Full path to file</td>
</tr>
<tr>
<td><code>&lt;encoding&gt;</code></td>
<td>Code page identifier</td>
</tr>
</tbody>
</table>

**Example:**

```
=> vfs.file.contents[/etc/passwd]
```

This item is limited to files no larger than 64 Kbytes.  

Supported since Zabbix 2.0.

See supported platforms.

### vfs.file.exists

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>file</code></td>
<td>Full path to file</td>
</tr>
<tr>
<td><code>&lt;types_incl&gt;</code></td>
<td>† Included types</td>
</tr>
<tr>
<td><code>&lt;types_excl&gt;</code></td>
<td>† Excluded types</td>
</tr>
</tbody>
</table>

**Supported platforms:**

Linux.  
The item may work on other UNIX-like platforms.  

Only directories with at least read permission for zabbix user are calculated.  

With large directories or slow drives this item may time out due to the Timeout setting in agent and server/proxy configuration files. Increase the timeout values as necessary.

**Examples:**

```
⇒ vfs.dir.size[/tmp,log] - calculates size of all files in /tmp which contain ‘log’
⇒ vfs.dir.size[/tmp,log,^.+_.old$] - calculates size of all files in /tmp which contain ‘.old’
```

The file size limit depends on `large file support`.  

The mode parameter is supported since Zabbix 6.0.

See supported platforms.
<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Checks if file exists.** | 0 - not found  
1 - file of the specified type exists |
| **vfs.file.get**[file] | Return JSON object about a file. |
| **vfs.file.md5sum**[file] | MD5 checksum of file. |
| **vfs.file.permissions**[file] | |
Item key

Return a 4-digit string containing the octal number with Unix permissions.

**vfs.file.regexp** [file,regexp,<encoding>,<start line>,<end line>,<output>]

*Find string in a file.*

- **file** - full path to file
- **regexp** - regular expression describing the required pattern
- **encoding** - code page identifier
- **start line** - the number of first line to search (first line of file by default).
- **end line** - the number of last line to search (last line of file by default).
- **output** - an optional output formatting template. The \0 escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an \N (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).

Supported since Zabbix 6.0.

See supported platforms.

Only the first matching line is returned. An empty string is returned if no line matched the expression.

Byte order mark (BOM) is excluded from the output.

Content extraction using the output parameter takes place on the agent.

The start line, end line and output parameters are supported from version 2.2.

Examples:

- `=> vfs.file.regexp[/etc/passwd,zabbix]`
- `=> vfs.file.regexp[/path/to/some/file,“(\[0-9]+)\”3,5,\1]`
- `=> vfs.file.regexp[/etc/passwd,“^zabbix:.:(\[0-9]+)\”\1]` → getting the ID of user zabbix

**vfs.file.regmatch** [file,regexp,<encoding>,<start line>,<end line>]

*Find string in a file.*

- **file** - full path to file
- **regexp** - regular expression describing the required pattern
- **encoding** - code page identifier
- **start line** - the number of first line to search (first line of file by default).
- **end line** - the number of last line to search (last line of file by default).

See supported platforms.

Byte order mark (BOM) is ignored.

The start line and end line parameters are supported from version 2.2.

Example:

- `=> vfs.file.regmatch[/var/log/app.log,error]`

**vfs.file.size** [file,<mode>]

Supported platforms:

Linux. The item may work on other UNIX-like platforms.

Example:

- `=> vfs.file.permissions[/etc/passwd]` → return permissions of /etc/passwd, for example, ‘0644’
### Item key

| File size (in bytes). | Integer | file - full path to file
|                     |         | mode - possible values:
|                     |         | bytes (default) or lines (empty lines are counted, too) |

The file must have read permissions for user zabbix.

Example:

```=> vfs.file.size[/var/log/syslog]```

The file size limit depends on large file support.

The mode parameter is supported since Zabbix 6.0.

### vfs.file.time[file,<mode>]

File time information. Integer (Unix timestamp)

The mode parameter is supported since Zabbix 6.0.

Example:

```=> vfs.file.time[/etc/passwd,modify]```

The file size limit depends on large file support.

### vfs.fs.discovery

List of mounted filesystems and their types. Used for low-level discovery.

Supported platforms:

- Linux, FreeBSD, Solaris, HP-UX, AIX, MacOS X, OpenBSD, NetBSD.

### vfs.fs.get

List of mounted filesystems, their types, disk space and inode statistics. Can be used for low-level discovery.

Supported platforms:

- Linux, FreeBSD, Solaris, HP-UX, AIX, MacOS X, OpenBSD, NetBSD.

Supported since Zabbix 4.4.5.

### vfs.fs.inode[fs,<mode>]

Number or percentage of inodes. Integer - for number

Float - for percentage

See supported platforms.

Example:

```=> vfs.fs.inode[/,pfree]```

### vfs.fs.size[fs,<mode>]

Disk space in bytes or in percentage from total. Integer - for bytes

Float - for percentage

In case of a mounted volume, disk space for local file system is returned.

Example:

```=> vfs.fs.size[/tmp,free]```

Reserved space of a file system is taken into account and not included when using the free mode.

### Virtual memory data

201
### Item key

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>vm.memory.size</strong></td>
<td>&lt;mode&gt;</td>
<td><strong>mode</strong> - possible values:</td>
<td>See supported platforms.</td>
</tr>
<tr>
<td>Memory size in</td>
<td>Integer - for</td>
<td>total (default), active, anon, buffers, cached, exec, file, free, inactive,</td>
<td>The active mode parameter is supported only on FreeBSD, HP-UX, MacOS X, OpenBSD, NetBSD.</td>
</tr>
<tr>
<td>bytes or in</td>
<td>bytes</td>
<td>pinned, shared, slab, wired, used, pused (used, percentage), available,</td>
<td>The anon, exec, file mode parameters are supported only on NetBSD.</td>
</tr>
<tr>
<td>percentage from</td>
<td>Float - for</td>
<td>pavalvable (available, percentage)</td>
<td>The buffers mode parameter is supported only on Linux, FreeBSD, OpenBSD, NetBSD.</td>
</tr>
<tr>
<td>total.</td>
<td>percentage</td>
<td>The mode parameter support is platform-specific (see item comments).</td>
<td>The cached mode parameter is supported only on Linux, FreeBSD, AIX, OpenBSD, NetBSD.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See also additional details for this parameter.</td>
<td>The inactive, wired mode parameters are supported only on FreeBSD, MacOS X, OpenBSD, NetBSD.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The pinned mode parameter is supported only on AIX.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The shared mode parameter is supported only on Linux 2.4, FreeBSD, OpenBSD, NetBSD.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This item accepts three categories of parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1) total - total amount of memory;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2) platform-specific memory types: active, anon, buffers, cached, exec, file,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>free, inactive, pinned, shared, slab, wired;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3) user-level estimates on how much memory is used and available: used,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pused, available, pavalvable.</td>
<td></td>
</tr>
</tbody>
</table>

### Web monitoring data

<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>web.page.get</strong></td>
<td>Get content of</td>
<td>[host,&lt;path&gt;,&lt;port&gt;]</td>
<td><strong>host</strong> - hostname or URL (as scheme://host:port/path, where only</td>
<td>See supported platforms.</td>
</tr>
<tr>
<td></td>
<td>web page</td>
<td></td>
<td>host is mandatory). Missing scheme will be treated as http. If URL is</td>
<td>This item turns unsupported if the resource specified in host does not exist or is unavailable.</td>
</tr>
<tr>
<td></td>
<td>source as text</td>
<td></td>
<td>specified path and port must be empty. Specifying user name/password</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(including</td>
<td></td>
<td>when connecting to servers that require authentication, for example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>headers)</td>
<td></td>
<td><a href="http://user:password@www.example.com">http://user:password@www.example.com</a> is only possible with cURL support.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Punycode is supported in hostnames.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>path</strong> - path to HTML document (default is /)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>port</strong> - port number (default is 80 for HTTP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>=&gt; web.page.get[<a href="http://www.example.com,index.php,80">www.example.com,index.php,80</a>]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>=&gt; web.page.get[<a href="https://www.example.com">https://www.example.com</a>]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>=&gt; web.page.get[<a href="https://blog.example.com/?s=zabbix">https://blog.example.com/?s=zabbix</a>]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>=&gt; web.page.get=localhost:80</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>=&gt; web.page.get[&quot;[::1]/server-status&quot;]</td>
<td></td>
</tr>
</tbody>
</table>
Item key

**web.page.perf**[host,<path>,<port>]

Loading time of full web page (in seconds).

**host** - hostname or URL (as scheme://host:port/path, where only host is mandatory).
- Allowed URL schemes: http, https
- Missing scheme will be treated as http. If URL is specified path and port must be empty. Specifying user name/password when connecting to servers that require authentication, for example: http://user:password@www.example.com
- Punycode is supported in hostnames.
- path - path to HTML document (default is /)
- port - port number (default is 80 for HTTP)

This item turns unsupported if the resource specified in host does not exist or is unavailable.

host can be hostname, domain name, IPv4 or IPv6 address. But for IPv6 address Zabbix agent must be compiled with IPv6 support enabled.

Example:
```python
web.page.perf[www.example.com,index.php,80]
web.page.perf[https://www.example.com]
```

**web.page.regexp**[host,<path>,<port>,regexp,<length>,<output>]

Find string on a web page.

**host** - hostname or URL (as scheme://host:port/path, where only host is mandatory).
- Allowed URL schemes: http, https
- Missing scheme will be treated as http. If URL is specified path and port must be empty. Specifying user name/password when connecting to servers that require authentication, for example: http://user:password@www.example.com
- Punycode is supported in hostnames.
- path - path to HTML document (default is /)
- port - port number (default is 80 for HTTP)
- regexp - regular expression describing the required pattern
- length - maximum number of characters to return
- output - an optional output formatting template. The \0 escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an \N (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).

Content extraction using the output parameter takes place on the agent.

The output parameter is supported from version 2.2.

Example:
```python
web.page.regexp[www.example.com,index.php,80,OK,2]
web.page.regexp[https://www.example.com,,OK,2]
```

**Zabbix metrics**

<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>agent.hostmetadata</strong></td>
<td>Agent host metadata.</td>
<td>String</td>
<td></td>
<td>See supported platforms.</td>
</tr>
<tr>
<td><strong>agent.hostname</strong></td>
<td></td>
<td></td>
<td></td>
<td>Returns the value of HostMetadata or HostMetadataItem parameters, or empty string if none are defined.</td>
</tr>
</tbody>
</table>

Supported since Zabbix 6.0.
<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Agent host name.** | String See supported platforms. | Returns:  
- As passive check - the name of the first host listed in the Hostname parameter of the agent configuration file;  
- As active check - the name of the current hostname. |
| **agent.ping** | Agent - Nothing - unavailable  
availability check.  
1 - available | See supported platforms.  
Use the `nodata()` trigger function to check for host unavailability. |
| **agent.variant** | Variant of Zabbix agent (Zabbix agent or Zabbix agent 2). | See supported platforms.  
Example of returned value:  
1 - Zabbix agent  
2 - Zabbix agent 2 |
| **agent.version** | Version of Zabbix agent. | See supported platforms. |
| **zabbix.stats[<ip>,<port>]** | Return a set of Zabbix server or proxy internal metrics remotely. | JSON object  
**ip** - IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1)  
**port** - port of server/proxy to be remotely queried (default is 10051) | See supported platforms.  
Note that the stats request will only be accepted from the addresses listed in the 'StatsAllowedIP' server/proxy parameter on the target instance.  
A selected set of internal metrics is returned by this item. For details, see Remote monitoring of Zabbix stats. |
| **zabbix.stats[<ip>,<port>,queue,<from>,<to>]** | Return number of monitored items in the queue which are delayed on Zabbix server or proxy remotely. | JSON object  
**ip** - IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1)  
**port** - port of server/proxy to be remotely queried (default is 10051)  
**queue** - constant (to be used as is)  
**from** - delayed by at least (default is 6 seconds)  
**to** - delayed by at most (default is infinity) | See supported platforms.  
Note that the stats request will only be accepted from the addresses listed in the 'StatsAllowedIP' server/proxy parameter on the target instance. |

**Footnotes**

1 A Linux-specific note. Zabbix agent must have read-only access to filesystem /proc. Kernel patches from www.grsecurity.org limit access rights of non-privileged users.

2 `vfs.dev.read()`, `vfs.dev.write()`: Zabbix agent will terminate "stale" device connections if the item values are not accessed for more than 3 hours. This may happen if a system has devices with dynamically changing paths or if a device gets manually removed. Note also that these items, if using an update interval of 3 hours or more, will always return '0'.

3 `vfs.dev.read()`, `vfs.dev.write()`: If default all is used for the first parameter then the key will return summary statistics, including all block devices like sda, sdb and their partitions (sda1, sda2, sdb3...) and multiple devices (MD raid) based on those block devices/partitions and logical volumes (LVM) based on those block devices/partitions. In such cases returned values should be considered only as relative value (dynamic in time) but not as absolute values.

4 SSL (HTTPS) is supported only if agent is compiled with cURL support. Otherwise the item will turn unsupported.
The bytes and errors values are not supported for loopback interfaces on Solaris systems up to and including Solaris 10 6/06 as byte, error and utilization statistics are not stored and/or reported by the kernel. However, if you’re monitoring a Solaris system via net-snmp, values may be returned as net-snmp carries legacy code from the cmu-snmp dated as old as 1997 that, upon failing to read byte values from the interface statistics returns the packet counter (which does exist on loopback interfaces) multiplied by an arbitrary value of 308. This makes the assumption that the average length of a packet is 308 octets, which is a very rough estimation as the MTU limit on Solaris systems for loopback interfaces is 8892 bytes. These values should not be assumed to be correct or even closely accurate. They are guestimates. The Zabbix agent does not do any guess work, but net-snmp will return a value for these fields.

The command line on Solaris, obtained from /proc/pid/psinfo, is limited to 80 bytes and contains the command line as it was when the process was started.

Usage with command-line utilities

Note that when testing or using item keys with zabbix_agentd or zabbix_get from the command line you should consider shell syntax too.

For example, if a certain parameter of the key has to be enclosed in double quotes you have to explicitly escape double quotes, otherwise they will be trimmed by the shell as special characters and will not be passed to the Zabbix utility.

Examples:

$ zabbix_agentd -t 'vfs.dir.count[/var/log,,"file,dir",,0]'

$ zabbix_agentd -t vfs.dir.count[/var/log,,"file,dir\",,0]

Encoding settings

To make sure that the acquired data are not corrupted you may specify the correct encoding for processing the check (e.g. 'vfs.file.contents') in the encoding parameter. The list of supported encodings (code page identifiers) may be found in documentation for libiconv (GNU Project) or in Microsoft Windows SDK documentation for "Code Page Identifiers".

If no encoding is specified in the encoding parameter the following resolution strategies are applied:

- If encoding is not specified (or is an empty string) it is assumed to be UTF-8, the data is processed “as-is”;
- BOM analysis - applicable for items ‘vfs.file.contents’, ‘vfs.file.regexp’, ‘vfs.file.regmatch’. An attempt is made to determine the correct encoding by using the byte order mark (BOM) at the beginning of the file. If BOM is not present - standard resolution (see above) is applied instead.

Troubleshooting agent items

- If used with the passive agent, Timeout value in server configuration may need to be higher than Timeout in the agent configuration file. Otherwise the item may not get any value because the server request to agent timed out first.

Windows Zabbix agent

Overview

The Windows Zabbix agent items are presented in two lists:

- **Shared items** - the item keys that are shared with the UNIX Zabbix agent
- **Windows-specific items** - the item keys that are supported only on Windows:
  - eventlog[]
  - net.if.list
  - perf_counter[]
  - perf_counter_en[]
  - perf_instance.discovery[]
  - proc_info[]
  - registry.data[]
  - registry.get[]
  - service.discovery
  - service.info[]
  - services
  - vm.vmemory.size[]
  - wmi.get[]
  - wmi.getall[]

Windows-specific items sometimes are an approximate counterpart of a similar agent item, for example proc_info, supported on Windows, roughly corresponds to the proc.mem item, not supported on Windows.
Note that all item keys supported by Zabbix agent on Windows are also supported by the new generation Zabbix agent 2. See the additional item keys that you can use with the agent 2 only.

See also: Minimum permissions for Windows items

Shared items

The table below lists Zabbix agent items that are supported on Windows and are shared with the UNIX Zabbix agent:

- The item key is a link to full details in the UNIX Zabbix agent item group
- The item key signature includes only those parameters that are supported on Windows
- Windows-relevant item comments are included

<table>
<thead>
<tr>
<th>Item key</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>agent.hostmetadata</strong></td>
<td></td>
</tr>
<tr>
<td><strong>agent.hostname</strong></td>
<td></td>
</tr>
<tr>
<td><strong>agent.ping</strong></td>
<td></td>
</tr>
<tr>
<td><strong>agent.variant</strong></td>
<td></td>
</tr>
<tr>
<td><strong>agent.version</strong></td>
<td></td>
</tr>
<tr>
<td><strong>log</strong></td>
<td>This item is not supported for Windows Event Logs. The persistent_dir parameter is not supported on Windows.</td>
</tr>
<tr>
<td><strong>log.count</strong></td>
<td>This item is not supported for Windows Event Logs. The persistent_dir parameter is not supported on Windows.</td>
</tr>
<tr>
<td><strong>logrt</strong></td>
<td>This item is not supported for Windows Event Logs. The persistent_dir parameter is not supported on Windows.</td>
</tr>
<tr>
<td><strong>logrt.count</strong></td>
<td>This item is not supported for Windows Event Logs. The persistent_dir parameter is not supported on Windows.</td>
</tr>
<tr>
<td><strong>modbus.get</strong></td>
<td></td>
</tr>
<tr>
<td><strong>net.dns</strong></td>
<td>The ip, timeout and count parameters are ignored on Windows. Some Windows versions (for example, Server 2008) might require the latest updates installed to support non-ASCII characters in interface names. On Windows, the item gets values from 64-bit counters if available. 64-bit interface statistic counters were introduced in Windows Vista and Windows Server 2008. If 64-bit counters are not available, the agent uses 32-bit counters. Multi-byte interface names on Windows are supported. You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. On Windows, the item gets values from 64-bit counters if available. 64-bit interface statistic counters were introduced in Windows Vista and Windows Server 2008. If 64-bit counters are not available, the agent uses 32-bit counters. Multi-byte interface names on Windows are supported. You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. On Windows, the item gets values from 64-bit counters if available. 64-bit interface statistic counters were introduced in Windows Vista and Windows Server 2008. If 64-bit counters are not available, the agent uses 32-bit counters. You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. Checking of LDAP and HTTPS on Windows is only supported by Zabbix agent 2. Checking of LDAP and HTTPS on Windows is only supported by Zabbix agent 2. *This item is supported on Linux by Zabbix agent, but on Windows it is supported only by Zabbix agent 2 on 64-bit Windows. *This item is supported on Linux by Zabbix agent, but on Windows it is supported only by Zabbix agent 2 on 64-bit Windows.</td>
</tr>
<tr>
<td>Item key</td>
<td>Comments</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>proc.num[&lt;name&gt;,&lt;user&gt;]</td>
<td>On Windows, only the name and user parameters are supported.</td>
</tr>
<tr>
<td>system.cpu.discovery</td>
<td></td>
</tr>
<tr>
<td>system.cpu.load[&lt;cpu&gt;,&lt;mode&gt;]</td>
<td>The value is acquired using the Processor Time performance counter. Note that since Windows 8 its Task Manager shows CPU utilization based on the Processor Utility performance counter, while in previous versions it was the Processor Time counter. System is the only type parameter supported on Windows. The value is acquired by either GetComputerName() (for netbios) or gethostname() (for host) functions on Windows.</td>
</tr>
<tr>
<td>system.cpu.num[&lt;type&gt;]</td>
<td></td>
</tr>
<tr>
<td>system.cpu.util[&lt;cpu&gt;,&lt;type&gt;,&lt;mode&gt;]</td>
<td></td>
</tr>
<tr>
<td>system.hostname[&lt;type&gt;,&lt;transform&gt;]</td>
<td>The value is acquired by either GetComputerName() (for netbios) or gethostname() (for host) function on Windows. Examples of returned values: =&gt; system.hostname → WIN-SERV2008-I6 =&gt; system.hostname[host] → Win-Serv2008-I6LonG =&gt; system.hostname[host,lower] → win-serv2008-i6long See also a more detailed description.</td>
</tr>
<tr>
<td>system.localtime[&lt;type&gt;]</td>
<td></td>
</tr>
<tr>
<td>system.run[command,&lt;mode&gt;]</td>
<td></td>
</tr>
<tr>
<td>system.sw.arch</td>
<td></td>
</tr>
<tr>
<td>system.swap.size[&lt;device&gt;,&lt;type&gt;]</td>
<td>The pused type parameter is supported on Linux by Zabbix agent, but on Windows it is supported only by Zabbix agent 2. Note that this key might report incorrect swap space size/percentage on virtualized (VMware ESXi, VirtualBox) Windows platforms. In this case you may use the perf_counter[\700(_Total)\702] key to obtain correct swap space percentage. Example of returned value: Windows ZABBIX-WIN 6.0.6001 Microsoft® Windows Server® 2008 Standard Service Pack 1 x86</td>
</tr>
<tr>
<td>system.uname</td>
<td>On Windows the value for this item is obtained from Win32_OperatingSystem and Win32_Processor WMI classes. The OS name (including edition) might be translated to the user’s display language. On some versions of Windows it contains trademark symbols and extra spaces.</td>
</tr>
<tr>
<td>system.uptime</td>
<td></td>
</tr>
<tr>
<td>vfs.dir.count[dir,&lt;regex_incl&gt;,&lt;regex_excl&gt;,&lt;types_incl&gt;,&lt;types_excl&gt;,&lt;max_depth&gt;,&lt;min_size&gt;,&lt;max_size&gt;,&lt;min_age&gt;,&lt;max_age&gt;]</td>
<td>On Windows directory symlinks are skipped and hard links are counted only once.</td>
</tr>
<tr>
<td>Example:</td>
<td>=&gt; vfs.dir.count[&quot;C:\Users\ADMINI~1\AppData\Local\Temp&quot;] - monitors the number of files in temporary directory</td>
</tr>
<tr>
<td>vfs.dir.get[dir,&lt;regex_incl&gt;,&lt;regex_excl&gt;,&lt;types_incl&gt;,&lt;types_excl&gt;,&lt;max_depth&gt;,&lt;min_size&gt;,&lt;max_size&gt;,&lt;min_age&gt;,&lt;max_age&gt;]</td>
<td>On Windows, directory symlinks are skipped and hard links are counted only once.</td>
</tr>
<tr>
<td>Example:</td>
<td>=&gt; vfs.dir.get[&quot;C:\Users\ADMINI~1\AppData\Local\Temp&quot;] - retrieves the file list in temporary directory</td>
</tr>
<tr>
<td>vfs.dir.size[dir,&lt;regex_incl&gt;,&lt;regex_excl&gt;,&lt;mode&gt;,&lt;max_depth&gt;,&lt;regex_excl_dir&gt;]</td>
<td>On Windows any symlinks are skipped and hard links are taken into account only once.</td>
</tr>
<tr>
<td>vfs.file.cksum[file,&lt;mode&gt;]</td>
<td></td>
</tr>
<tr>
<td>vfs.file.contents[file,&lt;encoding&gt;]</td>
<td></td>
</tr>
<tr>
<td>vfs.file.exists[file,&lt;types_incl&gt;,&lt;types_excl&gt;]</td>
<td>On Windows the double quotes have to be backslash &quot;&quot; escaped and the whole item key enclosed in double quotes when using the command line utility for calling zabbix_get.exe or agent2. Note that the item may turn unsupported on Windows if a directory is searched within a non-existing directory, e.g. vfs.file.exists[C:\no\dir,dir] (where 'no' does not exist). Supported file types on Windows: regular file, directory, symbolic link</td>
</tr>
<tr>
<td>vfs.file.get[file]</td>
<td></td>
</tr>
<tr>
<td>vfs.file.md5sum[file]</td>
<td></td>
</tr>
<tr>
<td>vfs.file.owner[file,&lt;ownertype&gt;,&lt;resulttype&gt;]</td>
<td></td>
</tr>
<tr>
<td>vfs.file.regexp[file,regexp,&lt;encoding&gt;,&lt;start line&gt;,&lt;end line&gt;,&lt;output&gt;]</td>
<td></td>
</tr>
<tr>
<td>vfs.file.regmatch[file,regexp,&lt;encoding&gt;,&lt;start line&gt;,&lt;end line&gt;]</td>
<td></td>
</tr>
<tr>
<td>vfs.file.size[file,&lt;mode&gt;]</td>
<td></td>
</tr>
<tr>
<td>Item key</td>
<td>Comments</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>vfs.file.time[file,&lt;mode&gt;]</code></td>
<td>On Windows XP <code>vfs.file.time[file_change]</code> may be equal to <code>vfs.file.time[file,access]</code>. The <code>{#FSLABEL} macro is supported on Windows since Zabbix 6.0. The </code>{#FSLABEL} macro is supported on Windows since Zabbix 6.0.</td>
</tr>
<tr>
<td><code>vfs.fs.discovery</code></td>
<td></td>
</tr>
<tr>
<td><code>vfs.fs.get</code></td>
<td></td>
</tr>
<tr>
<td><code>vfs.fs.size[fs,&lt;mode&gt;]</code></td>
<td></td>
</tr>
<tr>
<td><code>vm.memory.size[&lt;mode&gt;]</code></td>
<td></td>
</tr>
<tr>
<td><code>web.page.get[&lt;host&gt;,&lt;path&gt;,&lt;port&gt;]</code></td>
<td></td>
</tr>
<tr>
<td><code>web.page.perf[&lt;host&gt;,&lt;path&gt;,&lt;port&gt;]</code></td>
<td></td>
</tr>
<tr>
<td><code>web.page.regexp[&lt;host&gt;,&lt;path&gt;,&lt;port&gt;,regexp,&lt;length&gt;,&lt;output&gt;]</code></td>
<td></td>
</tr>
<tr>
<td><code>zabbix.stats[&lt;ip&gt;,&lt;port&gt;]</code></td>
<td></td>
</tr>
<tr>
<td><code>zabbix.stats[&lt;ip&gt;,&lt;port&gt;,queue,&lt;from&gt;,&lt;to&gt;]</code></td>
<td></td>
</tr>
</tbody>
</table>

### Windows-specific items

The table provides details on the item keys that are supported only by the Windows Zabbix agent.

<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>eventlog</code></td>
<td>Event log monitoring.</td>
<td>Log</td>
<td>name - name of event log</td>
<td>The item must be configured as an active check.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>regexp - regular expression describing the required pattern</td>
<td>Examples:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>severity - regular expression describing severity</td>
<td>=&gt; eventlog[Application]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This parameter accepts the following values: &quot;Information&quot;, &quot;Warning&quot;, &quot;Error&quot;, &quot;Critical&quot;, &quot;Verbose&quot; (since Zabbix 2.2.0 running on Windows Vista or newer)</td>
<td>=&gt; eventlog[Security,&quot;Failure Audit&quot;,^(529</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>source - regular expression describing source identifier (regular expression is supported since Zabbix 2.2.0)</td>
<td>=&gt; eventlog[System,&quot;Warning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>eventid - regular expression describing the event identifier(s)</td>
<td>=&gt; eventlog[System,^(1$)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>maxlines - maximum number of new lines per second the agent will send to Zabbix server or proxy</td>
<td>=&gt; eventlog[System,...@TWOSHORT] - here a custom regular expression named TVOSHORT is referenced (defined as a Result is TRUE type, the expression itself being &quot;1|^70$).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>mode - possible values: all (default), skip - skip processing of older data (affects only newly created items).</td>
<td>Note that the agent is unable to send in events from the &quot;Forwarded events&quot; log.</td>
</tr>
</tbody>
</table>

Not select a non-Log type of information for this item will lead to the loss of local timestamp, as well as log severity and source information.

See also additional information on log monitoring.

### net.if.list
<table>
<thead>
<tr>
<th>Item key</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network interface list</strong> (includes interface type, status, IPv4 address, description).</td>
<td>Supported since Zabbix agent version 1.8.1. Multi-byte interface names supported since Zabbix agent version 1.8.6. Disabled interfaces are not listed. Note that enabling/disabling some components may change their ordering in the Windows interface name. Some Windows versions (for example, Server 2008) might require the latest updates installed to support non-ASCII characters in interface names.</td>
</tr>
<tr>
<td><code>perf_counter[counter,&lt;interval&gt;]</code></td>
<td>Value of any Windows performance counter. <code>counter</code> - path to the counter <code>interval</code> - last N seconds for storing the average value. The interval must be between 1 and 900 seconds (included) and the default value is 1. Performance Monitor can be used to obtain list of available counters. Until version 1.6 this parameter will return correct value only for counters that require just one sample (like <code>System\Threads</code>). It will not work as expected for counters that require more than one sample - like CPU utilization. Since 1.6, <code>interval</code> is used, so the check returns an average value for last &quot;interval&quot; seconds every time. See also: Windows performance counters.</td>
</tr>
<tr>
<td><code>perf_counter_en[counter,&lt;interval&gt;]</code></td>
<td>Value of any Windows performance counter in English. <code>counter</code> - path to the counter in English <code>interval</code> - last N seconds for storing the average value. The interval must be between 1 and 900 seconds (included) and the default value is 1. This item is only supported on Windows Server 2008/Vista and above. You can find the list of English strings by viewing the following registry key: <code>HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\009</code>.</td>
</tr>
<tr>
<td><code>proc_info[process,&lt;attribute&gt;,&lt;type&gt;]</code></td>
<td>Supported since Zabbix agent version 5.0.1.</td>
</tr>
<tr>
<td>Item key</td>
<td>Float</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td><strong>process</strong></td>
<td></td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td></td>
</tr>
<tr>
<td><strong>type</strong></td>
<td></td>
</tr>
</tbody>
</table>

The following attributes are supported:

- vmsize (default) - size of process virtual memory in Kbytes
- wkset - size of process working set (amount of physical memory used by process) in Kbytes
- pf - number of page faults
- ktime - process kernel time in milliseconds
- utime - process user time in milliseconds
- io_read_b - number of bytes read by process during I/O operations
- io_read_op - number of read operation performed by process
- io_write_b - number of bytes written by process during I/O operations
- io_write_op - number of write operation performed by process
- io_other_b - number of bytes transferred by process during operations other than read and write operations
- io_other_op - number of I/O operations performed by process, other than read and write operations
- gdiobj - number of GDI objects used by process
- userobj - number of USER objects used by process

Valid types are:

- avg (default) - average value for all processes named <process>
- min - minimum value among all processes named <process>
- max - maximum value among all processes named <process>
- sum - sum of values for all processes named <process>

Examples:

- `proc_info[iexplore.exe,wkset,sum]` - to get the amount of physical memory taken by all Internet Explorer processes
- `proc_info[iexplore.exe,pf,avg]` - to get the average number of page faults for Internet Explorer processes

Note that on a 64-bit system, a 64-bit Zabbix agent is required for this item to work correctly.

Note: `io_*`, gdiobj and userobj attributes are available only on Windows 2000 and later versions of Windows, not on Windows NT 4.0.

**registry.data**

`[key,<value name>]`
Item key

Return data for the specified value name in the Windows Registry key.

| key | - registry key including the root key; root abbreviations (e.g. HKLM) are allowed
| value name | - registry value name in the key (empty string "" by default). The default value is returned if the value name is not supplied.
| Integer, string or text | (depending on the value type) Supported root abbreviations:

HKCR - HKEY_CLASSES_ROOT
HKCC - HKEY_CURRENT_CONFIG
HKCU - HKEY_CURRENT_USER
HKCUULS - HKEY_CURRENT_USER_LOCAL_SETTINGS
HKLM - HKEY_LOCAL_MACHINE
HKPD - HKEY_PERFORMANCE_DATA
HKPN - HKEY_PERFORMANCE_NLSTEXT
HKPT - HKEY_PERFORMANCE_TEXT
HKU - HKEY_USERS

Keys with spaces must be double-quoted.

Examples:

=> reg-istry.data["HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Windows Error Reporting"] - return the data of the default value of this key

=> reg-istry.data["HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting","EnableZip"] - return the data of the value named "Enable Zip" in this key

This key is supported since Zabbix 6.2.0.

registry.get[key,<mode>,<name regexp>] List of Windows Registry values or keys located at given key.

| key | - registry key including the root key; root abbreviations (e.g. HKLM) are allowed (see comments for registry.data[] to see full list of abbreviations)
| mode | - possible values: values (default), keys
| name regexp | - only discover values with names that match the regexp (default - discover all values). Allowed only with values as mode.

Keys with spaces must be double-quoted.

Examples:

=> reg-istry.get[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall,values,"^DisplayName|DisplayVersion$"] - return the data of the values named "DisplayName" or "DisplayValue" in this key. The JSON will include details of the key, last subkey, value name, value type and value data.

=> reg-istry.get[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall] - return the data of the all values in this key. The JSON will include details of the key, last subkey, value name, value type and value data.

=> reg-istry.get[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall] - return all subkeys of this key. The JSON will include details of the key and last subkey.

This key is supported since Zabbix 6.2.0.

Supported since Zabbix agent version 3.0.

service.discovery List of JSON object Windows services. Used for low-level discovery.

service.info[service,<param>]
<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
<th>Examples:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>service</strong></td>
<td>a real service name or its display name as seen in MMC Services snap-in</td>
<td>=&gt; service.info[SNMPTRAP] - state of the SNMPTRAP service</td>
</tr>
<tr>
<td><strong>param</strong></td>
<td>state (default), displayname, path, user, startup or description</td>
<td>=&gt; service.info[SNMP Trap] - state of the same service, but with display name specified</td>
</tr>
<tr>
<td><strong>state</strong></td>
<td>0-running, 1-paused, 2-start pending, 3-pause pending, 4-continue pending, 5-stop pending, 6-stopped, 7-unknown, 255-no such service</td>
<td>=&gt; service.info[EventLog,startup] - startup type of the EventLog service</td>
</tr>
<tr>
<td><strong>displayname</strong></td>
<td></td>
<td>Items service.info[service,state] and service.info[service] will return the same information.</td>
</tr>
<tr>
<td><strong>type</strong></td>
<td>all (default), automatic, manual or disabled</td>
<td>Note that only with <strong>param</strong> as state this item returns a value for non-existing services (255).</td>
</tr>
<tr>
<td><strong>state</strong></td>
<td>all (default), stopped, started, start_pending, stop_pending, running, continue_pending, pause_pending or paused</td>
<td>This item is supported since Zabbix 3.0.0. It should be used instead of the deprecated service.state[service] item.</td>
</tr>
<tr>
<td><strong>exclude</strong></td>
<td>services to exclude from the result. Excluded services should be listed in double quotes, separated by comma, without spaces.</td>
<td>The exclude parameter is supported since Zabbix 1.8.1.</td>
</tr>
</tbody>
</table>

### Examples:

- => service.info[SNMPTRAP] - state of the SNMPTRAP service
- => service.info[SNMP Trap] - state of the same service, but with display name specified
- => service.info[EventLog,startup] - startup type of the EventLog service
- => service.info[service,state] and service.info[service] will return the same information.
- Note that only with **param** as state this item returns a value for non-existing services (255).
- This item is supported since Zabbix 3.0.0. It should be used instead of the deprecated service.state[service] item.

### The exclude parameter is supported since Zabbix 1.8.1.
Monitoring Windows services

This tutorial provides step-by-step instructions for setting up the monitoring of Windows services. It is assumed that Zabbix server and agent are configured and operational.

Step 1

Get the service name.

You can get that name by going to MMC Services snap-in and bringing up the properties of the service. In the General tab you should see a field called ‘Service name’. The value that follows is the name you will use when setting up an item for monitoring.

For example, if you wanted to monitor the “workstation” service then your service might be: lanmanworkstation.

Step 2

Configure an item for monitoring the service.

The item service.info[service,<param>] retrieves the information about a particular service. Depending on the information you need, specify the param option which accepts the following values: displayname, state, path, user, startup or description. The default value is state if param is not specified (service.info[service]).

The type of return value depends on chosen param: integer for state and startup; character string for displayname, path and user; text for description.
Example:

- **Key**: service.info[lanmanworkstation]
- **Type of information**: Numeric (unsigned)
- **Show value**: select the Windows service state value mapping

Two value maps are available Windows service state and Windows service startup type to map a numerical value to a text representation in the Frontend.

**Discovery of Windows services**

Low-level discovery provides a way to automatically create items, triggers, and graphs for different entities on a computer. Zabbix can automatically start monitoring Windows services on your machine, without the need to know the exact name of a service or create items for each service manually. A filter can be used to generate real items, triggers, and graphs only for services of interest.

### Zabbix agent 2

Zabbix agent 2 supports all item keys supported for Zabbix agent on **Unix** and **Windows**. This page provides details on the additional item keys, which you can use with Zabbix agent 2 only, grouped by the plugin they belong to.

See also: Plugins supplied out-of-the-box

Parameters without angle brackets are mandatory. Parameters marked with angle brackets `< >` are optional.

**Ceph**

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ceph.df.details</td>
<td>Cluster’s data usage and distribution among pools.</td>
<td>JSON object</td>
<td><strong>connString</strong> - URI or session name.</td>
<td><strong>user, password</strong> - Ceph login credentials.</td>
</tr>
<tr>
<td>ceph.osd.stats</td>
<td>Aggregated and per OSD statistics.</td>
<td>JSON object</td>
<td><strong>connString</strong> - URI or session name.</td>
<td><strong>user, password</strong> - Ceph login credentials.</td>
</tr>
<tr>
<td>ceph.osd.discovery</td>
<td>List of discovered OSDs. Used for low-level discovery.</td>
<td>JSON object</td>
<td><strong>connString</strong> - URI or session name.</td>
<td><strong>user, password</strong> - Ceph login credentials.</td>
</tr>
<tr>
<td>ceph.osd.dump</td>
<td>Usage thresholds and statuses of OSDs.</td>
<td>JSON object</td>
<td><strong>connString</strong> - URI or session name.</td>
<td><strong>user, password</strong> - Ceph login credentials.</td>
</tr>
</tbody>
</table>
Key

<table>
<thead>
<tr>
<th>Tests whether a connection to Ceph can be established.</th>
<th>0 - connection is broken (if there is any error presented including AUTH and configuration issues)</th>
<th>connString - URI or session name. user, password - Ceph login credentials.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ceph.pool.discovery [connString, &lt;user&gt;, &lt;apikey&gt;]</td>
<td>JSON object</td>
<td>connString - URI or session name. user, password - Ceph login credentials.</td>
</tr>
<tr>
<td>List of discovered pools. Used for low-level discovery.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ceph.status [connString, &lt;user&gt;, &lt;apikey&gt;]</td>
<td>JSON object</td>
<td>connString - URI or session name. user, password - Ceph login credentials.</td>
</tr>
<tr>
<td>Overall cluster’s status.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Docker

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>docker.container_info [&lt;ID&gt;]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low-level information about a container.</td>
<td>An output of the Container-Inspect API call serialized as JSON</td>
<td>ID - ID or name of the container</td>
<td>The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.</td>
</tr>
<tr>
<td>docker.container_stats [&lt;ID&gt;]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Container resource usage statistics.</td>
<td>An output of the Container-Stats API call and CPU usage percentage serialized as JSON</td>
<td>ID - ID or name of the container</td>
<td>The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.</td>
</tr>
<tr>
<td>docker.containers</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A list of containers.</td>
<td>An output of the ContainerList API call serialized as JSON</td>
<td>-</td>
<td>The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.</td>
</tr>
<tr>
<td>docker.containers.discovery [options&gt;]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Key**

<table>
<thead>
<tr>
<th>A list of containers. Used for low-level discovery.</th>
<th>JSON object</th>
<th><strong>options</strong> - specifies whether all or only running containers should be discovered. Supported values: true - return all containers; false - return only running containers (default).</th>
<th>The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.</th>
</tr>
</thead>
<tbody>
<tr>
<td>docker.data_usage Information about current data usage.</td>
<td>An output of the SystemDataUsage API call serialized as JSON</td>
<td>-</td>
<td>The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.</td>
</tr>
<tr>
<td>docker.images A list of images.</td>
<td>An output of the ImageList API call serialized as JSON</td>
<td>-</td>
<td>The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.</td>
</tr>
<tr>
<td>docker.images.discovery A list of images. Used for low-level discovery.</td>
<td>JSON object</td>
<td>-</td>
<td>The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.</td>
</tr>
<tr>
<td>docker.info System information.</td>
<td>An output of the SystemInfo API call serialized as JSON</td>
<td>-</td>
<td>The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.</td>
</tr>
<tr>
<td>docker.ping Test if a Docker daemon is alive or not.</td>
<td>1 - connection is alive 0 - connection is broken</td>
<td>-</td>
<td>The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.</td>
</tr>
</tbody>
</table>

---

**Memcached**

<table>
<thead>
<tr>
<th><strong>Key</strong></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th><strong>Return value</strong></th>
<th><strong>Parameters</strong></th>
<th><strong>Comments</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>memcached.ping[connString,&lt;user&gt;,&lt;password&gt;] Test if a connection is alive or not.</td>
<td>1 - connection is alive 0 - connection is broken (if there is any error presented including AUTH and configuration issues)</td>
<td>connString - URI or session name.</td>
<td></td>
</tr>
</tbody>
</table>
Get the output of the `STATS` command. JSON - output is serialized as JSON.

**Key**

- **connString**: URI or session name.
- **user, password**: Memcached login credentials.
- **type**: stat type to be returned: items, sizes, slabs or settings (empty by default, returns general statistics).

## MongoDB

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
</table>
| `mongodb.collection.stats[connString,<user>,<password>,<database>,collection]` | JSON object  | **connString**: URI or session name.  
**user, password**: MongoDB login credentials.  
**database**: database name (default: admin).  
**collection**: collection name. |          |
| Returns a variety of storage statistics for a given collection. |             |                            |          |
| `mongodb.collections.discovery[connString,<user>,<password>]` | JSON object  | **connString**: URI or session name.  
**user, password**: MongoDB login credentials. |          |
| Returns a list of discovered collections. Used for low-level discovery. |             |                            |          |
| `mongodb.collections.usage[connString,<user>,<password>]` | JSON object  | **connString**: URI or session name.  
**user, password**: MongoDB login credentials. |          |
| Returns usage statistics for collections. |             |                            |          |
| `mongodb.connpool.stats[connString,<user>,<password>]` | JSON object  | **connString**: URI or session name.  
**user, password**: MongoDB login credentials. |          |
| Returns information regarding the open outgoing connections from the current database instance to other members of the sharded cluster or replica set. |             |                            |          |
| `mongodb.db.stats[connString,<user>,<password>,<database>]` | JSON object  | **connString**: URI or session name.  
**user, password**: MongoDB login credentials.  
**database**: database name (default: admin). |          |
| Returns statistics reflecting a given database system state. |             |                            |          |
| `mongodb.db.discovery[connString,<user>,<password>]` | JSON object  | **connString**: URI or session name.  
**user, password**: MongoDB login credentials. |          |
| Returns a list of discovered databases. Used for low-level discovery. |             |                            |          |
### Key

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Return Value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mongodb.jumbo_chunks.count[connString,&lt;user&gt;,&lt;password&gt;]</code></td>
<td>Returns count of jumbo chunks.</td>
<td>JSON object</td>
<td><code>connString</code> - URI or session name.</td>
<td><code>user, password</code> - MongoDB login credentials.</td>
</tr>
<tr>
<td><code>mongodb.oplog.stats[connString,&lt;user&gt;,&lt;password&gt;]</code></td>
<td>Returns a status of the replica set, using data polled from the oplog.</td>
<td>JSON object</td>
<td><code>connString</code> - URI or session name.</td>
<td><code>user, password</code> - MongoDB login credentials.</td>
</tr>
<tr>
<td><code>mongodb.ping[connString,&lt;user&gt;,&lt;password&gt;]</code></td>
<td>Tests if a connection is alive or not.</td>
<td>1 - connection is alive</td>
<td><code>connString</code> - URI or session name.</td>
<td><code>user, password</code> - MongoDB login credentials.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - connection is broken (if there is any error presented including AUTH and configuration issues).</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>mongodb.rs.config[connString,&lt;user&gt;,&lt;password&gt;]</code></td>
<td>Returns a current configuration of the replica set.</td>
<td>JSON object</td>
<td><code>connString</code> - URI or session name.</td>
<td><code>user, password</code> - MongoDB login credentials.</td>
</tr>
<tr>
<td><code>mongodb.rs.status[connString,&lt;user&gt;,&lt;password&gt;]</code></td>
<td>Returns a replica set status from the point of view of the member where the method is run.</td>
<td>JSON object</td>
<td><code>connString</code> - URI or session name.</td>
<td><code>user, password</code> - MongoDB login credentials.</td>
</tr>
<tr>
<td><code>mongodb.server.status[connString,&lt;user&gt;,&lt;password&gt;]</code></td>
<td>Returns database state.</td>
<td>JSON object</td>
<td><code>connString</code> - URI or session name.</td>
<td><code>user, password</code> - MongoDB login credentials.</td>
</tr>
<tr>
<td><code>mongodb.sh.discovery[connString,&lt;user&gt;,&lt;password&gt;]</code></td>
<td>Returns a list of discovered shards present in the cluster.</td>
<td>JSON object</td>
<td><code>connString</code> - URI or session name.</td>
<td><code>user, password</code> - MongoDB login credentials.</td>
</tr>
</tbody>
</table>

### MQTT

<table>
<thead>
<tr>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td><code>mqtt.get[broker_url&gt;,topic,username&gt;,&lt;password&gt;]</code></td>
</tr>
</tbody>
</table>
### Key

Subscribes to a specific topic or topics (with wildcards) of the provided broker and waits for publications. Depending on topic content.

- **broker_url** - MQTT broker URL (if empty, localhost with port 1883 is used).
- **topic** - MQTT topic (mandatory). Wildcards (+, #) are supported.
- **username, password** - authentication credentials (if required)

The item must be configured as an active check (‘Zabbix agent (active)’ item type).

TLS encryption certificates can be used by saving them into a default location (e.g. /etc/ssl/certs/ directory for Ubuntu). For TLS, use the tls:// scheme.

---

### MySQL

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysql.db.discovery[connString, &lt;username&gt;, &lt;password&gt;]</td>
<td>List of MySQL databases. Used for low-level discovery.</td>
<td>Result of the &quot;show databases&quot; SQL query in LLD JSON format.</td>
<td>connString - URI or session name. username, password - MySQL login credentials.</td>
<td></td>
</tr>
<tr>
<td>mysql.db.size[connString, &lt;username&gt;, &lt;password&gt;, dbName]</td>
<td>Database size in bytes.</td>
<td>Result of the &quot;select coalesce(sum(data_length + index_length),0) as size from information_schema.tables where table_schema=?&quot; SQL query for specific database in bytes.</td>
<td>connString - URI or session name. username, password - MySQL login credentials. dbName - Database name.</td>
<td></td>
</tr>
<tr>
<td>mysql.get_status_variables[connString, &lt;username&gt;, &lt;password&gt;]</td>
<td>Values of global status variables.</td>
<td>Result of the &quot;show global status&quot; SQL query in JSON format.</td>
<td>connString - URI or session name. username, password - MySQL login credentials.</td>
<td></td>
</tr>
<tr>
<td>mysql.ping[connString, &lt;username&gt;, &lt;password&gt;]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---
Key

<table>
<thead>
<tr>
<th>Test if a connection is alive or not.</th>
<th>1 - connection is alive</th>
<th><strong>connString</strong> - URI or session name. <strong>username, password</strong> - MySQL login credentials.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>connString</strong> - URI or session name. <strong>username, password</strong> - MySQL login credentials. <strong>masterHost</strong> - Replication master host name.</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

**Oracle**

Key

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>oracle.diskgroups.stats[connString, &lt;user&gt;, &lt;password&gt;, &lt;service&gt;]</td>
<td>JSON object</td>
<td><strong>connString</strong> - URI or session name. <strong>user, password</strong> - Oracle login credentials. <strong>service</strong> - Oracle service name.</td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>JSON object</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>CDBs info</td>
<td>connString</td>
<td>URI or session name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>user, password</td>
<td>Oracle login credentials.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>service</td>
<td>Oracle service name.</td>
<td></td>
</tr>
<tr>
<td>oracle.custom.query</td>
<td>connString</td>
<td>URI or session name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>user, password</td>
<td>Oracle login credentials.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>service</td>
<td>Oracle service name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>queryName</td>
<td>name of a custom query (must be equal to a name of an sql file without an extension).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>args...</td>
<td>one or several comma-separated arguments to pass to a query.</td>
<td></td>
</tr>
<tr>
<td>oracle.datafiles.stats</td>
<td>connString</td>
<td>URI or session name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>user, password</td>
<td>Oracle login credentials.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>service</td>
<td>Oracle service name.</td>
<td></td>
</tr>
<tr>
<td>oracle.db.discovery</td>
<td>connString</td>
<td>URI or session name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>user, password</td>
<td>Oracle login credentials.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>service</td>
<td>Oracle service name.</td>
<td></td>
</tr>
<tr>
<td>oracle.fra.stats</td>
<td>connString</td>
<td>URI or session name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>user, password</td>
<td>Oracle login credentials.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>service</td>
<td>Oracle service name.</td>
<td></td>
</tr>
<tr>
<td>oracle.instance.info</td>
<td>connString</td>
<td>URI or session name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>user, password</td>
<td>Oracle login credentials.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>service</td>
<td>Oracle service name.</td>
<td></td>
</tr>
<tr>
<td>oracle.pdb.info</td>
<td>connString</td>
<td>URI or session name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>user, password</td>
<td>Oracle login credentials.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>service</td>
<td>Oracle service name.</td>
<td></td>
</tr>
<tr>
<td>oracle.pdb.discovery</td>
<td>connString</td>
<td>URI or session name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>user, password</td>
<td>Oracle login credentials.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>service</td>
<td>Oracle service name.</td>
<td></td>
</tr>
<tr>
<td>oracle.pga.stats</td>
<td>connString</td>
<td>URI or session name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>user, password</td>
<td>Oracle login credentials.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>service</td>
<td>Oracle service name.</td>
<td></td>
</tr>
<tr>
<td>oracle.ping</td>
<td>connString</td>
<td>URI or session name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>user, password</td>
<td>Oracle login credentials.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>service</td>
<td>Oracle service name.</td>
<td></td>
</tr>
<tr>
<td>oracle.proc.stats</td>
<td>connString</td>
<td>URI or session name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>user, password</td>
<td>Oracle login credentials.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>service</td>
<td>Oracle service name.</td>
<td></td>
</tr>
</tbody>
</table>
### Key

<table>
<thead>
<tr>
<th>Oracle Redolog Information</th>
<th>Description</th>
<th>JSON Object</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>oracle.redolog.info[connString,&lt;user&gt;,&lt;password&gt;,&lt;service&gt;]</td>
<td>Log file information from the control file.</td>
<td>connString - URI or session name. user, password - Oracle login credentials. service - Oracle service name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oracle.sga.stats[connString,&lt;user&gt;,&lt;password&gt;,&lt;service&gt;]</td>
<td>SGA statistics.</td>
<td>connString - URI or session name. user, password - Oracle login credentials. service - Oracle service name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oracle.sessions.stats[connString,&lt;user&gt;,&lt;password&gt;,&lt;service&gt;,&lt;lockMaxTime&gt;]</td>
<td>Sessions statistics.</td>
<td>connString - URI or session name. user, password - Oracle login credentials. service - Oracle service name. lockMaxTime - maximum session lock duration in seconds to count the session as a prolongedly locked. Default: 600 seconds.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oracle.sys.metrics[connString,&lt;user&gt;,&lt;password&gt;,&lt;service&gt;,&lt;duration&gt;]</td>
<td>A set of system metric values.</td>
<td>connString - URI or session name. user, password - Oracle login credentials. service - Oracle service name. duration - capturing interval (in seconds) of system metric values. Possible values: 60 — long duration (default), 15 — short duration.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oracle.sys.params[connString,&lt;user&gt;,&lt;password&gt;,&lt;service&gt;]</td>
<td>A set of system parameter values.</td>
<td>connString - URI or session name. user, password - Oracle login credentials. service - Oracle service name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oracle.ts.stats[connString,&lt;user&gt;,&lt;password&gt;,&lt;service&gt;]</td>
<td>Tablespaces statistics.</td>
<td>connString - URI or session name. user, password - Oracle login credentials. service - Oracle service name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oracle.ts.discovery[connString,&lt;user&gt;,&lt;password&gt;,&lt;service&gt;]</td>
<td>List of tablespaces. Used for low-level discovery.</td>
<td>connString - URI or session name. user, password - Oracle login credentials. service - Oracle service name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>oracle.user.info[connString,&lt;user&gt;,&lt;password&gt;,&lt;service&gt;,&lt;username&gt;]</td>
<td>List of tablespaces. Used for low-level discovery.</td>
<td>connString - URI or session name. user, password - Oracle login credentials. service - Oracle service name. username - a username, for which the information is needed. Lowercase usernames are not supported. Default: current user.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### PostgreSQL

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>pgsql.autovacuum.count[uri,&lt;username&gt;,&lt;password&gt;,&lt;dbName&gt;]</td>
<td>The number of autovacuum workers.</td>
<td>uri - URI or session name. username, password - PostgreSQL credentials. dbName - Database name.</td>
<td></td>
</tr>
</tbody>
</table>
pgsql.archive[uri,<username>,<password>, <dbName>]

Information about archived files.

JSON object

uri - URI or session name.
username, password - PostgreSQL credentials.
dbName - Database name.

Returned data are processed by dependent items:
pgsql.archive.count_archived_files - the number of WAL files that have been successfully archived.
pgsql.archive.failed_trying_to_archive - the number of failed attempts for archiving WAL files.
pgsql.archive.count_files_to_archive - the number of files to archive.
pgsql.archive.size_files_to_archive - the size of files to archive.

pgsql.bgwriter[uri,<username>,<password>, <dbName>]

Combined number of checkpoints for the database cluster, broken down by checkpoint type.

JSON object

uri - URI or session name.
username, password - PostgreSQL credentials.
dbName - Database name.

Returned data are processed by dependent items:
pgsql.bgwriter.buffers_alloc - the number of buffers allocated.
pgsql.bgwriter.buffers_backend - the number of buffers written directly by a backend.
pgsql.bgwriter.maxwritten_clean - the number of times the background writer stopped a cleaning scan, because it had written too many buffers.
pgsql.bgwriter.buffers_backend_fsync - the number of times a backend had to execute its own fsync call instead of the background writer.
pgsql.bgwriter.buffers_clean - the number of buffers written by the background writer.
pgsql.bgwriter.buffers_checkpoint - the number of buffers written during checkpoints.
pgsql.bgwriter.checkpoints_timed - the number of scheduled checkpoints that have been performed.
pgsql.bgwriter.checkpoints_req - the number of requested checkpoints that have been performed.
pgsql.bgwriter.checkpoint_write_time - the total amount of time spent in the portion of checkpoint processing where files are written to disk, in milliseconds.
pgsql.bgwriter.sync_time - the total amount of time spent in the portion of checkpoint processing where files are synchronized with disk.

pgsql.cache.hit[uri,<username>,<password>, <dbName>]

PostgreSQL buffer cache hit rate.

Float

uri - URI or session name.
username, password - PostgreSQL credentials.
dbName - Database name.

pgsql.connections[uri,<username>,<password>, <dbName>]

223
### Key

<table>
<thead>
<tr>
<th>Connections by type.</th>
<th>JSON object</th>
<th>Returned data are processed by dependent items:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>uri</strong> - URI or session name.</td>
<td><strong>postgres</strong>: <strong>connections</strong>: <strong>active</strong> - the backend is executing a query. <strong>connections</strong>: <strong>fastpath_function_call</strong> - the backend is executing a fast-path function. <strong>connections</strong>: <strong>idle</strong> - the backend is waiting for a new client command. <strong>connections</strong>: <strong>idle_in_transaction</strong> - the backend is in a transaction, but is not currently executing a query. <strong>connections</strong>: <strong>prepared</strong> - the number of prepared connections. <strong>connections</strong>: <strong>total</strong> - the total number of connections. <strong>connections</strong>: <strong>total_pct</strong> - percentage of total connections in respect to ‘max_connections’ setting of the PostgreSQL server. <strong>connections</strong>: <strong>waiting</strong> - number of connections in a query. <strong>connections</strong>: <strong>idle_in_transaction_aborted</strong> - the backend is in a transaction, but is not currently executing a query and one of the statements in the transaction caused an error.</td>
<td></td>
</tr>
<tr>
<td><strong>username, password</strong> - PostgreSQL credentials.</td>
<td><strong>dbname</strong> - Database name.</td>
<td></td>
</tr>
</tbody>
</table>

```javascript
pgsql.custom.query[uri,<username>,<password>,queryName[,args]]
```

Returns result of a custom query.

```javascript
pgsql.dbstat[uri,<username>,<password>,dbName]
```

Returns database statistics.
<table>
<thead>
<tr>
<th>Key</th>
<th>JSON object</th>
<th>Returned data are processed by dependent items:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collects statistics per database.</td>
<td>JSON object uri</td>
<td>pgsql.dbstat.numbackends[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td>Used for low-level discovery.</td>
<td>username, password - PostgreSQL credentials.</td>
<td>- time spent reading data file blocks by backends in this database, in milliseconds.</td>
</tr>
<tr>
<td></td>
<td>dbName - Database name.</td>
<td>pgsql.dbstat.sum.blk_read_time[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- time spent reading data file blocks by backends in this database, in milliseconds.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.sum.blk_write_time[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- time spent writing data file blocks by backends in this database, in milliseconds.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.sum.checksum_failures[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the number of data page checksum failures detected (or on a shared object), or NULL if data checksums are not enabled. (PostgreSQL version 12 only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.blks_read.rate[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the number of disk blocks read in this database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.deadlocks.rate[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the number of deadlocks detected in this database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.blks_hit.rate[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the number of times disk blocks were found already in the buffer cache, so that a read was not necessary (this only includes hits in the PostgreSQL Pro buffer cache, not the operating system’s file system cache).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.xact_rollback.rate[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the number of transactions in this database that have been rolled back.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.xact_commit.rate[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the number of transactions in this database that have been committed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.tup_updated.rate[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the number of rows updated by queries in this database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.tup_returned.rate[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the number of rows returned by queries in this database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.tup_inserted.rate[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the number of rows inserted by queries in this database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.tup_fetched.rate[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the number of rows fetched by queries in this database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.tup_deleted.rate[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the number of rows deleted by queries in this database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.conflicts.rate[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the number of queries canceled due to conflicts with recovery in this database (the conflicts occur only on standby servers).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.temp_files.rate[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the number of temporary files created by queries in this database. All temporary files are counted, regardless of the log_temp_files settings and reasons for which the temporary file was created (e.g., sorting or hashing).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pgsql.dbstat.temp_bytes.rate[&quot;{#DBNAME}&quot;&quot;]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the total amount of data written to temporary files by queries in this database.</td>
</tr>
</tbody>
</table>
Key

pgsql.dbstat.sum[uri,<username>,<password>,<dbName>]
Summarized data for all databases in a cluster.

JSON object

uri - URI or session name.
username, password - PostgreSQL credentials.
dbName - Database name.

Returned data are processed by the dependent items:

pgsql.dbstat.numbackends - the number of backends currently connected to this database.
pgsql.dbstat.sum.blk_read_time - time spent reading data file blocks by backends in this database, in milliseconds.
pgsql.dbstat.sum.blk_write_time - time spent writing data file blocks by backends in this database, in milliseconds.
pgsql.dbstat.sum.checksum_failures - the number of data page checksum failures detected (or on a shared object), or NULL if data checksums are not enabled (PostgreSQL version 12 only).
pgsql.dbstat.sum.xact_commit - the number of transactions in this database that have been committed.
pgsql.dbstat.sum.conflicts - database statistics about query cancels due to conflict with recovery on standby servers.
pgsql.dbstat.sum.deadlocks - the number of deadlocks detected in this database.
pgsql.dbstat.sum.blks_read - the number of disk blocks read in this database.
pgsql.dbstat.sum.blks_hit - the number of times disk blocks were found already in the buffer cache, so a read was not necessary (only hits in the PostgreSQL Pro buffer cache are included).
pgsql.dbstat.sum.temp_bytes - the total amount of data written to temporary files by queries in this database. Includes data from all temporary files, regardless of the log_temp_files settings and reasons for which the temporary file was created (e.g., sorting or hashing).
pgsql.dbstat.sum.temp_files - the number of temporary files created by queries in this database. All temporary files are counted, regardless of the log_temp_files settings and reasons for which the temporary file was created (e.g., sorting or hashing).
pgsql.dbstat.sum.xact_rollback - the number of rolled-back transactions in this database.
pgsql.dbstat.sum.tup_deleted - the number of rows deleted by queries in this database.
pgsql.dbstat.sum.tup_fetched - the number of rows fetched by queries in this database.
pgsql.dbstat.sum.tup_inserted - the number of rows inserted by queries in this database.
pgsql.dbstat.sum.tup_returned - the number of rows returned by queries in this database.
pgsql.dbstat.sum.tup_updated - the number of rows updated by queries in this database.
pgsql.db.age[uri,<username>,<password>,
dbName]
Age of the oldest FrozenXID of the database. 
Used for low-level discovery.

pgsql.db.bloating_tables[uri,<username>,<password>,
<dbName>]
The number of bloating tables per database. 
Used for low-level discovery.

pgsql.db.discovery[uri,<username>,<password>,
<dbName>]
List of the PostgreSQL databases. 
Used for low-level discovery.

pgsql.db.size[uri,<username>,<password>,
dbName]
Database size in bytes. 
Used for low-level discovery.

pgsql.locks[uri,<username>,<password>,
<dbName>]
Information about granted locks per database. 
Used for low-level discovery.

pgsql.oldest.xid[uri,<username>,<password>,
<dbName>]
Age of the oldest XID.

pgsql.ping[uri,<username>,<password>,
<dbName>]

Returned data are processed by dependent items:
pgsql.locks.shareupdateexclusive[“{#DBNAME}”]
- the number of share update exclusive locks.
pgsql.locks.accessexclusive[“{#DBNAME}”]
- the number of access exclusive locks.
pgsql.locks.accesssshare[“{#DBNAME}”]
- the number of access share locks.
pgsql.locks.exclusive[“{#DBNAME}”]
- the number of exclusive locks.
pgsql.locks.rowexclusive[“{#DBNAME}”]
- the number of row exclusive locks.
pgsql.locks.rowshare[“{#DBNAME}”]
- the number of row share locks.
pgsql.locks.share[“{#DBNAME}”]
- the number of share locks.
pgsql.locks.sharerowexclusive[“{#DBNAME}”]
- the number of share row exclusive locks.
Tests whether a connection is alive or not.

1 - connection is alive
0 - connection is broken (if there is any error presented including AUTH and configuration issues).

pgsql.queries[uri,<username>,<password>,<dbName>,timePeriod] Queries metrics by execution time.

uri - URI or session name.
username, password - PostgreSQL credentials.
dbName - Database name.
timePeriod - execution time limit for count of slow queries (must be a positive integer).

Returned data are processed by dependent items:
pgsql.queries.mro.time_max["{#DBNAME}"] - max maintenance query time.
pgsql.queries.query.time_max["{#DBNAME}"] - max query time.
pgsql.queries.tx.time_max["{#DBNAME}"] - max transaction query time.
pgsql.queries.mro.slow_count["{#DBNAME}"] - slow maintenance query count.
pgsql.queries.query.slow_count["{#DBNAME}"] - slow query count.
pgsql.queries.tx.slow_count["{#DBNAME}"] - slow transaction query count.
pgsql.queries.mro.time_sum["{#DBNAME}"] - sum maintenance query time.
pgsql.queries.query.time_sum["{#DBNAME}"] - sum query time.
pgsql.queries.tx.time_sum["{#DBNAME}"] - sum transaction query time.

This item is supported since Zabbix 6.0.3
**Key**

<table>
<thead>
<tr>
<th>The status of replication.</th>
<th>0 - streaming is down</th>
<th>uri - URI or session name.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 - streaming is up</td>
<td>username, password - PostgreSQL credentials.</td>
</tr>
<tr>
<td></td>
<td>2 - master mode</td>
<td></td>
</tr>
</tbody>
</table>

`pgsql.replication_lag[b|sec][uri,<username>,<password>]`

Replication lag in bytes. Integer uri - URI or session name. username, password - PostgreSQL credentials.

`pgsql.replication_lag.sec[uri,<username>,<password>]`

Replication lag in seconds. Integer uri - URI or session name. username, password - PostgreSQL credentials.

`pgsql.uptime[uri,<username>,<password>,<dbName>]`

PostgreSQL uptime in milliseconds. Float uri - URI or session name. username, password - PostgreSQL credentials. dbName - Database name.

`pgsql.wal.stat[uri,<username>,<password>,<dbName>]`

WAL statistics. JSON object uri - URI or session name. username, password - PostgreSQL credentials. dbName - Database name.

Redis

**Key**

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>redis.config[connString,&lt;password&gt;,&lt;pattern&gt;]</code></td>
<td>GETS THE CONFIGURATION PARAMETERS OF A REDIS INSTANCE THAT MATCH THE PATTERN.</td>
<td>connString - URI or session name. password - Redis password. pattern - glob-style pattern (* by default).</td>
<td>Returned data are processed by dependent items: <code>pgsql.wal.count</code> — the number of WAL files. <code>pgsql.wal.write</code> - the WAL Isn used (in bytes).</td>
</tr>
<tr>
<td><code>redis.info[connString,&lt;password&gt;,&lt;section&gt;]</code></td>
<td>GETS THE OUTPUT OF THE INFO COMMAND.</td>
<td>connString - URI or session name. password - Redis password. section - section of information (default by default).</td>
<td></td>
</tr>
<tr>
<td><code>redis.ping[connString,&lt;password&gt;]</code></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Key

<table>
<thead>
<tr>
<th>Test if a connection is alive or not.</th>
<th>1 - connection is alive</th>
<th>connString - URI or session name. password - Redis password.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 - connection is broken (if there is any error presented including AUTH and configuration issues)</td>
<td></td>
</tr>
</tbody>
</table>

redis.slowlog.count(connString, <password>)

| The number of slow log entries since Redis was started. | Integer | connString - URI or session name. password - Redis password. |

---

### S.M.A.R.T.

<table>
<thead>
<tr>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>smart.attribute.discovery</td>
<td>JSON object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Returns a list of S.M.A.R.T. device attributes.</td>
<td></td>
<td></td>
<td>The following macros and their values are returned: #{NAME}, #{DISKTYPE}, #{ID}, #{ATTRNAME}, #{THRESH}. HDD, SSD and NVME drive types are supported. Drives can be alone or combined in a RAID. #{NAME} will have an add-on in case of RAID, e.g: &quot;{#{NAME}&quot;: &quot;/dev/sdacciss,2&quot;}</td>
</tr>
</tbody>
</table>

smart.disk.discovery

<table>
<thead>
<tr>
<th>Returns a list of S.M.A.R.T. devices.</th>
<th>JSON object</th>
</tr>
</thead>
</table>

smart.disk.get(<path>, <raid_type>)

| Returns all available properties of S.M.A.R.T. devices. | JSON object | path (since Zabbix 6.0.4) - disk path, the {#PATH} macro may be used as a value raid_type (since Zabbix 6.0.4) - RAID type, the {#RAID} macro may be used as a value | HDD, SSD and NVME drive types are supported. Drives can be alone or combined in a RAID. The data includes smartctl version and call arguments, and additional fields: disk_name - holds the name with the required add-ons for RAID discovery, e.g: "{disk_name": "/dev/sdacciss,2"} disk_type - holds the disk type HDD, SSD, or NVME, e.g: "{disk_type": "ssd"} If no parameters are specified, the item will return information about all disks. |

---

Systemd

231
<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>systemd.unit.get</strong>[unit name, &lt;interface&gt;]</td>
<td>JSON object</td>
<td><strong>unit name</strong> - unit name (you may want to use the {#UNIT.NAME} macro in item prototype to discover the name)</td>
<td>This item is supported on Linux platform only.</td>
</tr>
<tr>
<td></td>
<td>Returns all properties of a systemd unit.</td>
<td></td>
<td><strong>interface</strong> - unit interface type, possible values: Unit (default), Service, Socket, Device, Mount, Automount, Swap, Target, Path</td>
<td>LoadState, ActiveState and UnitFileState for Unit interface are returned as text and integer: &quot;ActiveState&quot;: {&quot;state&quot;: 1, &quot;text&quot;: &quot;active&quot;}</td>
</tr>
<tr>
<td></td>
<td><strong>systemd.unit.info</strong>[unit name, &lt;property&gt;, &lt;interface&gt;]</td>
<td>String</td>
<td><strong>unit name</strong> - unit name (you may want to use the {#UNIT.NAME} macro in item prototype to discover the name)</td>
<td>This item allows to retrieve a specific property from specific type of interface as described in dbus API.</td>
</tr>
<tr>
<td></td>
<td>Systemd unit information.</td>
<td></td>
<td><strong>property</strong> - unit property (e.g. ActiveState (default), LoadState, Description)</td>
<td>This item is supported on Linux platform only.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>interface</strong> - unit interface type (e.g. Unit (default), Socket, Service)</td>
<td>Examples:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>=&gt; systemd.unit.info[&quot;{#UNIT.NAME}&quot;] - collect active state (active, reloading, inactive, failed, activating, deactivating) info on discovered systemd units</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>=&gt; systemd.unit.info[&quot;{#UNIT.NAME}&quot;, LoadState] - collect load state info on discovered systemd units</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>=&gt; systemd.unit.info[mysqld.service, Id] - retrieve service technical name (mysqld.service)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>=&gt; systemd.unit.info[mysqld.service, Description] - retrieve service description (MySQL Server)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>=&gt; systemd.unit.info[mysqld.service, ActiveEnterTimestamp] - retrieve the last time the service entered the active state (1562565036283903)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>=&gt; systemd.unit.info[dbus.socket, NConnections, Socket] - collect the number of connections from this socket unit</td>
</tr>
<tr>
<td></td>
<td><strong>systemd.unit.discovery</strong>[&lt;type&gt;]</td>
<td>JSON object</td>
<td><strong>type</strong> - possible values: all, automount, device, mount, path, service (default), socket, swap, target</td>
<td>This item is supported on Linux platform only.</td>
</tr>
</tbody>
</table>

**Web certificate**

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>web.certificate.get</strong>[hostname, &lt;port&gt;, &lt;address&gt;]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2 SNMP agent

Overview

You may want to use SNMP monitoring on devices such as printers, network switches, routers or UPS that usually are SNMP-enabled and on which it would be impractical to attempt setting up complete operating systems and Zabbix agents.

To be able to retrieve data provided by SNMP agents on these devices, Zabbix server must be initially configured with SNMP support.

SNMP checks are performed over the UDP protocol only.

Zabbix server and proxy daemons query SNMP devices for multiple values in a single request. This affects all kinds of SNMP items (regular SNMP items, SNMP items with dynamic indexes, and SNMP low-level discovery) and should make SNMP processing much more efficient. See the bulk processing section for technical details on how it works internally. Bulk requests can also be disabled for devices that cannot handle them properly using the “Use bulk requests” setting for each interface.

Zabbix server and proxy daemons log lines similar to the following if they receive an incorrect SNMP response:

SNMP response from host "gateway" does not contain all of the requested variable bindings

While they do not cover all the problematic cases, they are useful for identifying individual SNMP devices for which bulk requests should be disabled.

Zabbix server/proxy will always retry at least one time after an unsuccessful query attempt: either through the SNMP library’s retrying mechanism or through the internal bulk processing mechanism.

If monitoring SNMPv3 devices, make sure that msgAuthoritativeEngineID (also known as snmpEngineID or “Engine ID”) is never shared by two devices. According to RFC 2571 (section 3.1.1.1) it must be unique for each device.

RFC3414 requires the SNMPv3 devices to persist their engineBoots. Some devices do not do that, which results in their SNMP messages being discarded as outdated after being restarted. In such situation, SNMP cache needs to be manually cleared on a server/proxy (by using -R snmp_cache_reload) or the server/proxy needs to be restarted.

Configuring SNMP monitoring

To start monitoring a device through SNMP, the following steps have to be performed:

Step 1

Find out the SNMP string (or OID) of the item you want to monitor.

To get a list of SNMP strings, use the snmpwalk command (part of net-snmp software which you should have installed as part of the Zabbix installation) or equivalent tool:

shell> snmpwalk -v 2c -c public <host IP> .
As ‘2c’ here stands for SNMP version, you may also substitute it with ‘1’, to indicate SNMP Version 1 on the device.

This should give you a list of SNMP strings and their last value. If it doesn’t then it is possible that the SNMP ‘community’ is different from the standard ‘public’ in which case you will need to find out what it is.

You can then go through the list until you find the string you want to monitor, e.g. if you wanted to monitor the bytes coming in to your switch on port 3 you would use the IF-MIB::ifInOctets.3 string from this line:

```
IF-MIB::ifInOctets.3 = Counter32: 3409739121
```

You may now use the `snmpget` command to find out the numeric OID for ‘IF-MIB::ifInOctets.3’:

```
shell> snmpget -v 2c -c public -On 10.62.1.22 IF-MIB::ifInOctets.3
```

Note that the last number in the string is the port number you are looking to monitor. See also: Dynamic indexes.

This should give you something like the following:

```
.1.3.6.1.2.1.2.2.1.10.3 = Counter32: 3472126941
```

Again, the last number in the OID is the port number.

3COM seem to use port numbers in the hundreds, e.g. port 1 = port 101, port 3 = port 103, but Cisco use regular numbers, e.g. port 3 = 3.

Some of the most used SNMP OIDs are translated automatically to a numeric representation by Zabbix.

In the last example above value type is “Counter32”, which internally corresponds to ASN_COUNTER type. The full list of supported types is ASN_COUNTER, ASN_COUNTER64, ASN_UINT64, ASN_UNSIGNED64, ASN_INTEGER, ASN_INTEGER64, ASN_FLOAT, ASN_DOUBLE, ASN_TIMETICKS, ASN_GAUGE, ASN_IPADDRESS, ASN_OCTET_STR and ASN_OBJECT_ID (since 2.2.8, 2.4.3). These types roughly correspond to “Counter32”, “Counter64”, “UInteger32”, “Integer”, “Float”, “Double”, “Timeticks”, “Gauge32”, “IpAddress”, “OCTET STRING”, “OBJECT IDENTIFIER” in `snmpget` output, but might also be shown as “STRING”, “Hex-STRING”, “OID” and other, depending on the presence of a display hint.

**Step 2**

Create a host corresponding to a device.

Add an SNMP interface for the host:

- Enter the IP address/DNS name and port number
• Select the SNMP version from the dropdown
• Add interface credentials depending on the selected SNMP version:
  – SNMPv1, v2 require only the community (usually ‘public’)
  – SNMPv3 requires more specific options (see below)
• Leave the Use bulk requests checkbox marked to allow bulk processing of SNMP requests

**SNMPv3 parameter** | **Description**
---|---
Context name | Enter context name to identify item on SNMP subnet. Context name is supported for SNMPv3 items since Zabbix 2.2. User macros are resolved in this field.
Security name | Enter security name. User macros are resolved in this field.
Security level | Select security level:
  - noAuthNoPriv - no authentication nor privacy protocols are used
  - AuthNoPriv - authentication protocol is used, privacy protocol is not
  - AuthPriv - both authentication and privacy protocols are used
Authentication protocol, Authentication passphrase | Select authentication protocol - MD5, SHA1, SHA224, SHA256, SHA384 or SHA512. Enter authentication passphrase. User macros are resolved in this field.
Privacy protocol, Privacy passphrase | Select privacy protocol - DES, AES128, AES192, AES256, AES192C (Cisco) or AES256C (Cisco). Enter privacy passphrase. User macros are resolved in this field.

In case of wrong SNMPv3 credentials (security name, authentication protocol/passphrase, privacy protocol) Zabbix receives an ERROR from net-snmp, except for wrong Privacy passphrase in which case Zabbix receives a TIMEOUT error from net-snmp.

Changes in Authentication protocol, Authentication passphrase, Privacy protocol or Privacy passphrase, made without changing the Security name, will take effect only after the cache on a server/proxy is manually cleared (by using -R snmp_cache_reload) or the server/proxy is restarted. In cases, where Security name is also changed, all parameters will be updated immediately.

You can use one of the provided SNMP templates (Template SNMP Device and others) that will automatically add a set of items. However, the template may not be compatible with the host. Click on Add to save the host.

**Step 3**
Create a monitor.

So, now go back to Zabbix and click on Items for the SNMP host you created earlier. Depending on whether you used a template or not when creating your host, you will have either a list of SNMP items associated with your host or just an empty list. We will work on the assumption that you are going to create the item yourself using the information you have just gathered using snmpwalk and snmpget, so click on Create item. In the new item form:

• Enter the item name
• Change the ‘Type’ field to ‘SNMP agent’
• Enter the ‘Key’ as something meaningful, e.g. SNMP-InOctets-Bps
• Make sure the ‘Host interface’ field has your switch/router in it
• Enter the textual or numeric OID that you retrieved earlier into the ‘SNMP OID’ field, for example: .1.3.6.1.2.1.2.2.1.10.3
• Set the ‘Type of information’ to Numeric (float)
• Enter an ‘Update interval’ and ‘History storage’ period if you want them to be different from the default
• In the Preprocessing tab, add a Change per second step (important, otherwise you will get cumulative values from the SNMP device instead of the latest change). Choose a custom multiplier if you want one.
All mandatory input fields are marked with a red asterisk.

Now save the item and go to Monitoring → Latest data for your SNMP data!

Example 1

General example:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OID</strong></td>
<td>1.2.3.45.6.7.8.0 (or .1.2.3.45.6.7.8.0)</td>
</tr>
</tbody>
</table>
| **Key**   | <Unique string to be used as reference to triggers> For example, “my_param”.

Note that OID can be given in either numeric or string form. However, in some cases, string OID must be converted to numeric representation. Utility snmpget may be used for this purpose:

```
shell> snmpget -On localhost public enterprises.ucdavis.memory.memTotalSwap.0
```

Monitoring of SNMP parameters is possible if --with-net-snmp flag was specified while configuring Zabbix sources.

Example 2

Monitoring of uptime:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OID</strong></td>
<td>MIB::sysUpTime.0</td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td>router.uptime</td>
</tr>
<tr>
<td><strong>Value type</strong></td>
<td>Float</td>
</tr>
<tr>
<td><strong>Units</strong></td>
<td>uptime</td>
</tr>
<tr>
<td><strong>Multiplier</strong></td>
<td>0.01</td>
</tr>
</tbody>
</table>

Internal workings of bulk processing

Starting from 2.2.3 Zabbix server and proxy query SNMP devices for multiple values in a single request. This affects several types of SNMP items:

- regular SNMP items
- SNMP items with dynamic indexes
- SNMP low-level discovery rules

All SNMP items on a single interface with identical parameters are scheduled to be queried at the same time. The first two types of items are taken by pollers in batches of at most 128 items, whereas low-level discovery rules are processed individually, as before.

On the lower level, there are two kinds of operations performed for querying values: getting multiple specified objects and walking an OID tree.
For “getting”, a GetRequest-PDU is used with at most 128 variable bindings. For “walking”, a GetNextRequest-PDU is used for SNMPv1 and GetBulkRequest with “max-repetitions” field of at most 128 is used for SNMPv2 and SNMPv3.

Thus, the benefits of bulk processing for each SNMP item type are outlined below:

- regular SNMP items benefit from “getting” improvements;
- SNMP items with dynamic indexes benefit from both “getting” and “walking” improvements: “getting” is used for index verification and “walking” for building the cache;
- SNMP low-level discovery rules benefit from “walking” improvements.

However, there is a technical issue that not all devices are capable of returning 128 values per request. Some always return a proper response, but others either respond with a “tooBig(1)” error or do not respond at all once the potential response is over a certain limit.

In order to find an optimal number of objects to query for a given device, Zabbix uses the following strategy. It starts cautiously with querying 1 value in a request. If that is successful, it queries 2 values in a request. If that is successful again, it queries 3 values in a request and continues similarly by multiplying the number of queried objects by 1.5, resulting in the following sequence of request sizes: 1, 2, 3, 4, 6, 9, 13, 19, 28, 42, 63, 94, 128.

However, once a device refuses to give a proper response (for example, for 42 variables), Zabbix does two things.

First, for the current item batch it halves the number of objects in a single request and queries 21 variables. If the device is alive, then the query should work in the vast majority of cases, because 28 variables were known to work and 21 is significantly less than that. However, if that still fails, then Zabbix falls back to querying values one by one. If it still fails at this point, then the device is definitely not responding and request size is not an issue.

The second thing Zabbix does for subsequent item batches is it starts with the last successful number of variables (28 in our example) and continues incrementing request sizes by 1 until the limit is hit. For example, assuming the largest response size is 32 variables, the subsequent requests will be of sizes 29, 30, 31, 32, and 33. The last request will fail and Zabbix will never issue a request of size 33 again. From that point on, Zabbix will query at most 32 variables for this device.

If large queries fail with this number of variables, it can mean one of two things. The exact criteria that a device uses for limiting response size cannot be known, but we try to approximate that using the number of variables. So the first possibility is that this number of variables is around the device’s actual response size limit in the general case: sometimes response is less than the limit, sometimes it is greater than that. The second possibility is that a UDP packet in either direction simply got lost. For these reasons, if Zabbix gets a failed query, it reduces the maximum number of variables to try to get deeper into the device’s comfortable range, but (starting from 2.2.8) only up to two times.

In the example above, if a query with 32 variables happens to fail, Zabbix will reduce the count to 31. If that happens to fail, too, Zabbix will reduce the count to 30. However, Zabbix will not reduce the count below 30, because it will assume that further failures are due to UDP packets getting lost, rather than the device’s limit.

If, however, a device cannot handle bulk requests properly for other reasons and the heuristic described above does not work, since Zabbix 2.4 there is a “Use bulk requests” setting for each interface that allows to disable bulk requests for that device.

1 Dynamic indexes

Overview

While you may find the required index number (for example, of a network interface) among the SNMP OIDs, sometimes you may not completely rely on the index number always staying the same.

Index numbers may be dynamic - they may change over time and your item may stop working as a consequence.

To avoid this scenario, it is possible to define an OID which takes into account the possibility of an index number changing.

For example, if you need to retrieve the index value to append to ifInOctets that corresponds to the GigabitEthernet0/1 interface on a Cisco device, use the following OID:

ifInOctets["index","ifDescr","GigabitEthernet0/1"]

The syntax

A special syntax for OID is used:

```<OID of data>"index","<base OID of index>","<string to search for>"```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OID of data</td>
<td>Main OID to use for data retrieval on the item.</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>Method of processing. Currently one method is supported:</td>
</tr>
<tr>
<td>base OID of index</td>
<td>This OID will be looked up to get the index value corresponding to the string.</td>
</tr>
<tr>
<td>string to search for</td>
<td>The string to use for an exact match with a value when doing lookup. Case sensitive.</td>
</tr>
</tbody>
</table>

**Example**

Getting memory usage of apache process.

If using this OID syntax:

```
HOST-RESOURCES-MIB::hrSWRunPerfMem["index","HOST-RESOURCES-MIB::hrSWRunPath", "/usr/sbin/apache2"]
```

the index number will be looked up here:

```...```

```
HOST-RESOURCES-MIB::hrSWRunPath.5376 = STRING: "/sbin/getty"
HOST-RESOURCES-MIB::hrSWRunPath.5377 = STRING: "/sbin/getty"
HOST-RESOURCES-MIB::hrSWRunPath.5388 = STRING: "/usr/sbin/apache2"
HOST-RESOURCES-MIB::hrSWRunPath.5389 = STRING: "/sbin/sshd"
```

```...
```

Now we have the index, 5388. The index will be appended to the data OID in order to receive the value we are interested in:

```
HOST-RESOURCES-MIB::hrSWRunPerfMem.5388 = INTEGER: 31468 KBytes
```

**Index lookup caching**

When a dynamic index item is requested, Zabbix retrieves and caches whole SNMP table under base OID for index, even if a match would be found sooner. This is done in case another item would refer to the same base OID later - Zabbix would look up index in the cache, instead of querying the monitored host again. Note that each poller process uses separate cache.

In all subsequent value retrieval operations only the found index is verified. If it has not changed, value is requested. If it has changed, cache is rebuilt - each poller that encounters a changed index walks the index SNMP table again.

### 2 Special OIDs

Some of the most used SNMP OIDs are translated automatically to a numeric representation by Zabbix. For example, `ifIndex` is translated to `1.3.6.1.2.1.2.1.1`. `ifIndex.0` is translated to `1.3.6.1.2.1.2.1.1.0`.

The table contains list of the special OIDs.

<table>
<thead>
<tr>
<th>Special OID</th>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifIndex</td>
<td>1.3.6.1.2.1.2.1.1</td>
<td>A unique value for each interface.</td>
</tr>
<tr>
<td>ifDescr</td>
<td>1.3.6.1.2.1.2.1.2</td>
<td>A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the hardware interface.</td>
</tr>
<tr>
<td>ifType</td>
<td>1.3.6.1.2.1.2.1.3</td>
<td>The type of interface, distinguished according to the physical/link protocol(s) immediately ‘below’ the network layer in the protocol stack.</td>
</tr>
<tr>
<td>ifMtu</td>
<td>1.3.6.1.2.1.2.1.4</td>
<td>The size of the largest datagram which can be sent / received on the interface, specified in octets.</td>
</tr>
<tr>
<td>ifSpeed</td>
<td>1.3.6.1.2.1.2.1.5</td>
<td>An estimate of the interface’s current bandwidth in bits per second.</td>
</tr>
<tr>
<td>ifPhysAddress</td>
<td>1.3.6.1.2.1.2.1.6</td>
<td>The interface’s address at the protocol layer immediately ‘below’ the network layer in the protocol stack.</td>
</tr>
<tr>
<td>ifAdminStatus</td>
<td>1.3.6.1.2.1.2.1.7</td>
<td>The current administrative state of the interface.</td>
</tr>
<tr>
<td>ifOperStatus</td>
<td>1.3.6.1.2.1.2.1.8</td>
<td>The current operational state of the interface.</td>
</tr>
<tr>
<td>ifInOctets</td>
<td>1.3.6.1.2.1.2.1.10</td>
<td>The total number of octets received on the interface, including framing characters.</td>
</tr>
<tr>
<td>ifInUcastPkts</td>
<td>1.3.6.1.2.1.2.1.11</td>
<td>The number of subnetwork-unicast packets delivered to a higher-layer protocol.</td>
</tr>
<tr>
<td>ifInNUcastPkts</td>
<td>1.3.6.1.2.1.2.1.12</td>
<td>The number of non-unicast (i.e., subnetwork- broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol.</td>
</tr>
<tr>
<td>Special OID</td>
<td>Identifier</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ifInDiscards</td>
<td>1.3.6.1.2.1.2.1.13</td>
<td>The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space.</td>
</tr>
<tr>
<td>ifInErrors</td>
<td>1.3.6.1.2.1.2.1.14</td>
<td>The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol.</td>
</tr>
<tr>
<td>ifInUnknownProtos</td>
<td>1.3.6.1.2.1.2.1.15</td>
<td>The number of packets received via the interface which were discarded because of an unknown or unsupported protocol.</td>
</tr>
<tr>
<td>ifOutOctets</td>
<td>1.3.6.1.2.1.2.1.16</td>
<td>The total number of octets transmitted out of the interface, including framing characters.</td>
</tr>
<tr>
<td>ifOutUcastPkts</td>
<td>1.3.6.1.2.1.2.1.17</td>
<td>The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent.</td>
</tr>
<tr>
<td>ifOutNUcastPkts</td>
<td>1.3.6.1.2.1.2.1.18</td>
<td>The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent.</td>
</tr>
<tr>
<td>ifOutDiscards</td>
<td>1.3.6.1.2.1.2.1.19</td>
<td>The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space.</td>
</tr>
<tr>
<td>ifOutErrors</td>
<td>1.3.6.1.2.1.2.1.20</td>
<td>The number of outbound packets that could not be transmitted because of errors.</td>
</tr>
<tr>
<td>ifOutQLen</td>
<td>1.3.6.1.2.1.2.1.21</td>
<td>The length of the output packet queue (in packets).</td>
</tr>
</tbody>
</table>

3 MIB files

Introduction

MIB stands for a Management Information Base. MIB files allow you to use textual representation of the OID (Object Identifier).

For example,

```plaintext
ifHCOutOctets
```

is textual representation of OID

```plaintext
1.3.6.1.2.1.31.1.1.1.10
```

You can use either, when monitoring SNMP devices with Zabbix, but if you feel more comfortable when using textual representation you have to install MIB files.

Installing MIB files

On Debian-based systems:

```bash
# apt install snmp-mibs-downloader
# download-mibs
```

On RedHat-based systems:

```bash
# yum install net-snmp-libs
```

Enabling MIB files

On RedHat-based systems the mib files should be enabled by default. On Debian-based systems you have to edit file `/etc/snmp/snmp.conf` and comment out the line that says `mibs:

```plaintext
# As the snmp packages come without MIB files due to license reasons, loading # of MIBs is disabled by default. If you added the MIBs you can re-enable # loading them by commenting out the following line.
#mibs:
```

Testing MIB files

Testing snmp MIBs can be done using `snmpwalk` utility. If you don’t have it installed, use the following instructions.
On Debian-based systems:

```
# apt install snmp
```

On RedHat-based systems:

```
# yum install net-snmp-utils
```

After that, the following command must not give error when you query a network device:

```
$ snmpwalk -v 2c -c public <NETWORK DEVICE IP> ifInOctets
```

```text
IF-MIB::ifInOctets.1 = Counter32: 176137634
IF-MIB::ifInOctets.2 = Counter32: 0
IF-MIB::ifInOctets.3 = Counter32: 240375057
IF-MIB::ifInOctets.4 = Counter32: 220893420
[…]
```

Using MIBs in Zabbix

The most important to keep in mind is that Zabbix processes do not get informed of the changes made to MIB files. So after every change you must restart Zabbix server or proxy, e. g.:

```
# service zabbix-server restart
```

After that, the changes made to MIB files are in effect.

Using custom MIB files

There are standard MIB files coming with every GNU/Linux distribution. But some device vendors provide their own.

Let’s say, you would like to use `CISCO-SMI` MIB file. The following instructions will download and install it:

```
# mkdir -p /usr/local/share/snmp/mibs
# grep -q '^mibdirs +/usr/local/share/snmp/mibs' /etc/snmp/snmp.conf 2>/dev/null || echo "mibdirs +/usr/local/share/snmp/mibs" >> /etc/snmp/snmp.conf
# cp /tmp/CISCO-SMI.my /usr/local/share/snmp/mibs
```

Now you should be able to use it. Try to translate the name of the object `ciscoProducts` from the MIB file to OID:

```
# snmptranslate -IR -On CISCO-SMI::ciscoProducts
.1.3.6.1.4.1.9.1
```

If you receive errors instead of the OID, ensure all the previous commands did not return any errors.

The object name translation worked, you are ready to use custom MIB file. Note the MIB name prefix (`CISCO-SMI::`) used in the query. You will need this when using command-line tools as well as Zabbix.

Don’t forget to restart Zabbix server/proxy before using this MIB file in Zabbix.

Keep in mind that MIB files can have dependencies. That is, one MIB may require another. In order to satisfy these dependencies you have to install all the affected MIB files.

### 3 SNMP traps

**Overview**

Receiving SNMP traps is the opposite to querying SNMP-enabled devices.

In this case, the information is sent from an SNMP-enabled device and is collected or “trapped” by Zabbix.

Usually, traps are sent upon some condition change and the agent connects to the server on port 162 (as opposed to port 161 on the agent side that is used for queries). Using traps may detect some short problems that occur amidst the query interval and may be missed by the query data.

Receiving SNMP traps in Zabbix is designed to work with `snmptrapd` and one of the built-in mechanisms for passing the traps to Zabbix - either a Perl script or SNMPTT.

The workflow of receiving a trap:

1. `snmptrapd` receives a trap
2. `snmptrapd` passes the trap to SNMPTT or calls Perl trap receiver
3. SNMPTT or Perl trap receiver parses, formats, and writes the trap to a file
4. Zabbix SNMP trapper reads and parses the trap file
5. For each trap Zabbix finds all “SNMP trapper” items with host interfaces matching the received trap address. Note that only the selected “IP” or “DNS” in host interface is used during the matching.
6. For each found item, the trap is compared to regexp in "snmptrap[regexp]". The trap is set as the value of all matched items. If no matching item is found and there is an "snmptrap.fallback" item, the trap is set as the value of that.

7. If the trap was not set as the value of any item, Zabbix by default logs the unmatched trap. (This is configured by "Log unmatched SNMP traps" in Administration → General → Other.)

1 Configuring SNMP traps

Configuring the following fields in the frontend is specific for this item type:

- Your host must have an SNMP interface

In Configuration → Hosts, in the Host interface field set an SNMP interface with the correct IP or DNS address. The address from each received trap is compared to the IP and DNS addresses of all SNMP interfaces to find the corresponding hosts.

- Configure the item

In the Key field use one of the SNMP trap keys:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>snmptrap[regexp]</code></td>
<td>Catches all SNMP traps that match the regular expression specified in regexp. If regexp is unspecified, catches any trap.</td>
<td>SNMP trap</td>
<td>This item can be set only for SNMP interfaces.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This item is supported since Zabbix 2.0.0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: Starting with Zabbix 2.0.5, user macros and global regular expressions are supported in the parameter of this item key.</td>
</tr>
<tr>
<td><code>snmptrap.fallback</code></td>
<td>Catches all SNMP traps that were not caught by any of the snmptrap[] items for that interface.</td>
<td>SNMP trap</td>
<td>This item can be set only for SNMP interfaces.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This item is supported since Zabbix 2.0.0.</td>
</tr>
</tbody>
</table>

Multi-line regexp matching is not supported at this time.

Set the Type of information to be 'Log' for the timestamps to be parsed. Note that other formats such as 'Numeric' are also acceptable but might require a custom trap handler.

For SNMP trap monitoring to work, it must first be correctly set up.

2 Setting up SNMP trap monitoring

Configuring Zabbix server/proxy

To read the traps, Zabbix server or proxy must be configured to start the SNMP trapper process and point to the trap file that is being written by SNMPTT or a perl trap receiver. To do that, edit the configuration file (zabbix_server.conf or zabbix_proxy.conf):

1. StartSNMPTrapper=1
2. SNMPTrapperFile=\[TRAPFILE\]

If systemd parameter PrivateTmp is used, this file is unlikely to work in /tmp.

Configuring SNMPTT

At first, snmptrapd should be configured to use SNMPTT.

For the best performance, SNMPTT should be configured as a daemon using snmpthandler-embedded to pass the traps to it. See instructions for configuring SNMPTT in its homepage:
http://snmptt.sourceforge.net/docs/snmptt.shtml

When SNMPTT is configured to receive the traps, configure snmptt.ini:

1. enable the use of the Perl module from the NET-SNMP package:
   ```
   net_snmp_perl_enable = 1
   ```
2. log traps to the trap file which will be read by Zabbix:
   ```
   log_enable = 1
   log_file = [TRAP FILE]
   ```
3. set the date-time format:
   ```
   date_time_format = %H:%M:%S %Y/%m/%d = [DATE TIME FORMAT]
   ```
The “net-snmp-perl” package has been removed in RHEL/CentOS 8.0-8.2; re-added in RHEL 8.3. For more information, see the known issues.

Now format the traps for Zabbix to recognize them (edit snmptt.conf):

1. Each FORMAT statement should start with “ZBXTRAP [address]”, where [address] will be compared to IP and DNS addresses of SNMP interfaces on Zabbix. E.g.:
   ```
   EVENT coldStart .1.3.6.1.6.3.1.1.5.1 "Status Events" Normal
   FORMAT ZBXTRAP $aA Device reinitialized (coldStart)
   ```
2. See more about SNMP trap format below.

Do not use unknown traps - Zabbix will not be able to recognize them. Unknown traps can be handled by defining a general event in snmptt.conf:

```plaintext
EVENT general .* "General event" Normal
```

Configuring Perl trap receiver

Requirements: Perl, Net-SNMP compiled with --enable-embedded-perl (done by default since Net-SNMP 5.4)

Perl trap receiver (look for misc/snmptrap/zabbix_trap_receiver.pl) can be used to pass traps to Zabbix server directly from snmptrapd. To configure it:

1. add the perl script to snmptrapd configuration file (snmptrapd.conf), e.g.:
   ```
   perldo "[FULL PATH TO PERL RECEIVER SCRIPT]"
   ```
2. configure the receiver, e.g:
   ```
   $SNMPTrapperFile = '[TRAP FILE]';
   $DateTimeFormat = '[DATE TIME FORMAT]';
   ```

If script name is not quoted, snmptrapd will refuse to start up with messages, similar to these:

```
Regexp modifiers "/l" and "/a" are mutually exclusive at (eval 2) line 1, at end of line
Regexp modifier "/l" may not appear twice at (eval 2) line 1, at end of line
```

net-snmp agent does not support AES256 with SNMPv3/USM.

SNMP trap format

All customized perl trap receivers and SNMPTT trap configuration must format the trap in the following way: `[timestamp] [the trap, part 1] ZBXTRAP [address] [the trap, part 2]`, where

- `[timestamp]` - timestamp used for log items
- `ZBXTRAP` - header that indicates that a new trap starts in this line
- `[address]` - IP address used to find the host for this trap

Note that “ZBXTRAP” and “[address]” will be cut out from the message during processing. If the trap is formatted otherwise, Zabbix might parse the traps unexpectedly.

Example trap:

```
11:30:15 2011/07/27 .1.3.6.1.6.3.1.1.5.3 Normal "Status Events" localhost - ZBXTRAP 192.168.1.1 Link down on interface 2. Admin state: 1. Operational state: 2
```

This will result in the following trap for SNMP interface with IP=192.168.1.1:

```
11:30:15 2011/07/27 .1.3.6.1.6.3.1.1.5.3 Normal "Status Events" localhost - Link down on interface 2. Admin state: 1.
```

3 System requirements

Large file support

Zabbix has “Large file support” for SNMP trapper files. The maximum file size that Zabbix can read is $2^63$ (8 EiB). Note that the filesystem may impose a lower limit on the file size.

Log rotation

Zabbix does not provide any log rotation system - that should be handled by the user. The log rotation should first rename the old file and only later delete it so that no traps are lost:

1. Zabbix opens the trap file at the last known location and goes to step 3
2. Zabbix checks if the currently opened file has been rotated by comparing the inode number to the defined trap file’s inode number. If there is no opened file, Zabbix resets the last location and goes to step 1.
3. Zabbix reads the data from the currently opened file and sets the new location.
4. The new data are parsed. If this was the rotated file, the file is closed and goes back to step 2.
5. If there was no new data, Zabbix sleeps for 1 second and goes back to step 2.
File system

Because of the trap file implementation, Zabbix needs the file system to support inodes to differentiate files (the information is acquired by a stat() call).

4 Setup example

This example uses snmptrapd + SNMPTT to pass traps to Zabbix server. Setup:

1. **zabbix_server.conf** - configure Zabbix to start SNMP trapper and set the trap file:
   - StartSNMPTrapper=1
   - SNMPTrapperFile=/tmp/my_zabbix_traps.tmp
2. **snmptrapd.conf** - add SNMPTT as the trap handler:
   - traphandle default snmp
3. **snmp.ini** - enable the use of the Perl module from the NET-SNMP package:
   - net_snmp_perl_enable = 1
   - configure output file and time format:
   - log_file = /tmp/my_zabbix_traps.tmp
   - date_time_format = %H:%M:%S %Y/%m/%d
4. **snmptt.conf** - define a default trap format:
   - EVENT general.* "General event" Normal
   - FORMAT ZBXTRAP $a $a
5. Create an SNMP item TEST:
   - Host's SNMP interface IP: 127.0.0.1
   - Key: snmptrap["General"]
   - Log time format: hh:mm:ss yyyy/MM/dd

This results in:

1. Command used to send a trap:
   snmptrap -v 1 -c public 127.0.0.1.1.3.6.1.6.3.1.1.5.3.0 0.0.0.0 6 33 55 .1.3.6.1.6.3.1.1.5.3 0 "teststring000"
2. The received trap:
   15:48:18 2011/07/26 .1.3.6.1.6.3.1.1.5.3.0.33 Normal "General event" localhost - ZBXTRAP 127.0.0.1 127.0.0.1
3. Value for item TEST:
   15:48:18 2011/07/26 .1.3.6.1.6.3.1.1.5.3.0.33 Normal "General event" localhost - 127.0.0.1

This simple example uses SNMPTT as traphandle. For better performance on production systems, use embedded Perl to pass traps from snmptrapd to SNMPTT or directly to Zabbix.

5 See also

- Zabbix blog article on SNMP traps
- CentOS based SNMP trap tutorial on zabbix.org

4 IPMI checks

Overview

You can monitor the health and availability of Intelligent Platform Management Interface (IPMI) devices in Zabbix. To perform IPMI checks Zabbix server must be initially configured with IPMI support.

IPMI is a standardized interface for remote “lights-out” or “out-of-band” management of computer systems. It allows to monitor hardware status directly from the so-called “out-of-band” management cards, independently from the operating system or whether the machine is powered on at all.

Zabbix IPMI monitoring works only for devices having IPMI support (HP iLO, DELL DRAC, IBM RSA, Sun SSP, etc).

Since Zabbix 3.4, a new IPMI manager process has been added to schedule IPMI checks by IPMI pollers. Now a host is always polled by only one IPMI poller at a time, reducing the number of open connections to BMC controllers. With those changes it’s safe to increase the number of IPMI pollers without worrying about BMC controller overloading. The IPMI manager process is automatically started when at least one IPMI poller is started.

See also known issues for IPMI checks.

Configuration

Host configuration

A host must be configured to process IPMI checks. An IPMI interface must be added, with the respective IP and port numbers, and IPMI authentication parameters must be defined.
See the configuration of hosts for more details.

Server configuration

By default, the Zabbix server is not configured to start any IPMI pollers, thus any added IPMI items won’t work. To change this, open the Zabbix server configuration file (zabbix_server.conf) as root and look for the following line:

```bash
# StartIPMPollers=0
```

Uncomment it and set poller count to, say, 3, so that it reads:

```bash
StartIPMPollers=3
```

Save the file and restart zabbix_server afterwards.

Item configuration

When configuring an item on a host level:

- Select ‘IPMI agent’ as the Type
- Enter an item key that is unique within the host (say, ipmi.fan.rpm)
- For Host interface select the relevant IPMI interface (IP and port). Note that an IPMI interface must exist on the host.
- Specify the IPMI sensor (for example ‘FAN MOD 1A RPM’ on Dell Poweredge) to retrieve the metric from. By default, the sensor ID should be specified. It is also possible to use prefixes before the value:
  - id: - to specify sensor ID;
  - name: - to specify sensor full name. This can be useful in situations when sensors can only be distinguished by specifying the full name.
- Select the respective type of information (‘Numeric (float)’ in this case; for discrete sensors - ‘Numeric (unsigned)’), units (most likely ‘rpm’) and any other required item attributes

Supported checks

The table below describes in-built items that are supported in IPMI agent checks.

<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
<th>Return value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipmi.get</td>
<td>IPMI-sensor related information.</td>
<td>JSON object</td>
<td>This item can be used for the discovery of IPMI sensors. Supported since Zabbix 5.0.0.</td>
</tr>
</tbody>
</table>

Timeout and session termination

IPMI message timeouts and retry counts are defined in OpenIPMI library. Due to the current design of OpenIPMI, it is not possible to make these values configurable in Zabbix, neither on interface nor item level.

IPMI session inactivity timeout for LAN is 60 +/-3 seconds. Currently it is not possible to implement periodic sending of Activate Session command with OpenIPMI. If there are no IPMI item checks from Zabbix to a particular BMC for more than the session timeout configured in BMC then the next IPMI check after the timeout expires will time out due to individual message timeouts, retries or receive error. After that a new session is opened and a full rescans of the BMC is initiated. If you want to avoid unnecessary rescans of the BMC it is advised to set the IPMI item polling interval below the IPMI session inactivity timeout configured in BMC.

Notes on IPMI discrete sensors

To find sensors on a host start Zabbix server with `DebugLevel=4` enabled. Wait a few minutes and find sensor discovery records in Zabbix server logfile:

```bash
$ grep 'Added sensor' zabbix_server.log
```

8358:20130318:111122.170 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:7 id:'CATERR' reading_type:0x3 ('discrete_state') type:0x7 ('processor') full_name:'(r0.32.3.0).CATERR'
8358:20130318:111122.170 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:15 id:'CPU Therm Trip' reading_type:0x3 ('discrete_state') type:0x1 ('temperature') full_name:'(7.1).CPU Therm Trip'
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'System Event Log' reading_type:0x6f ('sensor specific') type:0x10 ('event_logging_disabled') full_name:'(7.1).System Event Log'
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'PhysicalSecurity' reading_type:0x6f ('sensor specific') type:0x5 ('physical_security') full_name:'(23.1).PhysicalSecurity'
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'IPMI Watchdog' reading_type:0x6f ('sensor specific') type:0x23 ('watchdog_2') full_name:'(7.7).IPMI Watchdog'
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'Power Unit Stat' reading_type:0x1 ('threshold') type:0x1 ('temperature') full_name:'(21.1).Power Unit Stat'
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'P1 Therm Ctrl %' reading_type:0x1 ('threshold') type:0x4 ('fan') full_name:'(29.1).P1 Therm Ctrl %'
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'P1 Mem Margin' reading_type:0x1 ('threshold') type:0x1 ('temperature') full_name:'(7.6).P1 Mem Margin'
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'Front Panel Temp' reading_type:0x1 ('threshold') type:0x1 ('temperature') full_name:'(7.6).Front Panel Temp'

The first parameter to start with is "reading_type". Use "Table 42-1, Event/Reading Type Code Ranges" from the specifications to decode "reading_type" code. Most of the sensors in our example have "reading_type:0x1" which means "threshold" sensor. "Table 42-3, Sensor Type Codes" shows that "type:0x1" means temperature sensor, "type:0x2" - voltage sensor, "type:0x4" - Fan etc. Threshold sensors are sometimes called "analog" sensors as they measure continuous parameters like temperature, voltage, revolutions per minute.

Another example - a sensor with "reading_type:0x3". "Table 42-1, Event/Reading Type Code Ranges" says that reading type codes 02h-0Ch mean "Generic Discrete" sensor. Discrete sensors have up to 15 possible states (in other words - up to 15 meaningful bits). For example, for sensor 'CATERR' with "type:0x7" the "Table 42-3, Sensor Type Codes" shows that this type means "Processor" and the meaning of individual bits is: 00h (the least significant bit) - IERR, 01h - Thermal Trip etc.

There are few sensors with "reading_type:0x6f" in our example. For these sensors the "Table 42-1, Event/Reading Type Code Ranges" advises to use "Table 42-3, Sensor Type Codes" for decoding meanings of bits. For example, sensor 'Power Unit Stat' has type "type:0x9" which means "Power Unit". Offset 00h means "Power Off/Power Down". In other words if the least significant bit is 1, then server is powered off. To test this bit, the `bitand` function with mask '1' can be used. The trigger expression could be like:

```
bitand(last(/www.example.com/Power Unit Stat,#1),1)=1
```
to warn about a server power off.

Notes on discrete sensor names in OpenIPMI-2.0.16, 2.0.17, 2.0.18 and 2.0.19

Names of discrete sensors in OpenIPMI-2.0.16, 2.0.17 and 2.0.18 often have an additional "0" (or some other digit or letter) appended at the end. For example, while `ipmitool` and OpenIPMI-2.0.19 display sensor names as "PhysicalSecurity" or "CATERR", in OpenIPMI-2.0.16, 2.0.17 and 2.0.18 the names are "PhysicalSecurity0" or "CATERR0", respectively.

When configuring an IPMI item with Zabbix server using OpenIPMI-2.0.16, 2.0.17 and 2.0.18, use these names ending with "0" in the IPMI sensor field of IPMI agent items. When your Zabbix server is upgraded to a new Linux distribution, which uses OpenIPMI-2.0.19 (or later), items with these IPMI discrete sensors will become "NOT SUPPORTED". You have to change their IPMI sensor names (remove the '0' in the end) and wait for some time before they turn "Enabled" again.

Notes on threshold and discrete sensor simultaneous availability

Some IPMI agents provide both a threshold sensor and a discrete sensor under the same name. In Zabbix versions prior to 2.2.8 and 2.4.3, the first provided sensor was chosen. Since versions 2.2.8 and 2.4.3, preference is always given to the threshold sensor.

Notes on connection termination

If IPMI checks are not performed (by any reason: all host IPMI items disabled/not supported, host disabled/deleted, host in maintenance etc.) the IPMI connection will be terminated from Zabbix server or proxy in 3 to 4 hours depending on the time when Zabbix server/proxy was started.

### 5 Simple checks

**Overview**

Simple checks are normally used for remote agent-less checks of services.

Note that Zabbix agent is not needed for simple checks. Zabbix server/proxy is responsible for the processing of simple checks (making external connections, etc).

Examples of using simple checks:

```
net.tcp.service[ftp,,155]
neta.tcp.service[http]
net.tcp.service.perf[http,,8080]
net.udp.service.perf[ntp]
```

245
User name and Password fields in simple check item configuration are used for VMware monitoring items; ignored otherwise.

Supported simple checks

List of supported simple checks:

See also:

- VMware monitoring item keys

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return Value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>icmpping</td>
<td>Host accessibility by ICMP ping</td>
<td>0 - ICMP ping fails</td>
<td>[target],&lt;packets&gt;,&lt;interval&gt;,&lt;size&gt;,&lt;timeout&gt;</td>
<td>Example: =&gt; icmpping[,4] → if at least one packet of the four is returned, the item will return 1. See also: table of default values.</td>
</tr>
<tr>
<td>icmppingloss</td>
<td>Percentage of lost packets.</td>
<td>Float.</td>
<td>[target],&lt;packets&gt;,&lt;interval&gt;,&lt;size&gt;,&lt;timeout&gt;</td>
<td>See also: table of default values.</td>
</tr>
<tr>
<td>icmppingsec</td>
<td>ICMP ping response time (in seconds).</td>
<td>Float.</td>
<td>[target],&lt;packets&gt;,&lt;interval&gt;,&lt;size&gt;,&lt;timeout&gt;,&lt;mode&gt;</td>
<td>Packets which are lost or timed out are not used in the calculation. If host is not available (timeout reached), the item will return 0. If the return value is less than 0.0001 seconds, the value will be set to 0.0001 seconds. See also: table of default values.</td>
</tr>
<tr>
<td>net.tcp.service</td>
<td>Checks if service is running and accepting TCP connections.</td>
<td>0 - service is down</td>
<td>[service],[&lt;ip&gt;],[&lt;port&gt;]</td>
<td>Example: =&gt; net.tcp.service[ftp,45] → can be used to test the availability of FTP server on TCP port 45. Note that with tcp service indicating the port is mandatory. These checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually). Checking of encrypted protocols (like IMAP on port 993 or POP on port 995) is currently not supported. As a workaround, please use net.tcp.service[tcp,&lt;ip&gt;;&lt;port&gt;] for checks like these. https and telnet services are supported since Zabbix 2.0.</td>
</tr>
<tr>
<td>net.tcp.service.perf</td>
<td></td>
<td></td>
<td>[service],[&lt;ip&gt;],[&lt;port&gt;]</td>
<td></td>
</tr>
<tr>
<td>Checks</td>
<td>Float.</td>
<td>service - possible values: ssh, ldap, smtp, ftp, http, pop, nntp, imap, tcp, https, telnet (see details)</td>
<td>Example: =&gt; net.tcp.service.perf[ssh] → can be used to test the speed of initial response from SSH server.</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>------------------------------------------------</td>
<td>--</td>
<td>------</td>
</tr>
<tr>
<td>performance of TCP service.</td>
<td>0.000000 - service is down</td>
<td>ip - IP address or DNS name (by default, host IP/DNS is used) port - port number (by default standard service port number is used).</td>
<td>Note that with tcp service indicating the port is mandatory. Checking of encrypted protocols (like IMAP on port 993 or POP on port 995) is currently not supported. As a workaround, please use net.tcp.service.perf[tcp,&lt;ip&gt;,port] for checks like these. https and telnet services are supported since Zabbix 2.0. Called tcp_perf before Zabbix 2.0.</td>
<td></td>
</tr>
<tr>
<td>net.udp.service[service,&lt;ip&gt;,&lt;port&gt;]</td>
<td>Checks if service is running and responding to UDP requests.</td>
<td>0 - service is down</td>
<td>Example: =&gt; net.udp.service[ntp,45] → can be used to test the availability of NTP service on UDP port 45.</td>
<td></td>
</tr>
<tr>
<td>service - possible values: ntp (see details)</td>
<td>ip - IP address or DNS name (by default host IP/DNS is used)</td>
<td>port - port number (by default standard service port number is used).</td>
<td>This item is supported since Zabbix 3.0, but ntp service was available for net.tcp.service[] item in prior versions.</td>
<td></td>
</tr>
<tr>
<td>net.udp.service.perf[service,&lt;ip&gt;,&lt;port&gt;]</td>
<td>Checks performance of UDP service.</td>
<td>Float.</td>
<td>Example: =&gt; net.udp.service.perf[ntp] → can be used to test response time from NTP service.</td>
<td></td>
</tr>
<tr>
<td>service - possible values: ntp (see details)</td>
<td>ip - IP address or DNS name (by default host IP/DNS is used)</td>
<td>port - port number (by default standard service port number is used).</td>
<td>This item is supported since Zabbix 3.0, but ntp service was available for net.tcp.service[] item in prior versions.</td>
<td></td>
</tr>
</tbody>
</table>

For SourceIP support in LDAP simple checks (e.g. net.tcp.service[ldap]), OpenLDAP version 2.6.1 or above is required.

Timeout processing
Zabbix will not process a simple check longer than the Timeout seconds defined in the Zabbix server/proxy configuration file.

ICMP pings
Zabbix uses external utility **fping** for processing of ICMP pings.

The utility is not part of Zabbix distribution and has to be additionally installed. If the utility is missing, has wrong permissions or its location does not match the location set in the Zabbix server/proxy configuration file (`FpingLocation` parameter), ICMP pings (**icmpping**, **icmppingloss**, **icmppingsec**) will not be processed.

See also: known issues

**fping** must be executable by the user Zabbix daemons run as and setuid root. Run these commands as user **root** in order to set up correct permissions:

```
shell> chown root:zabbix /usr/sbin/fping
shell> chmod 4710 /usr/sbin/fping
```

After performing the two commands above check ownership of the **fping** executable. In some cases the ownership can be reset by executing the chmod command.

Also check, if user zabbix belongs to group zabbix by running:
shell> groups zabbix

and if it’s not add by issuing:

shell> usermod -a -G zabbix zabbix

Defaults, limits and description of values for ICMP check parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Description</th>
<th>Fping’s flag</th>
<th>Defaults set by</th>
<th>Allowed limits by Zabbix</th>
<th>min</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>packets</td>
<td>number</td>
<td>number of request packets to a target packets</td>
<td>-C</td>
<td>fping</td>
<td>Zabbix min</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>interval</td>
<td>millisecond</td>
<td>time to wait between successive packets</td>
<td>-p</td>
<td>1000</td>
<td>20</td>
<td>20</td>
<td>unlimited</td>
</tr>
<tr>
<td>size</td>
<td>bytes</td>
<td>packet size in bytes</td>
<td>-b</td>
<td>56 or 68</td>
<td>24</td>
<td>24</td>
<td>65507</td>
</tr>
<tr>
<td>timeout</td>
<td>millisecond</td>
<td>timeout to wait after last packet sent, affected by -C flag</td>
<td>-t</td>
<td>fping v3.x</td>
<td>500</td>
<td>50</td>
<td>unlimited</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fping v4.x - individual timeout for each packet</td>
<td></td>
<td>fping v4.x</td>
<td>inherited from -p flag, but not more than 2000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In addition Zabbix uses fping options -i interval ms (do not mix up with the item parameter interval ms mentioned in the table above, which corresponds to fping option -p) and -S source IP address (or -I in older fping versions). Those options are auto-detected by running checks with different option combinations. Zabbix tries to detect the minimal value in milliseconds that fping allows to use with -i by trying 3 values: 0, 1 and 10. The value that first succeeds is then used for subsequent ICMP checks. This process is done by each ICMP pinger process individually.

Auto-detected fping options are invalidated every hour and detected again on the next attempt to perform ICMP check. Set DebugLevel>=4 in order to view details of this process in the server or proxy log file.

Warning: fping defaults can differ depending on platform and version - if in doubt, check fping documentation.

Zabbix writes IP addresses to be checked by any of three icmpping* keys to a temporary file, which is then passed to fping. If items have different key parameters, only ones with identical key parameters are written to a single file. All IP addresses written to the single file will be checked by fping in parallel, so Zabbix icmp pinger process will spend fixed amount of time disregarding the number of IP addresses in the file.

VMware monitoring item keys

List of VMware monitoring item keys has been moved to VMware monitoring section.

6 Log file monitoring

Overview

Zabbix can be used for centralized monitoring and analysis of log files with/without log rotation support.

Notifications can be used to warn users when a log file contains certain strings or string patterns.

To monitor a log file you must have:

- Zabbix agent running on the host
- log monitoring item set up

The size limit of a monitored log file depends on large file support.

Configuration
Verify agent parameters

Make sure that in the **agent configuration file**:

- ‘Hostname’ parameter matches the hostname in the frontend
- Servers in the ‘ServerActive’ parameter are specified for the processing of active checks

Item configuration

Configure a log monitoring **item**.

<table>
<thead>
<tr>
<th>Item</th>
<th>Tags</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Log item</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Zabbix agent (active)</td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>log[/var/log/syslog.error]</td>
<td></td>
</tr>
<tr>
<td>Type of information</td>
<td>Log</td>
<td></td>
</tr>
<tr>
<td>Update interval</td>
<td>30s</td>
<td></td>
</tr>
<tr>
<td>Custom intervals</td>
<td></td>
<td></td>
</tr>
<tr>
<td>History storage period</td>
<td>Do not keep history</td>
<td>Storage period</td>
</tr>
<tr>
<td>Log time format</td>
<td>Nppdphh:mm:ss</td>
<td></td>
</tr>
</tbody>
</table>

All mandatory input fields are marked with a red asterisk.

Specifically for log monitoring items you enter:

- **Type**
  - **Zabbix agent (active)** here.

- **Key**
  - Use one of the following item keys:
    - **log[]** or **logrt[]**:
      - These two item keys allow to monitor logs and filter log entries by the content regexp, if present.
      - For example: `log[/var/log/syslog.error]`. Make sure that the file has read permissions for the ‘zabbix’ user otherwise the item status will be set to ‘unsupported’.
    - **log.count[]** or **logrt.count[]**:
      - These two item keys allow to return the number of matching lines only.
      - See supported **Zabbix agent item** key section for details on using these item keys and their parameters.

- **Type of information**
  - Prefilled automatically:
    - For log[] or logrt[] items - Log;
    - For log.count[] or logrt.count[] items - Numeric (unsigned).
    - If optionally using the output parameter, you may manually select the appropriate type of information other than Log.
    - Note that choosing a non-Log type of information will lead to the loss of local timestamp.

- **Update interval (in sec)**
  - The parameter defines how often Zabbix agent will check for any changes in the log file. Setting it to 1 second will make sure that you get new records as soon as possible.
Log time format

In this field you may optionally specify the pattern for parsing the log line timestamp. If left blank the timestamp will not be parsed.

Supported placeholders:
- y: Year (0001-9999)
- M: Month (01-12)
- d: Day (01-31)
- h: Hour (00-23)
- m: Minute (00-59)
- s: Second (00-59)

For example, consider the following line from the Zabbix agent log file:

```
```

It begins with six character positions for PID, followed by date, time, and the rest of the line.

Log time format for this line would be “pppppp:yyyyMMdd:hhmmss”.

Note that “p” and “:” chars are just placeholders and can be anything but “yMdhrs”.

Important notes

• The server and agent keep the trace of a monitored log’s size and last modification time (for logrt) in two counters. Additionally:
  - The agent also internally uses inode numbers (on UNIX/GNU/Linux), file indexes (on Microsoft Windows) and MDS sums of the first 512 log file bytes for improving decisions when logfiles get truncated and rotated.
  - On UNIX/GNU/Linux systems it is assumed that the file systems where log files are stored report inode numbers, which can be used to track files.
  - On Microsoft Windows Zabbix agent determines the file system type the log files reside on and uses:
    - On NTFS file systems 64-bit file indexes.
    - On ReFS file systems (only from Microsoft Windows Server 2012) 128-bit file IDs.
    - On file systems where file indexes change (e.g. FAT32, exFAT) a fall-back algorithm is used to take a sensible approach in uncertain conditions when log file rotation results in multiple log files with the same last modification time.
  - The inode numbers, file indexes and MDS sums are internally collected by Zabbix agent. They are not transmitted to Zabbix server and are lost when Zabbix agent is stopped.
  - Do not modify the last modification time of log files with ‘touch’ utility, do not copy a log file with later restoration of the original name (this will change the file inode number). In both cases the file will be counted as different and will be analyzed from the start, which may result in duplicated alerts.
  - If there are several matching log files for logrt[] item and Zabbix agent is following the most recent of them and this most recent log file is deleted, a warning message "there are no files matching "<regexp mask>" in "<directory>" is logged. Zabbix agent ignores log files with modification time less than the most recent modification time seen by the agent for the logrt[] item being checked.
  - The agent starts reading the log file from the point it stopped the previous time.
  - The number of bytes already analyzed (the size counter) and last modification time (the time counter) are stored in the Zabbix database and are sent to the agent to make sure the agent starts reading the log file from this point in cases when the agent is just started or has received items which were previously disabled or not supported. However, if the agent has received a non-zero size counter from server, but the logrt[] or logrt.count[] item has not found and does not find matching files, the size counter is reset to 0 to analyze from the start if the files appear later.
  - Whenever the log file becomes smaller than the log size counter known by the agent, the counter is reset to zero and the agent starts reading the log file from the beginning taking the time counter into account.
  - If there are several matching files with the same last modification time in the directory, then the agent tries to correctly analyze all log files with the same modification time and avoid skipping data or analyzing the same data twice, although it cannot be guaranteed in all situations. The agent does not assume any particular log file rotation scheme nor determines one. When presented multiple log files with the same last modification time, the agent will process them in a lexicographically descending order. Thus, for some rotation schemes the log files will be analyzed and reported in their original order. For other rotation schemes the original log file order will not be honored, which can lead to reporting matched log file records in altered order (the problem does not happen if log files have different last modification times).
  - Zabbix agent processes new records of a log file once per Update interval seconds.
  - Zabbix agent does not send more than maxlines of a log file per second. The limit prevents overloading of network and CPU resources and overrides the default value provided by MaxLinesPerSecond parameter in the agent configuration file.
  - To find the required string Zabbix will process 10 times more new lines than set in MaxLinesPerSecond. Thus, for example, if a log[] or logrt[] item has Update interval of 1 second, by default the agent will analyze no more than 200 log file records and will send no more than 20 matching records to Zabbix server in one check. By increasing MaxLinesPerSecond in the agent configuration file or setting maxlines parameter in the item key, the limit can be increased up to 10000 analyzed log file records and 1000 matching records sent to Zabbix server in one check. If the Update interval is set to 2 seconds the limits for one check would be set 2 times higher than with Update interval of 1 second.
Additionally, log and log.count values are always limited to 50% of the agent send buffer size, even if there are no non-log values in it. So for the maxlines values to be sent in one connection (and not in several connections), the agentBufferSize parameter must be at least maxlines x 2.

- In the absence of log items all agent buffer size is used for non-log values. When log values come in they replace the older non-log values as needed, up to the designated 50%.
- For log file records longer than 256kB, only the first 256kB are matched against the regular expression and the rest of the record is ignored. However, if Zabbix agent is stopped while it is dealing with a long record the agent internal state is lost and the long record may be analyzed again and differently after the agent is started again.
- Special note for “/” path separators: if file_format is "file\.log", then there should not be a “file” directory, since it is not possible to unambiguously define whether “." is escaped or is the first symbol of the file name.
- Regular expressions for logrt are supported in filename only, directory regular expression matching is not supported.
- On UNIX platforms a logrt[] item becomes NOTSUPPORTED if a directory where the log files are expected to be found does not exist.
- On Microsoft Windows, if a directory does not exist the item will not become NOTSUPPORTED (for example, if directory is misspelled in item key).
- An absence of log files for logrt[] item does not make it NOTSUPPORTED. Errors of reading log files for logrt[] item are logged as warnings into Zabbix agent log file but do not make the item NOTSUPPORTED.
- Zabbix agent log file can be helpful to find out why a log[] or logrt[] item became NOTSUPPORTED. Zabbix can monitor its agent log file except when at DebugLevel=4.

Extracting matching part of regular expression

Sometimes we may want to extract only the interesting value from a target file instead of returning the whole line when a regular expression match is found.

Since Zabbix 2.2.0, log items have the ability to extract desired values from matched lines. This is accomplished by the additional output parameter in log and logrt items.

Using the ‘output’ parameter allows to indicate the "capturing group" of the match that we may be interested in.

So, for example

log[/path/to/the/file,"large result buffer allocation.*Entries: ([0-9]+)",,,,
\1] should allow returning the entry count as found in the content of:


Only the number will be returned because \1 refers to the first and only capturing group: ([0-9]+).

And, with the ability to extract and return a number, the value can be used to define triggers.

Using maxdelay parameter

The ‘maxdelay’ parameter in log items allows ignoring some older lines from log files in order to get the most recent lines analyzed within the ‘maxdelay’ seconds.

Specifying ‘maxdelay’ > 0 may lead to ignoring important log file records and missed alerts. Use it carefully at your own risk only when necessary.

By default items for log monitoring follow all new lines appearing in the log files. However, there are applications which in some situations start writing an enormous number of messages in their log files. For example, if a database or a DNS server is unavailable, such applications flood log files with thousands of nearly identical error messages until normal operation is restored. By default, all those messages will be dutifully analyzed and matching lines sent to server as configured in log and logrt items.

Built-in protection against overload consists of a configurable ‘maxlines’ parameter (protects server from too many incoming matching log lines) and a 4*‘maxlines’ limit (protects host CPU and I/O from overloading by agent in one check). Still, there are 2 problems with the built-in protection. First, a large number of potentially not-so-informative messages are reported to server and consume space in the database. Second, due to the limited number of lines analyzed per second the agent may lag behind the newest log records for hours. Quite likely, you might prefer to be sooner informed about the current situation in the log files instead of crawling through old records for hours.

The solution to both problems is using the ‘maxdelay’ parameter. If ‘maxdelay’ > 0 is specified, during each check the number of processed bytes, the number of remaining bytes and processing time is measured. From these numbers the agent calculates an estimated delay - how many seconds it would take to analyze all remaining records in a log file.

If the delay does not exceed ‘maxdelay’ then the agent proceeds with analyzing the log file as usual.

If the delay is greater than ‘maxdelay’ then the agent ignores a chunk of a log file by “jumping” over it to a new estimated position so that the remaining lines could be analyzed within ‘maxdelay’ seconds.
Note that agent does not even read ignored lines into buffer, but calculates an approximate position to jump to in a file.

The fact of skipping log file lines is logged in the agent log file like this:

```
14287:20160602:174344.206 item:"logrt[/home/zabbix32/test[0-9].log",ERROR,,1000,,120.0]"
logfile:"/home/zabbix32/test1.log" skipping 679858 bytes (from byte 75653115 to byte 76332973) to meet maxdelay
```

The “to byte” number is approximate because after the “jump” the agent adjusts the position in the file to the beginning of a log line which may be further in the file or earlier.

Depending on how the speed of growing compares with the speed of analyzing the log file you may see no “jumps”, rare or often “jumps”, large or small “jumps”, or even a small “jump” in every check. Fluctuations in the system load and network latency also affect the calculation of delay and hence, “jumping” ahead to keep up with the “maxdelay” parameter.

Setting ‘maxdelay’ < ‘update interval’ is not recommended (it may result in frequent small “jumps”).

Notes on handling ’copytruncate’ log file rotation

`logrt` with the `copytruncate` option assumes that different log files have different records (at least their timestamps are different), therefore MD5 sums of initial blocks (up to the first 512 bytes) will be different. Two files with the same MD5 sums of initial blocks means that one of them is the original, another - a copy.

`logrt` with the `copytruncate` option makes effort to correctly process log file copies without reporting duplicates. However, things like producing multiple log file copies with the same timestamp, log file rotation more often than `logrt[]item update interval`, frequent restarting of agent are not recommended. The agent tries to handle all these situations reasonably well, but good results cannot be guaranteed in all circumstances.

Notes on persistent files for log[*] items

Purpose of persistent files

When Zabbix agent is started it receives a list of active checks from Zabbix server or proxy. For log[*] metrics it receives the processed log size and the modification time for finding where to start log file monitoring from. Depending on the actual log file size and modification time reported by file system the agent decides either to continue log file monitoring from the processed log size or re-analyze the log file from the beginning.

A running agent maintains a larger set of attributes for tracking all monitored log files between checks. This in-memory state is lost when the agent is stopped.

The new optional parameter `persistent_dir` specifies a directory for storing this state of log[], log.count[], logrt[] or logrt.count[] item in a file. The state of log item is restored from the persistent file after the Zabbix agent is restarted.

The primary use-case is monitoring of log file located on a mirrored file system. Until some moment in time the log file is written to both mirrors. Then mirrors are split. On the active copy the log file is still growing, getting new records. Zabbix agent analyzes it and sends processed logs size and modification time to server. On the passive copy the log file stays the same, well behind the active copy. Later the operating system and Zabbix agent are rebooted from the passive copy. The processed log size and modification time the Zabbix agent receives from server may not be valid for situation on the passive copy. To continue log file monitoring from the place the agent left off at the moment of file system mirror split the agent restores its state from the persistent file.

Agent operation with persistent file

On startup Zabbix agent knows nothing about persistent files. Only after receiving a list of active checks from Zabbix server (proxy) the agent sees that some log items should be backed by persistent files under specified directories.

During agent operation the persistent files are opened for writing (with fopen(filename, "w")) and overwritten with the latest data. The chance of losing persistent file data if the overwriting and file system mirror split happen at the same time is very small, no special handling for it. Writing into persistent file is NOT followed by enforced synchronization to storage media (fsync() is not called).

Overwriting with the latest data is done after successful reporting of matching log file record or metadata (processed log size and modification time) to Zabbix server. That may happen as often as every item check if log file keeps changing.

No special actions during agent shutdown.

After receiving a list of active checks the agent marks obsolete persistent files for removal. A persistent file becomes obsolete if:
1) the corresponding log item is no longer monitored, 2) a log item is reconfigured with a different `persistent_dir` location than before.

Removing is done with delay 24 hours because log files in NOTSUPPORTED state are not included in the list of active checks but they may become SUPPORTED later and their persistent files will be useful.
If the agent is stopped before 24 hours expire, then the obsolete files will not be deleted as Zabbix agent is not getting info about their location from Zabbix server anymore.

Reconfiguring a log item’s **persistent_dir** back to the old **persistent_dir** location while the agent is stopped, without deleting the old persistent file by user - will cause restoring the agent state from the old persistent file resulting in missed messages or false alerts.

Naming and location of persistent files

Zabbix agent distinguishes active checks by their keys. For example, logrt[/home/zabbix/test.log] and logrt[/home/zabbix/test.log,] are different items. Modifying the item logrt[/home/zabbix/test.log,,10] in frontend to logrt[/home/zabbix/test.log,,20] will result in deleting the item logrt[/home/zabbix/test.log,,10] from the agent’s list of active checks and creating logrt[/home/zabbix/test.log,,20] item (some attributes are carried across modification in frontend/server, not in agent).

The file name is composed of MD5 sum of item key with item key length appended to reduce possibility of collisions. For example, the state of logrt[/home/zabbix50/test.log,,] item will be kept in persistent file c963ade4008054813bbc0a650bb8e09266.

Multiple log items can use the same value of **persistent_dir**.

**persistent_dir** is specified by taking into account specific file system layouts, mount points and mount options and storage mirroring configuration - the persistent file should be on the same mirrored filesystem as the monitored log file.

If **persistent_dir** directory cannot be created or does not exist, or access rights for Zabbix agent does not allow to create/write/read/delete files the log item becomes NOTSUPPORTED.

If access rights to persistent storage files are removed during agent operation or other errors occur (e.g. disk full) then errors are logged into the agent log file but the log item does not become NOTSUPPORTED.

Load on I/O

Item’s persistent file is updated after successful sending of every batch of data (containing item’s data) to server. For example, default ‘BufferSize’ is 100. If a log item has found 70 matching records then the first 50 records will be sent in one batch, persistent file will be updated, then remaining 20 records will be sent (maybe with some delay when more data is accumulated) in the 2nd batch, and the persistent file will be updated again.

Actions if communication fails between agent and server

Each matching line from log[] and logrt[] item and a result of each log..count[] and logrt..count[] item check requires a free slot in the designated 50% area in the agent send buffer. The buffer elements are regularly sent to server (or proxy) and the buffer slots are free again.

While there are free slots in the designated log area in the agent send buffer and communication fails between agent and server (or proxy) the log monitoring results are accumulated in the send buffer. This helps to mitigate short communication failures.

During longer communication failures all log slots get occupied and the following actions are taken:

- log[] and logrt[] item checks are stopped. When communication is restored and free slots in the buffer are available the checks are resumed from the previous position. No matching lines are lost, they are just reported later.
- log..count[] and logrt..count[] checks are stopped if maxdelay = 0 (default). Behavior is similar to log[] and logrt[] items as described above. Note that this can affect log..count[] and logrt..count[] results: for example, one check counts 100 matching lines in a log file, but as there are no free slots in the buffer the check is stopped. When communication is restored the agent counts the same 100 matching lines and also 70 new matching lines. The agent now sends count = 170 as if they were found in one check.
- log..count[] and logrt..count[] checks with maxdelay > 0: if there was no "jump" during the check, then behavior is similar to described above. If a "jump" over log file lines took place then the position after "jump" is kept and the counted result is discarded. So, the agent tries to keep up with a growing log file even in case of communication failure.

7 Calculated items

Overview

With calculated items it is possible to create calculations based on the values of other items.

Calculations may use both:

- single values of individual items
- complex filters to select multiple items for aggregations (see aggregate calculations for details)

Thus, calculated items are a way of creating virtual data sources. All calculations are done by Zabbix server only. The values are periodically calculated based on the arithmetical expression used.

253
The resulting data is stored in the Zabbix database as for any other item; both history and trend values are stored and graphs can be generated.

If the calculation result is a float value it will be trimmed to an integer if the calculated item type of information is Numeric (unsigned).

Calculated items share their syntax with trigger expressions. Comparison to strings is allowed in calculated items. Calculated items may be referenced by macros or other entities same as any other item type.

To use calculated items, choose the item type Calculated.

Configurable fields
The key is a unique item identifier (per host). You can create any key name using supported symbols.

Calculation definition should be entered in the Formula field. There is virtually no connection between the formula and the key. The key parameters are not used in the formula in any way.

The syntax of a simple formula is:

\[ \text{function}(/\text{host}/\text{key},<\text{parameter}_1>,<\text{parameter}_2>,...) \]

where:

| function | One of the supported functions: last, min, max, avg, count, etc |
| host     | Host of the item that is used for calculation. |
| key      | The current host can be omitted (i.e. as in function(/\text{key},\text{parameter},...)). |
| parameter(s) | Key of the item that is used for calculation. |
|          | Parameters of the function, if required. |

User macros in the formula will be expanded if used to reference a function parameter, item filter parameter, or a constant. User macros will NOT be expanded if referencing a function, host name, item key, item key parameter or operator.

A more complex formula may use a combination of functions, operators and brackets. You can use all functions and operators supported in trigger expressions. The logic and operator precedence are exactly the same.

Unlike trigger expressions, Zabbix processes calculated items according to the item update interval, not upon receiving a new value.

All items that are referenced by history functions in the calculated item formula must exist and be collecting data. Also, if you change the item key of a referenced item, you have to manually update any formulas using that key.

A calculated item may become unsupported in several cases:

- referenced item(s)
  - is not found
  - is disabled
  - belongs to a disabled host
  - is not supported (except with nodata() function and operators with unknown values)
- no data to calculate a function
- division by zero
- incorrect syntax used

Usage examples
Example 1
Calculating percentage of free disk space on ‘/’.
Use of function last:

\[ 100\times\text{last}(/\text{vfs.fs.size}[/,\text{free}])/\text{last}(/\text{vfs.fs.size}[/,\text{total}]) \]

Zabbix will take the latest values for free and total disk spaces and calculate percentage according to the given formula.

Example 2
Calculating a 10-minute average of the number of values processed by Zabbix.
Use of function avg:

\[ \text{avg}(/\text{Zabbix Server}/\text{zabbix[wcache,values]},10\text{m}) \]
Note that extensive use of calculated items with long time periods may affect performance of Zabbix server.

Example 3
Calculating total bandwidth on eth0.

Sum of two functions:
\[ \text{last}(/\text{net.if.in[eth0,bytes]}+\text{last}(/\text{net.if.out[eth0,bytes]}) \]

Example 4
Calculating percentage of incoming traffic.

More complex expression:
\[ 100\ast\text{last}(/\text{net.if.in[eth0,bytes]})/(\text{last}(/\text{net.if.in[eth0,bytes]}+\text{last}(/\text{net.if.out[eth0,bytes]})) \]
See also: Examples of aggregate calculations

Aggregate calculations

Overview
In aggregate calculations information from several items may be collected by Zabbix server (by doing direct database queries) to calculate an aggregate, depending on the aggregate function used.

Aggregate calculations do not require any agent running on the host being monitored.

To define aggregate calculations, select the Calculated item type.

Syntax
Aggregates can be retrieved by working with either:
- history of items:
  \[ \text{aggregate}_\text{function}(\text{function}(/\text{host/item,parameter}),\text{function}(/\text{host2/item2,parameter}),...) \]
- a foreach function as the only parameter:
  \[ \text{aggregate}_\text{function}(/\text{foreach}_\text{function}(/\ast?key?[\text{group}="\text{host group}"]\text{,timeperiod}) \]
where:
- aggregate_function is one of the supported aggregate functions: avg, max, min, sum, etc.
- foreach_function is one of the supported foreach functions: avg_foreach, count_foreach, etc.

Foreach functions use an item filter, to work with the history of multiple items, and return an array of values - one for each item.

If the aggregate results in a float value it will be trimmed to an integer if the aggregated item type of information is Numeric (unsigned).

An aggregate calculation may become unsupported if:
- none of the referenced items is found (which may happen if the item key is incorrect, none of the items exists or all included groups are incorrect)
- no data to calculate a function

Usage examples
Examples of keys for aggregate calculations.

Example 1
Total disk space of host group ‘MySQL Servers’.
\[ \text{sum(last_foreach}(/\ast/vfs.fs.size[/,total]?[\text{group}="\text{MySQL Servers}"]))) \]

Example 2
Sum of latest values of all items matching net.if.in[*] on the host.
\[ \text{sum(last_foreach}(/\text{host/net.if.in[*]}) \]

Example 3
Average processor load of host group ‘MySQL Servers’.
avg(last_foreach(/*/system.cpu.load[,avg1]?[group="MySQL Servers"]))

Example 4

5-minute average of the number of queries per second for host group 'MySQL Servers'.

avg(avg_foreach(/*/mysql.qps?[group="MySQL Servers"],5m))

Example 5

Average CPU load on all hosts in multiple host groups that have the specific tags.

avg(last_foreach(/*/system.cpu.load?[(group="Servers A" or group="Servers B" or group="Servers C") and (tag="Service:" or tag="Importance:High")])

Example 6

Calculation used on the latest item value sums of a whole host group.

sum(last_foreach(/*/net.if.out[eth0,bytes]?[group="video"])) / sum(last_foreach(/*/nginx_stat.sh[active]?[group="video"]))

Example 7

The total number of unsupported items in host group ‘Zabbix servers’.

sum(last_foreach(/*/zabbix[host,,items_unsupported]?[group="Zabbix servers"]))

### 8 Internal checks

**Overview**

Internal checks allow to monitor the internal processes of Zabbix. In other words, you can monitor what goes on with Zabbix server or Zabbix proxy.

Internal checks are calculated:

- on Zabbix server - if the host is monitored by server
- on Zabbix proxy - if the host is monitored by proxy

Internal checks are processed by server or proxy regardless of host maintenance status.

To use this item, choose the **Zabbix internal** item type.

Internal checks are processed by Zabbix pollers.

**Supported checks**

- Parameters without angle brackets are constants - for example, ‘host’ and ‘available’ in zabbix[host,<type>,available]. Use them in the item key as is.
- Values for items and item parameters that are “not supported on proxy” can only be gathered if the host is monitored by server. And vice versa, values “not supported on server” can only be gathered if the host is monitored by proxy.

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>zabbix[boottime]</code></td>
<td>Startup time of Zabbix server or Zabbix proxy process in seconds.</td>
<td>Integer</td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>Description</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Discover</td>
<td>JSON</td>
<td>This item can be used in low-level discovery.</td>
<td></td>
</tr>
<tr>
<td>high availability cluster nodes.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zabbix[host_items]</td>
<td>Integer</td>
<td>This item is supported since Zabbix 3.0.0.</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Integer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>of enabled items supported and not supported on the host.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zabbix[host_items_unsupported]</td>
<td>Integer</td>
<td>This item is supported since Zabbix 3.0.0.*</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Integer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>of enabled unsupported items on the host.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zabbix[host_maintenance]</td>
<td>0 - host in normal state, 1 - host in maintenance with data collection, 2 - host in maintenance without data collection.</td>
<td>This item is always processed by Zabbix server regardless of host location (on server or proxy). The proxy will not receive this item with configuration data. The second parameter must be empty and is reserved for future use.</td>
<td></td>
</tr>
<tr>
<td>Current</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>main-re-nance status of a host.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zabbix[host,active_agent,available]</td>
<td>0 - unknown, 1 - available, 2 - not available.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Availability of active agent checks on the host.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zabbix[host,discovery,interfaces]</td>
<td>257</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key</td>
<td>JSON object.</td>
<td>This item can be used in low-level discovery. This item is supported since Zabbix 3.4.0. (not supported on proxy)</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>--------------</td>
<td>----------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Details of all configured interfaces of the host in Zabbix frontend.</td>
<td>zabbix[host,&lt;type&gt;,available]</td>
<td>Availability of the main interface of a particular type of checks on the host. Valid types are: agent, snmp, ipmi, jmx. The item value is calculated according to configuration parameters regarding host unreachability/unavailability. This item is supported since Zabbix 2.0.0.</td>
<td></td>
</tr>
<tr>
<td>zabbix[hosts]</td>
<td></td>
<td>Number of monitored hosts.</td>
<td></td>
</tr>
<tr>
<td>zabbix[items]</td>
<td></td>
<td>Number of enabled items (supported and not supported).</td>
<td></td>
</tr>
<tr>
<td>zabbix[items Unsupported]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Key

Number of not supported items.

**zabbix[java,<param>]**

<table>
<thead>
<tr>
<th>Information about Zabbix Java gateway availability using nodata() trigger function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid values for <em>param</em> are: ping, version</td>
</tr>
<tr>
<td>Second parameter must be empty and is reserved for future use.</td>
</tr>
</tbody>
</table>

If `<param>` is ping, "1" is returned. Can be used to check Java gateway availability using nodata() trigger function.

If `<param>` is version, version of Java gateway is returned. Example: "2.0.0".

**zabbix[lld_queue]**

<table>
<thead>
<tr>
<th>Count of values enqueued in the low-level discovery processing queue.</th>
</tr>
</thead>
<tbody>
<tr>
<td>This item can be used to monitor the low-level discovery processing queue length.</td>
</tr>
</tbody>
</table>

This item is supported since Zabbix 4.2.0.

**zabbix[preprocessing_queue]**

<table>
<thead>
<tr>
<th>Count of values enqueued in the preprocessing queue.</th>
</tr>
</thead>
<tbody>
<tr>
<td>This item can be used to monitor the preprocessing queue length.</td>
</tr>
</tbody>
</table>

This item is supported since Zabbix 3.4.0.

**zabbix[process,<type>,<mode>,<state>]**
Supported types of server processes:
alert manager, alert syncer,alerter, availability manager,
configuration syncer, discoverer, escalator, history poller,
history syncer, housekeeper, http poller, icmp pinger, ipmi
manager, ipmi poller, java poller, lld manager, lld worker, odbc
poller, poller, preprocessing manager, preprocessing worker,
proxy poller, self-monitoring, snmp trapper, task manager,
timer, trapper, unreachable poller, vmware collector

Supported types of proxy processes:
availability manager, configuration syncer, data sender,
discoverer, heartbeat sender, history syncer, housekeeper,
http poller, icmp pinger, ipmi manager, ipmi poller, java poller,
odbs poller, poller, preprocessing manager, preprocessing
worker, self-monitoring, snmp trapper, task manager, trapper,
unreachable poller, vmware collector

Valid modes are:
avg - average value for all processes of a given type (default)
count - returns number of forks for a given process type,
<state> should not be specified
max - maximum value
min - minimum value
<process number> - process number (between 1 and the
number of pre-forked instances). For example, if 4 trappers are
running, the value is between 1 and 4.

Valid states are:
busy - process is in busy state, for example, processing request
(default).
idle - process is in idle state doing nothing.

Examples:
=> zabbix[process,poller,avg,busy] → average time of poller
processes spent doing something during the last minute
=> zabbix[process,"icmp pinger",max,busy] → maximum time
spent doing something by any ICMP pinger process during the
last minute
=> zabbix[process,"history syncer",2,busy] → time spent doing
something by history syncer number 2 during the last minute
=> zabbix[process,trapper,count] → amount of currently
running trapper processes
### zabbix[proxy.<name>,<param>]

**Information about Zabbix proxy.**

**name:** proxy name

Valid values for **param** are:

- **lastaccess** - timestamp of last heartbeat message received from proxy
- **delay** - how long collected values are unsent, calculated as "proxy delay" (difference between the current proxy time and the timestamp of the oldest unsent value on proxy) + ("current server time" - "proxy lastaccess")

Example:

```shell
$ => zabbix[proxy,"Germany",lastaccess]
```

**fuzzytime()** function can be used to check availability of proxies.

This item is always processed by Zabbix server regardless of host location (on server or proxy).

### zabbix[proxy_history]

**Number of values in the proxy history table waiting to be sent to the server.**

(not supported on server)
<table>
<thead>
<tr>
<th>Key</th>
<th>Number of monitored items in the queue which are delayed at least by</th>
<th>Integer from default: 6 seconds to default: infinity</th>
<th>Time-unit symbols (s,m,h,d,w) are supported for these parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;from&gt; seconds but less than by &lt;to&gt; seconds.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>zabbix[rcache,&lt;cache&gt;,&lt;mode&gt;]</strong></td>
<td>Availability statistics of Zabbix configuration cache.</td>
<td>Integer (for size); float (for percentage).</td>
<td>cache: buffer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Valid modes are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>total - total size of buffer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>free - size of free buffer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>pfree - percentage of free buffer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>used - size of used buffer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>pused - percentage of used buffer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>pused mode is supported since Zabbix 4.0.0.</td>
</tr>
<tr>
<td><strong>zabbix[requiredperformance]</strong></td>
<td>Required performance of Zabbix server or Zabbix proxy in new values per second expected.</td>
<td>Float Approximately correlates with &quot;Required server performance, new values per second&quot; in Reports → System information.</td>
<td></td>
</tr>
</tbody>
</table>
**Key**

<table>
<thead>
<tr>
<th>zabbix[stats,&lt;ip&gt;,&lt;port&gt;]</th>
<th>JSON object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote Zabbix server or proxy internal metrics.</td>
<td></td>
</tr>
<tr>
<td>JSON object.</td>
<td></td>
</tr>
<tr>
<td>ip - IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1)</td>
<td></td>
</tr>
<tr>
<td>port - port of server/proxy to be remotely queried (default is 10051)</td>
<td></td>
</tr>
<tr>
<td>Note that the stats request will only be accepted from the addresses listed in the <code>StatsAllowedIP</code> server/proxy parameter on the target instance.</td>
<td></td>
</tr>
<tr>
<td>A selected set of internal metrics is returned by this item. For details, see Remote monitoring of Zabbix stats.</td>
<td></td>
</tr>
<tr>
<td>Supported since 4.2.0.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>zabbix[stats,&lt;ip&gt;,&lt;port&gt;,queue,&lt;from&gt;,&lt;to&gt;]</th>
<th>JSON object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote Zabbix server or proxy internal queue metrics (see zabbix[queue,&lt;from&gt;,&lt;to&gt;]).</td>
<td></td>
</tr>
<tr>
<td>JSON object.</td>
<td></td>
</tr>
<tr>
<td>ip - IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1)</td>
<td></td>
</tr>
<tr>
<td>port - port of server/proxy to be remotely queried (default is 10051)</td>
<td></td>
</tr>
<tr>
<td>from - delayed by at least (default is 6 seconds)</td>
<td></td>
</tr>
<tr>
<td>to - delayed by at most (default is infinity)</td>
<td></td>
</tr>
<tr>
<td>Note that the stats request will only be accepted from the addresses listed in the <code>StatsAllowedIP</code> server/proxy parameter on the target instance.</td>
<td></td>
</tr>
<tr>
<td>Supported since 4.2.0.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>zabbix[tcache,cache,&lt;parameter&gt;]</th>
<th>Integer (for size); float (for percentage).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effectiveness statistics of the Zabbix trend function cache.</td>
<td></td>
</tr>
<tr>
<td>Valid parameters are: all - total cache requests (default)</td>
<td></td>
</tr>
<tr>
<td>hits - cache hits</td>
<td></td>
</tr>
<tr>
<td>phits - percentage of cache hits</td>
<td></td>
</tr>
<tr>
<td>misses - cache misses</td>
<td></td>
</tr>
<tr>
<td>pmisses - percentage of cache misses</td>
<td></td>
</tr>
<tr>
<td>items - the number of cached items</td>
<td></td>
</tr>
<tr>
<td>requests - the number of cached requests</td>
<td></td>
</tr>
<tr>
<td>pitems - percentage of cached items from cached items + requests. Low percentage most likely means that the cache size can be reduced.</td>
<td></td>
</tr>
<tr>
<td>Supported since 5.4.0.</td>
<td></td>
</tr>
<tr>
<td>(not supported on proxy)</td>
<td></td>
</tr>
</tbody>
</table>
### Key

<table>
<thead>
<tr>
<th>Number of enabled triggers in Zabbix database, with all items enabled on enabled hosts.</th>
<th>Integer (not supported on proxy)</th>
</tr>
</thead>
</table>

#### zabbix[uptime]

Uptime of Zabbix server or Zabbix proxy process in seconds.

- **Integer**

#### zabbix[vcache,buffer,<mode>]

Availability statistics of Zabbix value cache.

- **Integer (for size); float (for percentage).**

**Valid modes are:**
- total - total size of buffer
- free - size of free buffer
- pfree - percentage of free buffer
- used - size of used buffer
- pused - percentage of used buffer

- **(not supported on proxy)**

#### zabbix[vcache,cache,<parameter>]

Effectiveness statistics of Zabbix value cache.

- **Integer**

**With the mode parameter:**
- 0 - normal mode,
- 1 - low memory mode

**Valid parameter values are:**
- requests - total number of requests
- hits - number of cache hits (history values taken from the cache)
- misses - number of cache misses (history values taken from the database)
- mode - value cache operating mode

**This item is supported since Zabbix 2.2.0 and the mode parameter since Zabbix 3.0.0.**

- **(not supported on proxy)**

Once the low memory mode has been switched on, the value cache will remain in this state for 24 hours, even if the problem that triggered this mode is resolved sooner.

You may use this key with the Change per second preprocessing step in order to get values per second statistics.
### Key

**zabbix[version]**

Version of Zabbix server or proxy.

| String | This item is supported since Zabbix 5.0.0. Example of return value: 5.0.0beta1 |

**zabbix[vmware,buffer,<mode>]**

Availability statistics of Zabbix vmware cache.

- Integer (for size);
- float (for percentage).

Valid **modes** are:
- total - total size of buffer
- free - size of free buffer
- pfree - percentage of free buffer
- used - size of used buffer
- pused - percentage of used buffer

**zabbix[wcache,<cache>,<mode>]**

Statistics and availability of Zabbix write cache.

Specifying `<cache>` is mandatory.

<table>
<thead>
<tr>
<th>Cache Mode values (default)</th>
<th>Total number of values processed by Zabbix server or Zabbix proxy, except unsupported items.</th>
<th>Integer Counter. You may use this key with the Change per second preprocessing step in order to get values per second statistics.</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>Number of processed float values.</td>
<td>Integer Counter.</td>
</tr>
<tr>
<td>uint</td>
<td>Number of processed unsigned integer values.</td>
<td>Integer Counter.</td>
</tr>
<tr>
<td>str</td>
<td>Number of processed character/string values.</td>
<td>Integer Counter.</td>
</tr>
<tr>
<td>log</td>
<td>Number of processed log values.</td>
<td>Integer Counter.</td>
</tr>
<tr>
<td>Key</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>text</td>
<td>Number of processed text values.</td>
<td>Integer</td>
</tr>
<tr>
<td>not supported</td>
<td>Number of times item processing resulted in item becoming unsupported or keeping that state.</td>
<td>Integer</td>
</tr>
<tr>
<td>histofree</td>
<td>Percentage of free history buffer.</td>
<td>Float</td>
</tr>
<tr>
<td>(default)</td>
<td>History cache is used to store item values. A low number indicates performance problems on the database side.</td>
<td></td>
</tr>
<tr>
<td>free</td>
<td>Size of free history buffer.</td>
<td>Integer</td>
</tr>
<tr>
<td>total</td>
<td>Total size of history buffer.</td>
<td>Integer</td>
</tr>
<tr>
<td>used</td>
<td>Size of used history buffer.</td>
<td>Integer</td>
</tr>
<tr>
<td>pused</td>
<td>Percentage of used history buffer.</td>
<td>Float</td>
</tr>
<tr>
<td>indepfree</td>
<td>Percentage of free history index buffer.</td>
<td>Float</td>
</tr>
<tr>
<td>(default)</td>
<td>History index cache is used to index values stored in history cache.</td>
<td></td>
</tr>
<tr>
<td>free</td>
<td>Size of free history index buffer.</td>
<td>Integer</td>
</tr>
<tr>
<td>total</td>
<td>Total size of history index buffer.</td>
<td>Integer</td>
</tr>
<tr>
<td>used</td>
<td>Size of used history index buffer.</td>
<td>Integer</td>
</tr>
<tr>
<td>pused</td>
<td>Percentage of used history index buffer.</td>
<td>Float</td>
</tr>
</tbody>
</table>

pused mode is supported since Zabbix 4.0.0.
### Key

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>trendfree</code> (default)</td>
<td>Percentage of free trend cache.</td>
<td>Float</td>
<td>Trend cache stores aggregate for the current hour for all items that receive data. (not supported on proxy)</td>
<td></td>
</tr>
<tr>
<td><code>free</code></td>
<td>Size of free trend buffer.</td>
<td>Integer</td>
<td></td>
<td>(not supported on proxy)</td>
</tr>
<tr>
<td><code>total</code></td>
<td>Total size of trend buffer.</td>
<td>Integer</td>
<td></td>
<td>(not supported on proxy)</td>
</tr>
<tr>
<td><code>used</code></td>
<td>Size of used trend buffer.</td>
<td>Integer</td>
<td></td>
<td>(not supported on proxy)</td>
</tr>
<tr>
<td><code>pused</code></td>
<td>Percentage of used trend buffer.</td>
<td>Float</td>
<td>pused mode is supported since Zabbix 4.0.0.</td>
<td>(not supported on proxy)</td>
</tr>
</tbody>
</table>

### 9 SSH checks

**Overview**

SSH checks are performed as agent-less monitoring. Zabbix agent is not needed for SSH checks. To perform SSH checks Zabbix server must be initially configured with SSH2 support (libssh2 or libssh). See also: **Requirements**. Only libssh is supported starting with RHEL/CentOS 8.

**Configuration**

**Passphrase authentication**

SSH checks provide two authentication methods, a user/password pair and key-file based. If you do not intend to use keys, no additional configuration is required, besides linking libssh2/libssh to Zabbix, if you’re building from source.

**Key file authentication**

To use key based authentication for SSH items, certain changes to the server configuration are required.

Open the Zabbix server configuration file (zabbix_server.conf) as root and look for the following line:

```
# SSHKeyLocation=
```

Uncomment it and set full path to a folder where public and private keys will be located:

```
SSHKeyLocation=/home/zabbix/.ssh
```

Save the file and restart zabbix_server afterwards.

/home/zabbix here is the home directory for the zabbix user account and .ssh is a directory where by default public and private keys will be generated by a ssh-keygen command inside the home directory. Usually installation packages of zabbix-server from different OS distributions create zabbix user account with a home directory in not very well-known places (as for system accounts), e. g. /var/lib/zabbix.

Before starting to generate the keys, an approach to reallocate the home directory to a better known place (intuitively expected) could be considered. This will correspond with the SSHKeyLocation Zabbix server configuration parameter mentioned above.

These steps can be skipped if zabbix account has been added manually according to the installation section because in this case most likely the home directory is already located at /home/zabbix.

To change the setting for the zabbix user account all working processes which are using it have to be stopped:

```
# service zabbix-agent stop
# service zabbix-server stop
```

To change the home directory location with an attempt to move it (if it exists) a command should be executed:
It's absolutely possible that a home directory did not exist in the old place (in the CentOS for example), so it should be created at the new place. A safe attempt to do that is:

```bash
# test -d /home/zabbix || mkdir /home/zabbix
```

To be sure that all is secure, additional commands could be executed to set permissions to the home directory:

```bash
# chown zabbix:zabbix /home/zabbix
# chmod 700 /home/zabbix
```

Previously stopped processes now can be started again:

```bash
# service zabbix-agent start
# service zabbix-server start
```

Now steps to generate public and private keys can be performed by a command:

```bash
# sudo -u zabbix ssh-keygen -t rsa
```

Generating public/private rsa key pair.
Enter file in which to save the key (/home/zabbix/.ssh/id_rsa):

Created directory '/home/zabbix/.ssh'.
Enter passphrase (empty for no passphrase):

Your identification has been saved in /home/zabbix/.ssh/id_rsa.
Your public key has been saved in /home/zabbix/.ssh/id_rsa.pub.
The key fingerprint is:
90:af:e4:c7:e3:f0:2e:5a:8d:ab:48:a2:0c:92:30:b9 zabbix@it0
The key's randomart image is:

```
+++[ RSA 2048]-----+
|              |
|          .   |
|          .   |
|+          .  |
|o           .|
|E          *   |
|*           o .|
|... oo.o+    |
+-----------------+
```

Note: public and private keys (id_rsa.pub and id_rsa respectively) have been generated by default in the /home/zabbix/.ssh directory which corresponds to the Zabbix server SSHKeyLocation configuration parameter.

Key types other than "rsa" may be supported by the ssh-keygen tool and SSH servers but they may not be supported by libssh2, used by Zabbix.

Shell configuration form

This step should be performed only once for every host that will be monitored by SSH checks.

By using the following command the **public** key file can be installed on a remote host 10.10.10.10 so that then SSH checks can be performed with a root account:

```bash
# sudo -u zabbix ssh-copy-id root@10.10.10.10
```

The authenticity of host '10.10.10.10 (10.10.10.10)' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.10.10' (RSA) to the list of known hosts.
root@10.10.10.10's password:
Now try logging into the machine, with "ssh 'root@10.10.10.10'", and check in:
.ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.

Now it's possible to check the SSH login using the default private key (/home/zabbix/.ssh/id_rsa) for zabbix user account:

```bash
# sudo -u zabbix ssh root@10.10.10.10
```

If the login is successful, then the configuration part in the shell is finished and remote SSH session can be closed.

**Item configuration**
Actual command(s) to be executed must be placed in the *Executed script* field in the item configuration.

Multiple commands can be executed one after another by placing them on a new line. In this case returned values also will be formatted as multi lined.

All mandatory input fields are marked with a red asterisk.

The fields that require specific information for SSH items are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Select <strong>SSH agent</strong> here.</td>
<td>&lt;unique short description&gt; is required and should be unique for all SSH items per host</td>
</tr>
<tr>
<td>Key</td>
<td>Unique (per host) item key in format <strong>ssh.run[&lt;unique short description&gt;,&lt;ip&gt;,&lt;port&gt;,&lt;encoding&gt;]</strong></td>
<td>Default port is 22, not the port specified in the interface to which this item is assigned</td>
</tr>
<tr>
<td>Authentication</td>
<td>One of the “Password” or “Public key”</td>
<td></td>
</tr>
<tr>
<td>method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User name</td>
<td>User name to authenticate on remote host. Required</td>
<td>Example: id_rsa.pub - default public key file name generated by a command <strong>ssh-keygen</strong></td>
</tr>
<tr>
<td>Public key file</td>
<td>File name of public key if Authentication method is “Public key”. Required</td>
<td>Example: id_rsa - default private key file name</td>
</tr>
<tr>
<td>Private key file</td>
<td>File name of private key if Authentication method is “Public key”. Required</td>
<td></td>
</tr>
<tr>
<td>Password or Key</td>
<td>Password to authenticate or Passphrase if it was used for the private key</td>
<td>Leave the Key passphrase field empty if passphrase was not used</td>
</tr>
<tr>
<td>passphrase</td>
<td></td>
<td>See also <strong>known issues</strong> regarding passphrase usage</td>
</tr>
<tr>
<td>Executed script</td>
<td>Executed shell command(s) using SSH remote session</td>
<td>Examples: date +%s service mysql-server status ps auxww</td>
</tr>
</tbody>
</table>
libssh2 library may truncate executable scripts to ~32kB.

10 Telnet checks

Overview
Telnet checks are performed as agent-less monitoring. Zabbix agent is not needed for Telnet checks.

Configurable fields
Actual command(s) to be executed must be placed in the **Executed script** field in the item configuration. Multiple commands can be executed one after another by placing them on a new line. In this case returned value also will be formatted as multi lined.

Supported characters that the shell prompt can end with:
- $  
- #  
- >  
- %

A telnet prompt line which ended with one of these characters will be removed from the returned value, but only for the first command in the commands list, i.e. only at a start of the telnet session.

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>telnet.run[&lt;unique short description&gt;,&lt;ip&gt;,&lt;port&gt;,&lt;encoding&gt;]</td>
<td>Run a command on a remote device using telnet connection</td>
</tr>
</tbody>
</table>

If a telnet check returns a value with non-ASCII characters and in non-UTF8 encoding then the <encoding> parameter of the key should be properly specified. See **encoding of returned values** page for more details.

11 External checks

Overview
External check is a check executed by Zabbix server by running a shell script or a binary. However, when hosts are monitored by a Zabbix proxy, the external checks are executed by the proxy.

External checks do not require any agent running on a host being monitored.

The syntax of the item key is:

```
script[<parameter1>,<parameter2>,...]
```

Where:

<table>
<thead>
<tr>
<th>ARGUMENT</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>script</td>
<td>Name of a shell script or a binary.</td>
</tr>
<tr>
<td>parameter(s)</td>
<td>Optional command line parameters.</td>
</tr>
</tbody>
</table>

If you don’t want to pass any parameters to the script you may use:

```
script[] or script
```

Zabbix server will look in the directory defined as the location for external scripts (parameter 'ExternalScripts' in Zabbix server configuration file) and execute the command. The command will be executed as the user Zabbix server runs as, so any access permissions or environment variables should be handled in a wrapper script, if necessary, and permissions on the command should allow that user to execute it. Only commands in the specified directory are available for execution.

Do not overuse external checks! As each script requires starting a fork process by Zabbix server, running many scripts can decrease Zabbix performance a lot.

Usage example
Executing the script `check_oracle.sh` with the first parameters `-h`. The second parameter will be replaced by IP address or DNS name, depending on the selection in the host properties.
check_oracle.sh["-h","{HOST.CONN}"]

Assuming host is configured to use IP address, Zabbix will execute:

check_oracle.sh '-h' '192.168.1.4'

External check result

The return value of the check is standard output together with standard error (the full output with trimmed trailing whitespace is returned since Zabbix 2.0).

A text (character, log or text type of information) item will not become unsupported in case of standard error output.

In case the requested script is not found or Zabbix server has no permissions to execute it, the item will become unsupported and corresponding error message will be set. In case of a timeout, the item will be marked as unsupported as well, an according error message will be displayed and the forked process for the script will be killed.

12 Trapper items

Overview

Trapper items accept incoming data instead of querying for it.

It is useful for any data you might want to “push” into Zabbix.

To use a trapper item you must:

• have a trapper item set up in Zabbix
• send in the data into Zabbix

Configuration

Item configuration

To configure a trapper item:

• Go to: Configuration → Hosts
• Click on Items in the row of the host
• Click on Create item
• Enter parameters of the item in the form

<table>
<thead>
<tr>
<th>Item</th>
<th>Tags</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Name</td>
<td>Trapper Item</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Zabbix trapper</td>
<td></td>
</tr>
<tr>
<td>* Key</td>
<td>trap</td>
<td></td>
</tr>
<tr>
<td>Type of information</td>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>* History storage period</td>
<td>Do not keep history</td>
<td>Storage period</td>
</tr>
</tbody>
</table>

All mandatory input fields are marked with a red asterisk.

The fields that require specific information for trapper items are:

- **Type** Select Zabbix trapper here.
- **Key** Enter a key that will be used to recognize the item when sending in data.
- **Type of information** Select the type of information that will correspond the format of data that will be sent in.
Allowed hosts

List of comma delimited IP addresses, optionally in CIDR notation, or hostnames. If specified, incoming connections will be accepted only from the hosts listed here. If IPv6 support is enabled then ‘127.0.0.1’, ‘::127.0.0.1’, ‘::ffff:127.0.0.1’ are treated equally and ‘::/0‘ will allow any IPv4 or IPv6 address. ‘0.0.0.0/0‘ can be used to allow any IPv4 address. Note, that “IPv4-compatible IPv6 addresses” (0000::/96 prefix) are supported but deprecated by RFC4291.

Example: Server=127.0.0.1, 192.168.1.0/24, 192.168.3.1-255, 192.168.1-10.1-255, ::1,2001:db8::/32, zabbix.domain
Spaces and user macros are allowed in this field since Zabbix 2.2.0.
Host macros: {HOST.HOST}, {HOST.NAME}, {HOST.IP}, {HOST.DNS}, {HOST.CONN} are allowed in this field since Zabbix 4.0.2.

You may have to wait up to 60 seconds after saving the item until the server picks up the changes from a configuration cache update, before you can send in values.

Sending data

In the simplest of cases, we may use zabbix_sender utility to send in some ‘test value’:

zabbix_sender -z <server IP address> -p 10051 -s "New host" -k trap -o "test value"

To send in the value we use these keys:
-z - to specify Zabbix server IP address
-p - to specify Zabbix server port number (10051 by default)
-s - to specify the host (make sure to use the ‘technical’ host name here, instead of the ‘visible’ name)
-k - to specify the key of the item we just defined
-o - to specify the actual value to send

Zabbix trapper process does not expand macros used in the item key in attempt to check corresponding item key existence for targeted host.

Display

This is the result in Monitoring → Latest data:

Latest data

<table>
<thead>
<tr>
<th>Host</th>
<th>Name</th>
<th>Last check</th>
<th>Last value</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>New host</td>
<td>Trapper item</td>
<td>05/04/2021 10:56:1...</td>
<td>last value</td>
<td></td>
</tr>
</tbody>
</table>

Note that if a single numeric value is sent in, the data graph will show a horizontal line to the left and to the right of the time point of the value.

13 JMX monitoring

Overview

JMX monitoring can be used to monitor JMX counters of a Java application.

JMX monitoring has native support in Zabbix in the form of a Zabbix daemon called “Zabbix Java gateway”, introduced since Zabbix 2.0.

To retrieve the value of a particular JMX counter on a host, Zabbix server queries the Zabbix Java gateway, which in turn uses the JMX management API to query the application of interest remotely.

For more details and setup see the Zabbix Java gateway section.

Communication between Java gateway and the monitored JMX application should not be firewalled.

Enabling remote JMX monitoring for Java application
A Java application does not need any additional software installed, but it needs to be started with the command-line options specified below to have support for remote JMX monitoring.

As a bare minimum, if you just wish to get started by monitoring a simple Java application on a local host with no security enforced, start it with these options:

```java
java \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.ssl=false \
-Dcom.sun.management.jmxremote.registry.ssl=false \
-jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar
```

This makes Java listen for incoming JMX connections on port 12345, from local host only, and tells it not to require authentication or SSL.

If you want to allow connections on another interface, set the `-Djava.rmi.server.hostname` parameter to the IP of that interface.

If you wish to be more stringent about security, there are many other Java options available to you. For instance, the next example starts the application with a more versatile set of options and opens it to a wider network, not just local host.

```java
java \
-Djava.rmi.server.hostname=192.168.3.14 \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.ssl=true \
-Dcom.sun.management.jmxremote.password.file=/etc/java-6-openjdk/management/jmxremote.password \
-Dcom.sun.management.jmxremote.access.file=/etc/java-6-openjdk/management/jmxremote.access \
-Dcom.sun.management.jmxremote.registry.ssl=true \
-Djavax.net.ssl.keyStore=$YOUR_KEY_STORE \
-Djavax.net.ssl.keyStorePassword=$YOUR_KEY_STORE_PASSWORD \
-Djavax.net.ssl.trustStore=$YOUR_TRUST_STORE \
-Djavax.net.ssl.trustStorePassword=$YOUR_TRUST_STORE_PASSWORD \
-Dcom.sun.management.jmxremote.ssl.need.client.auth=true \
-jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar
```

Most (if not all) of these settings can be specified in `/etc/java-6-openjdk/management/management.properties` (or wherever that file is on your system).

Note that if you wish to use SSL, you have to modify `startup.sh` script by adding `-Djavax.net.ssl.*` options to Java gateway, so that it knows where to find key and trust stores.

See [Monitoring and Management Using JMX](#) for a detailed description.

Configuring JMX interfaces and items in Zabbix frontend

With Java gateway running, server knowing where to find it and a Java application started with support for remote JMX monitoring, it is time to configure the interfaces and items in Zabbix GUI.

Configuring JMX interface

You begin by creating a JMX-type interface on the host of interest.

<table>
<thead>
<tr>
<th>Host</th>
<th>Templates</th>
<th>IPMI</th>
<th>Tags</th>
<th>Macros</th>
<th>Inventory</th>
<th>Encryption</th>
<th>Value mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Host name</td>
<td>JMX host</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visible name</td>
<td>JMX host</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Groups</td>
<td>Java (new)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interfaces</td>
<td>Type IP address</td>
<td>DNS name</td>
<td>Connect to</td>
<td>Port</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agent</td>
<td>127.0.0.1</td>
<td></td>
<td>IP DNS</td>
<td>10050</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JMX</td>
<td>127.0.0.1</td>
<td></td>
<td>IP DNS</td>
<td>12345</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
All mandatory input fields are marked with a red asterisk.

Adding JMX agent item

For each JMX counter you are interested in you add **JMX agent** item attached to that interface.

The key in the screenshot below says `jmx["java.lang:type=Memory","HeapMemoryUsage.used"]`.

<table>
<thead>
<tr>
<th>Item</th>
<th>Tags</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Name</td>
<td>Used heap memory</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>JMX agent</td>
<td></td>
</tr>
<tr>
<td>* Key</td>
<td><code>jmx[&quot;java.lang:type=Memory&quot;,&quot;HeapMemoryUsage.used&quot;]</code></td>
<td></td>
</tr>
<tr>
<td>Type of information</td>
<td>Numeric (unsigned)</td>
<td></td>
</tr>
<tr>
<td>* Host interface</td>
<td>127.0.0.1:12345</td>
<td></td>
</tr>
<tr>
<td>* JMX endpoint</td>
<td><code>service:jmx:rmi:///jndi/rmi://[HOST.CONN]:[HOST.PORT]/jmxrmi</code></td>
<td></td>
</tr>
<tr>
<td>User name</td>
<td><code>{JMX_USERNAME}</code></td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td><code>{JMX_PASSWORD}</code></td>
<td></td>
</tr>
<tr>
<td>Units</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All mandatory input fields are marked with a red asterisk.

The fields that require specific information for JMX items are:

- **Type**
  - Set **JMX agent** here.

- **Key**
  - The `jmx[]` item key contains three parameters:
    - **object name** - the object name of an MBean
    - **attribute name** - an MBean attribute name with optional composite data field names separated by dots
    - **unique short description** - a unique description that allows multiple JMX items with the same object name and attribute name on the host (optional)
  - See below for more detail on JMX item keys.
  - Since Zabbix 3.4, you may discover MBeans and MBean attributes using a `jmx.discovery[]` low-level discovery item.

- **JMX endpoint**
  - You may specify a custom JMX endpoint. Make sure that JMX endpoint connection parameters match the JMX interface. This can be achieved by using `{HOST.*}` macros as done in the default JMX endpoint.
  - This field is supported since 3.4.0. `{HOST.*}` macros and user macros are supported.

- **User name**
  - Specify the user name, if you have configured authentication on your Java application.
  - User macros are supported.

- **Password**
  - Specify the password, if you have configured authentication on your Java application.
  - User macros are supported.

If you wish to monitor a Boolean counter that is either “true” or “false”, then you specify type of information as “Numeric (unsigned)” and select “Boolean to decimal” preprocessing step in the Preprocessing tab. Server will store Boolean values as 1 or 0, respectively.

**JMX item keys in more detail**

**Simple attributes**

An MBean object name is nothing but a string which you define in your Java application. An attribute name, on the other hand, can be more complex. In case an attribute returns primitive data type (an integer, a string etc.) there is nothing to worry about, the key will look like this:

`jmx[com.example:Type=Hello,weight]`
In this example an object name is "com.example:Type=Hello", attribute name is "weight" and probably the returned value type should be "Numeric (float)".

Attributes returning composite data

It becomes more complicated when your attribute returns composite data. For example: your attribute name is "apple" and it returns a hash representing its parameters, like "weight", "color" etc. Your key may look like this:

```
jmx[com.example:Type=Hello,apple.weight]
```

This is how an attribute name and a hash key are separated, by using a dot symbol. Same way, if an attribute returns nested composite data the parts are separated by a dot:

```
jmx[com.example:Type=Hello,fruits.apple.weight]
```

Attributes returning tabular data

Tabular data attributes consist of one or multiple composite attributes. If such an attribute is specified in the attribute name parameter then this item value will return the complete structure of the attribute in JSON format. The individual element values inside the tabular data attribute can be retrieved using preprocessing.

Tabular data attribute example:

```
jmx[com.example:Type=Hello,foodinfo]
```

Item value:

```
[
  {
    "a": "apple",
    "b": "banana",
    "c": "cherry"
  },
  {
    "a": "potato",
    "b": "lettuce",
    "c": "onion"
  }
]
```

Problem with dots

So far so good. But what if an attribute name or a hash key contains dot symbol? Here is an example:

```
jmx[com.example:Type=Hello,all.fruits.apple.weight]
```

That's a problem. How to tell Zabbix that attribute name is "all.fruits", not just "all"? How to distinguish a dot that is part of the name from the dot that separates an attribute name and hash keys?

Before 2.0.4 Zabbix java gateway was unable to handle such situations and users were left with UNSUPPORTED items. Since 2.0.4 this is possible, all you need to do is to escape the dots that are part of the name with a backslash:

```
jmx[com.example:Type=Hello,all\.fruits.apple.weight]
```

Same way, if your hash key contains a dot you escape it:

```
jmx[com.example:Type=Hello,all\.fruits.apple.total\.weight]
```

Other issues

A backslash character in an attribute name should be escaped:

```
jmx[com.example:type=Hello,c:\documents]
```

For handling any other special characters in JMX item key, please see the item key format section.

This is actually all there is to it. Happy JMX monitoring!

Non-primitive data types

Since Zabbix 4.0.0 it is possible to work with custom MBeans returning non-primitive data types, which override the `toString()` method.

```
Using custom endpoint with JBoss EAP 6.4
```

Custom endpoints allow working with different transport protocols other than the default RMI.
To illustrate this possibility, let's try to configure JBoss EAP 6.4 monitoring as an example. First, let's make some assumptions:

- You have already installed Zabbix Java gateway. If not, then you can do it in accordance with the documentation.
- Zabbix server and Java gateway are installed with the prefix /usr/local/
- JBoss is already installed in /opt/jboss-eap-6.4/ and is running in standalone mode
- We shall assume that all these components work on the same host
- Firewall and SELinux are disabled (or configured accordingly)

Let's make some simple settings in zabbix_server.conf:

```
JavaGateway=127.0.0.1
StartJavaPollers=5
```

And in the zabbix_java/settings.sh configuration file (or zabbix_java_gateway.conf):

```
START_POLLERS=5
```

Check that JBoss listens to its standard management port:

```
$ netstat -natp | grep 9999
tcp 0 0 127.0.0.1:9999 0.0.0.0:* LISTEN 10148/java
```

Now let's create a host with JMX interface 127.0.0.1:9999 in Zabbix.

```
<table>
<thead>
<tr>
<th>Host</th>
<th>Templates</th>
<th>IP/M</th>
<th>Tags</th>
<th>Macros</th>
<th>Inventory</th>
<th>Encryption</th>
<th>Value mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Host name</td>
<td>JBoss</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visible name</td>
<td>JBoss</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Groups</td>
<td>Java (new) X</td>
<td>Select</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Interaces

<table>
<thead>
<tr>
<th>Type</th>
<th>IP address</th>
<th>DNS name</th>
<th>Connect to</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>127.0.0.1</td>
<td></td>
<td>IP</td>
<td>10050</td>
</tr>
<tr>
<td>JMX</td>
<td>127.0.0.1</td>
<td></td>
<td>IP</td>
<td>9999</td>
</tr>
</tbody>
</table>
```

As we know that this version of JBoss uses the JBoss Remoting protocol instead of RMI, we may mass update the JMX endpoint parameter for items in our JMX template accordingly:

```
service:jmx:remoting-jmx://{HOST.CONN}:{HOST.PORT}
```

Let's update the configuration cache:

```
$ /usr/local/sbin/zabbix_server -R config_cache_reload
```

Note that you may encounter an error first.
"Unsupported protocol: remoting-jmx" means that Java gateway does not know how to work with the specified protocol. That can be fixed by creating a ~/needed_modules.txt file with the following content:

```
jboss-as-remoting
jboss-logging
jboss-logmanager
jboss-marshalling
jboss-remoting
jboss-sasl
jcl-over-slf4j
jul-to-slf4j-stub
log4j-jboss-logmanager
remoting-jmx
slf4j-api
xnio-api
xnio-nio
```

and then executing the command:

```
$ for i in $(cat ~/needed_modules.txt); do find /opt/jboss-eap-6.4 -iname $i*.jar -exec cp {} /usr/local/sbin/zabbix_java/lib/ \; done
```

Thus, Java gateway will have all the necessary modules for working with jmx-remoting. What's left is to restart the Java gateway, wait a bit and if you did everything right, see that JMX monitoring data begin to arrive in Zabbix (see also: Latest data).

### 14 ODBC monitoring

**Overview**

ODBC monitoring corresponds to the Database monitor item type in the Zabbix frontend.

ODBC is a C programming language middle-ware API for accessing database management systems (DBMS). The ODBC concept was developed by Microsoft and later ported to other platforms.

Zabbix may query any database, which is supported by ODBC. To do that, Zabbix does not directly connect to the databases, but uses the ODBC interface and drivers set up in ODBC. This function allows for more efficient monitoring of different databases for multiple purposes - for example, checking specific database queues, usage statistics and so on. Zabbix supports unixODBC, which is one of the most commonly used open source ODBC API implementations.

See also the known issues for ODBC checks.

**Installing unixODBC**

The suggested way of installing unixODBC is to use the Linux operating system default package repositories. In the most popular Linux distributions unixODBC is included in the package repository by default. If it's not available, it can be obtained at the unixODBC homepage: [http://www.unixodbc.org/download.html](http://www.unixodbc.org/download.html).
Installing unixODBC on RedHat/Fedora based systems using the yum package manager:

```
shell> yum -y install unixODBC unixODBC-devel
```

Installing unixODBC on SUSE based systems using the zypper package manager:

```
# zypper in unixODBC-devel
```

The unixODBC-devel package is needed to compile Zabbix with unixODBC support.

Installing unixODBC drivers

A unixODBC database driver should be installed for the database, which will be monitored. unixODBC has a list of supported databases and drivers: [http://www.unixodbc.org/drivers.html](http://www.unixodbc.org/drivers.html). In some Linux distributions database drivers are included in package repositories. Installing MySQL database driver on RedHat/Fedora based systems using the yum package manager:

```
shell> yum install mysql-connector-odbc
```

Installing MySQL database driver on SUSE based systems using the zypper package manager:

```
zypper in MyODBC-unixODBC
```

Configuring unixODBC

ODBC configuration is done by editing the `odbcinst.ini` and `odbc.ini` files. To verify the configuration file location, type:

```
shell> odbcinst -j
```

`odbcinst.ini` is used to list the installed ODBC database drivers:

```
[mysql]
Description = ODBC for MySQL
Driver    = /usr/lib/libmyodbc5.so
```

Parameter details:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysql</td>
<td>Database driver name.</td>
</tr>
<tr>
<td>Description</td>
<td>Database driver description.</td>
</tr>
<tr>
<td>Driver</td>
<td>Database driver library location.</td>
</tr>
</tbody>
</table>

`odbc.ini` is used to define data sources:

```
[test]
Description = MySQL test database
Driver      = mysql
Server      = 127.0.0.1
User        = root
Password    =
Port        = 3306
Database    = zabbix
```

Parameter details:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>test</td>
<td>Data source name (DSN).</td>
</tr>
<tr>
<td>Description</td>
<td>Data source description.</td>
</tr>
<tr>
<td>Driver</td>
<td>Database driver name - as specified in odbcinst.ini</td>
</tr>
<tr>
<td>Server</td>
<td>Database server IP/DNS.</td>
</tr>
<tr>
<td>User</td>
<td>Database user for connection.</td>
</tr>
<tr>
<td>Password</td>
<td>Database user password.</td>
</tr>
<tr>
<td>Port</td>
<td>Database connection port.</td>
</tr>
<tr>
<td>Database</td>
<td>Database name.</td>
</tr>
</tbody>
</table>

To verify if ODBC connection is working successfully, a connection to database should be tested. That can be done with the `isql` utility (included in the unixODBC package):
shell> isql test
+---------------------------------------+
| Connected! |
| sql-statement |
| help [tablename] |
| quit |
+---------------------------------------+
SQL>

Compiling Zabbix with ODBC support

To enable ODBC support, Zabbix should be compiled with the following flag:

```
--with-unixodbc[]=ARG
```

use odbc driver against unixODBC package

See more about Zabbix installation from the [source code](#).

Item configuration in Zabbix frontend

Configure a database monitoring item.

<table>
<thead>
<tr>
<th>Item</th>
<th>Tags</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>MySQL host count</td>
<td></td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Database monitor</td>
<td></td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td><code>dt.odbc.select[mysql-simple-check.test]</code></td>
<td></td>
</tr>
<tr>
<td><strong>Type of information</strong></td>
<td>Numeric (unsigned)</td>
<td></td>
</tr>
<tr>
<td><strong>User name</strong></td>
<td>zabbix</td>
<td></td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SQL query</strong></td>
<td><code>select count(*) from hosts</code></td>
<td></td>
</tr>
</tbody>
</table>

All mandatory input fields are marked with a red asterisk.

Specifically for database monitoring items you must enter:

| Type | Select Database monitor here. |
Key Enter one of the two supported item keys:

`db.odbc.select(<unique short description>,<dsn>,<connection string>)` - this item is designed to return one value, i.e. the first column of the first row of the SQL query result. If a query returns more than one column, only the first column is read. If a query returns more than one line, only the first line is read.

`db.odbc.get(<unique short description>,<dsn>,<connection string>)` - this item is capable of returning multiple rows/columns in JSON format. Thus it may be used as a master item that collects all data in one system call, while JSONPath preprocessing may be used in dependent items to extract individual values. For more information, see an example of the returned format, used in low-level discovery. This item is supported since Zabbix 4.4.

The unique description will serve to identify the item in triggers, etc. Although `dsn` and `connection string` are optional parameters, at least one of them should be present. If both data source name (DSN) and connection string are defined, the DSN will be ignored.

The data source name, if used, must be set as specified in odbc.ini.

The connection string may contain driver-specific arguments.

Example (connection for MySQL ODBC driver 5):

```
=> db.odbc.get[MySQL example", "Driver=/usr/local/lib/libmyodbc5a.so;Database=master;Server=127.0.0.1;Port=3306"]
```

User name Enter the database user name

This parameter is optional if user is specified in odbc.ini.

If connection string is used, and User name field is not empty, it is appended to the connection string as `UID=<user>`.

Password Enter the database user password

This parameter is optional if password is specified in odbc.ini.

If connection string is used, and Password field is not empty, it is appended to the connection string as `PWD=<password>`.

SQL query Enter the SQL query.

Note that with the `db.odbc.select` item the query must return one value only.

Type of information It is important to know what type of information will be returned by the query, so that it is selected correctly here. With an incorrect type of information the item will turn unsupported.

**Important notes**

- Database monitoring items will become unsupported if no odbc poller processes are started in the server or proxy configuration. To activate ODBC pollers, set `StartODBCPollers` parameter in Zabbix server configuration file or, for checks performed by proxy, in Zabbix proxy configuration file.

- Zabbix does not limit the query execution time. It is up to the user to choose queries that can be executed in a reasonable amount of time.

- The `Timeout` parameter value from Zabbix server is used as the ODBC login timeout (note that depending on ODBC drivers the login timeout setting might be ignored).

- The SQL command must return a result set like any query with `select ...`. The query syntax will depend on the RDBMS which will process them. The syntax of request to a storage procedure must be started with `call` keyword.

**Error messages**

ODBC error messages are structured into fields to provide detailed information. For example:

```
Cannot execute ODBC query: [SQL_ERROR]:[42601][7][ERROR: syntax error at or near ";"; Error while executing];
```

<table>
<thead>
<tr>
<th>Native error code</th>
<th>Native error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLState</td>
<td>Zabbix message</td>
</tr>
<tr>
<td>ODBC return code</td>
<td></td>
</tr>
</tbody>
</table>

Note that the error message length is limited to 2048 bytes, so the message can be truncated. If there is more than one ODBC diagnostic record Zabbix tries to concatenate them (separated with `|`) as far as the length limit allows.

1 Recommended UnixODBC settings for MySQL

*** Red Hat Enterprise Linux/CentOS***:
# yum install mysql-connector-odbc

***Debian/Ubuntu***:

Please refer to MySQL documentation to download necessary database driver for the corresponding platform.

For some additional information please refer to: installing unixODBC.

Configuration

ODBC configuration is done by editing odbcinst.ini and odbc.ini files. These configuration files can be found in /etc folder. The file odbcinst.ini may be missing and in this case it is necessary to create it manually.

**odbcinst.ini**

```
[mysql]
Description = General ODBC for MySQL
Driver = /usr/lib64/libmyodbc5.so
Setup = /usr/lib64/libodbcmyS.so
FileUsage = 1
```

Please consider the following examples of odbc.ini configuration parameters.

- An example with a connection through an IP:

  ```ini
  [TEST_MYSQL]
  Description = MySQL database 1
  Driver = mysql
  Port = 3306
  Server = 127.0.0.1
  ```

- An example with a connection through an IP and with the use of credentials. A Zabbix database is used by default:

  ```ini
  [TEST_MYSQL_FILLED_CRED]
  Description = MySQL database 2
  Driver = mysql
  User = root
  Port = 3306
  Password = zabbix
  Database = zabbix
  Server = 127.0.0.1
  ```

- An example with a connection through a socket and with the use of credentials. A Zabbix database is used by default:

  ```ini
  [TEST_MYSQL_FILLED_CRED_SOCK]
  Description = MySQL database 3
  Driver = mysql
  User = root
  Password = zabbix
  Socket = /var/run/mysqld/mysqld.sock
  Database = zabbix
  ```

All other possible configuration parameter options can be found in MySQL official documentation web page.

Z Recommended UnixODBC settings for PostgreSQL

```
# yum install postgresql-odbc
```

- **Red Hat Enterprise Linux/CentOS**:

- **Debian/Ubuntu**:

Please refer to PostgreSQL documentation to download necessary database driver for the corresponding platform.

For some additional information please refer to: installing unixODBC.

Configuration
ODBC configuration is done by editing the `odbcinst.ini` and `odbc.ini` files. These configuration files can be found in `/etc` folder. The file `odbcinst.ini` may be missing and in this case it is necessary to create it manually.

Please consider the following examples:

**odbcinst.ini**

```ini
[postgresql]
Description = General ODBC for PostgreSQL
Driver = /usr/lib64/libodbcpsql.so
Setup = /usr/lib64/libodbcpsqlS.so
FileUsage = 1
# Since 1.6 if the driver manager was built with thread support you may add another entry to each driver entry.
# This entry alters the default thread serialization level.
Threading = 2
```

**odbc.ini**

```ini
[TEST_PSQL]
Description = PostgreSQL database 1
Driver = postgresql
#CommLog = /tmp/sql.log
Username = zbx_test
Password = zabbix
# Name of Server. IP or DNS
Servername = 127.0.0.1
# Database name
Database = zabbix
# Postmaster listening port
Port = 5432
# Database is read only
ReadOnly = No
# PostgreSQL backend protocol
# Note that when using SSL connections this setting is ignored.
# 7.4+: Use the 7.4(V3) protocol. This is only compatible with 7.4 and higher backends.
Protocol = 7.4+
# Includes the OID in SQLColumns
ShowOidColumn = No
# Fakes a unique index on OID
FakeOidIndex = No
# Row Versioning
# Allows applications to detect whether data has been modified by other users
# while you are attempting to update a row.
RowVersioning = No
# Show SystemTables
# The driver will treat system tables as regular tables in SQLTables. This is good for Access so you can see system tables.
ShowSystemTables = No
# If true, the driver automatically uses declare cursor/fetch to handle SELECT statements and keeps 100 rows in the cache.
Fetch = Yes
# Bools as Char
# Bools are mapped to SQL_CHAR, otherwise to SQL_BIT.
BoolsAsChar = Yes
# SSL mode
SSLmode = Yes
# Send to backend on connection
ConnSettings =
```

3 Recommended UnixODBC settings for Oracle

**Installation**

Please refer to Oracle documentation for all the necessary instructions.
For some additional information please refer to: Installing unixODBC.

4 Recommended UnixODBC settings for MSSQL

Installation

- **Red Hat Enterprise Linux/CentOS**:
  
  # yum -y install freetds unixODBC

- **Debian/Ubuntu**:
  
  Please refer to FreeTDS user guide to download necessary database driver for the corresponding platform.

For some additional information please refer to: installing unixODBC.

Configuration

ODBC configuration is done by editing the `odbcinst.ini` and `odbc.ini` files. These configuration files can be found in `/etc` folder. The file `odbcinst.ini` may be missing and in this case it is necessary to create it manually.

Please consider the following examples:

- **odbcinst.ini**
  
  $ vi /etc/odbcinst.ini
  
  [FreeTDS]
  
  Driver = /usr/lib64/libtdsodbc.so.0

- **odbc.ini**
  
  $ vi /etc/odbc.ini
  
  [sql1]
  
  Driver = FreeTDS
  
  Server = <SQL server 1 IP>
  
  PORT = 1433
  
  TDS_Version = 8.0

15 Dependent items

Overview

There are situations when one item gathers multiple metrics at a time or it even makes more sense to collect related metrics simultaneously, for example:

- CPU utilization of individual cores
- Incoming/outgoing/total network traffic

To allow for bulk metric collection and simultaneous use in several related items, Zabbix supports dependent items. Dependent items depend on the master item that collects their data simultaneously, in one query. A new value for the master item automatically populates the values of the dependent items. Dependent items cannot have a different update interval than the master item.

Zabbix preprocessing options can be used to extract the part that is needed for the dependent item from the master item data.

Preprocessing is managed by a preprocessing manager process, which has been added in Zabbix 3.4, along with workers that perform the preprocessing steps. All values (with or without preprocessing) from different data gatherers pass through the preprocessing manager before being added to the history cache. Socket-based IPC communication is used between data gatherers (pollers, trappers, etc) and the preprocessing process.

Zabbix server or Zabbix proxy (if host is monitored by proxy) are performing preprocessing steps and processing dependent items.

Item of any type, even dependent item, can be set as master item. Additional levels of dependent items can be used to extract smaller parts from the value of an existing dependent item.

Limitations

- Only same host (template) dependencies are allowed
- An item prototype can depend on another item prototype or regular item from the same host
- Maximum count of dependent items for one master item is limited to 29999 (regardless of the number of dependency levels)
- Maximum 3 dependency levels allowed
• Dependent item on a host with master item from template will not be exported to XML

**Item configuration**

A dependent item depends on its master item for data. That is why the **master item** must be configured (or exist) first:

- Go to: **Configuration → Hosts**
- Click on Items in the row of the host
- Click on Create item
- Enter parameters of the item in the form

<table>
<thead>
<tr>
<th>Item</th>
<th>Tags</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Name</td>
<td>Apache server status</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Zabbix agent</td>
<td></td>
</tr>
<tr>
<td>* Key</td>
<td>web.page.get[127.0.0.1/server-status]</td>
<td></td>
</tr>
<tr>
<td>Type of information</td>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>* Host interface</td>
<td>127.0.0.1:1050</td>
<td></td>
</tr>
<tr>
<td>* Update interval</td>
<td>30s</td>
<td></td>
</tr>
</tbody>
</table>

All mandatory input fields are marked with a red asterisk.

Click on Add to save the master item.

Then you can configure a **dependent item**.

<table>
<thead>
<tr>
<th>Item</th>
<th>Tags</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Name</td>
<td>Apache server uptime</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Dependent item</td>
<td></td>
</tr>
<tr>
<td>* Key</td>
<td>apache.server.uptime</td>
<td></td>
</tr>
<tr>
<td>Type of information</td>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>* Master item</td>
<td>Apache: Apache server status</td>
<td></td>
</tr>
</tbody>
</table>

All mandatory input fields are marked with a red asterisk.

The fields that require specific information for dependent items are:

- **Type**
  - Select **Dependent item** here.
- **Key**
  - Enter a key that will be used to recognize the item.
- **Master item**
  - Select the master item. Master item value will be used to populate dependent item value.
- **Type of information**
  - Select the type of information that will correspond the format of data that will be stored.

You may use item value **preprocessing** to extract the required part of the master item value.
Without preprocessing, the dependent item value will be exactly the same as the master item value.

Click on Add to save the dependent item.

A shortcut to creating a dependent item quicker is to use the wizard in the item list:

Display

In the item list dependent items are displayed with their master item name as prefix.

If a master item is deleted, so are all its dependent items.

16 HTTP agent

Overview

This item type allows data polling using the HTTP/HTTPS protocol. Trapping is also possible using Zabbix sender or Zabbix sender protocol.

HTTP item check is executed by Zabbix server. However, when hosts are monitored by a Zabbix proxy, HTTP item checks are executed by the proxy.

HTTP item checks do not require any agent running on a host being monitored.

HTTP agent supports both HTTP and HTTPS. Zabbix will optionally follow redirects (see the Follow redirects option below). Maximum number of redirects is hard-coded to 10 (using cURL option CURLOPT_MAXREDIRS).

See also known issues for when using HTTPS protocol.

Zabbix server/proxy must be initially configured with cURL (libcurl) support.

Configuration

To configure an HTTP item:

• Go to: Configuration → Hosts
• Click on Items in the row of the host
- Click on Create item
- Enter parameters of the item in the form

All mandatory input fields are marked with a red asterisk.

The fields that require specific information for HTTP items are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Select <strong>HTTP agent</strong> here.</td>
</tr>
<tr>
<td>Key</td>
<td>Enter a unique item key.</td>
</tr>
</tbody>
</table>
| URL           | URL to connect to and retrieve data. For example:  
               | https://www.example.com  
               | http://www.example.com/download  
               Domain names can be specified in Unicode characters. They are automatically  
               punycode-converted to ASCII when executing the HTTP check.  
               The Parse button can be used to separate optional query fields (like  
               ?name=Admin&password=mypassword) from the URL, moving the attributes and values into  
               Query fields for automatic URL-encoding.  
               Limited to 2048 characters.  
               Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME},  
               {ITEM.ID}, {ITEM.KEY}, {ITEM.KEY.ORIG}, user macros, low-level discovery macros.  
               This sets the **CURLOPT_URL** cURL option. |
| Query fields  | Variables for the URL (see above).  
               Specified as attribute and value pairs.  
               Values are URL-encoded automatically. Values from macros are resolved and then URL-encoded  
               automatically.  
               Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME},  
               {ITEM.ID}, {ITEM.KEY}, {ITEM.KEY.ORIG}, user macros, low-level discovery macros.  
               This sets the **CURLOPT_URL** cURL option. |
<p>| Request type  | Select request method type: GET, POST, PUT or HEAD |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>Zabbix will not spend more than the set amount of time on processing the URL (1-60 seconds). Actually this parameter defines the maximum time for making a connection to the URL and maximum time for performing an HTTP request. Therefore, Zabbix will not spend more than 2 x Timeout seconds on one check. Time suffixes are supported, e.g. 30s, 1m. Supported macros: user macros, low-level discovery macros. This sets the CURLOPT_TIMEOUT cURL option.</td>
</tr>
<tr>
<td>Request body type</td>
<td>Select the request body type: <strong>Raw data</strong> - custom HTTP request body, macros are substituted but no encoding is performed <strong>JSON data</strong> - HTTP request body in JSON format. Macros can be used as string, number, true and false; macros used as strings must be enclosed in double quotes. Values from macros are resolved and then escaped automatically. If “Content-Type” is not specified in headers then it will default to “Content-Type: application/json” <strong>XML data</strong> - HTTP request body in XML format. Macros can be used as a text node, attribute or CDATA section. Values from macros are resolved and then escaped automatically in a text node and attribute. If “Content-Type” is not specified in headers then it will default to “Content-Type: application/xml” Note that selecting XML data requires libxml2.</td>
</tr>
<tr>
<td>Request body</td>
<td>Enter the request body. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, {ITEM.KEY.ORIG}, user macros, low-level discovery macros.</td>
</tr>
<tr>
<td>Headers</td>
<td>Custom HTTP headers that will be sent when performing a request. Specified as attribute and value pairs. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, {ITEM.KEY.ORIG}, user macros, low-level discovery macros. This sets the CURLOPT_HTTPHEADER cURL option.</td>
</tr>
<tr>
<td>Required status codes</td>
<td>List of expected HTTP status codes. If Zabbix gets a code which is not in the list, the item will become unsupported. If empty, no check is performed. For example: 200,201,210-299 Supported macros in the list: user macros, low-level discovery macros. This uses the CURLOPT_HTTPHEADER cURL option.</td>
</tr>
<tr>
<td>Follow redirects</td>
<td>Mark the checkbox to follow HTTP redirects. This sets the CURLOPT_FOLLOWLOCATION cURL option.</td>
</tr>
<tr>
<td>Retrieve mode</td>
<td>Select the part of response that must be retrieved: <strong>Body</strong> - body only <strong>Headers</strong> - headers only <strong>Body and headers</strong> - body and headers</td>
</tr>
<tr>
<td>Convert to JSON</td>
<td>Headers are saved as attribute and value pairs under the “header” key. If ‘Content-Type: application/json’ is encountered then body is saved as an object, otherwise it is stored as string, for example:</td>
</tr>
</tbody>
</table>

```json
{
    "header": {
        "<key1>": "<value1>",
        "<key2>": "<value2>",
    },
    "body": <body>
}
```
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| HTTP proxy         | You can specify an HTTP proxy to use, using the format 
[protocol://[/][username[:password]@]proxy.example.com[:port]. The optional protocol:// prefix may be used to specify alternative proxy protocols (e.g. https, socks4, socks5; see documentation; the protocol prefix support was added in cURL 7.21.7). With no protocol specified, the proxy will be treated as an HTTP proxy. If you specify the wrong protocol, the connection will fail and the item will become unsupported. By default, 1080 port will be used. If specified, the proxy will overwrite proxy related environment variables like http_proxy, HTTPS_PROXY. If not specified, the proxy will not overwrite proxy-related environment variables. The entered value is passed on "as is", no sanity checking takes place. Note that only simple authentication is supported with HTTP proxy. Supported macros: (HOST.IP), (HOST.CONN), (HOST.DNS), (HOST.HOST), (HOST.NAME), (ITEM.ID), (ITEM.KEY), (ITEM.KEY.ORIG), user macros, low-level discovery macros. This sets the CURLOPT_PROXY cURL option. |
| HTTP authentication| Authentication type:  
None - no authentication used.  
Basic - basic authentication is used.  
NTLM - NTLM (Windows NT LAN Manager) authentication is used.  
Kerberos - Kerberos authentication is used. See also: Configuring Kerberos with Zabbix.  
Digest - Digest authentication is used. Selecting an authentication method will provide two additional fields for entering a user name and password, where user macros and low-level discovery macros are supported. This sets the CURLOPT_HTTPAUTH cURL option. |
| SSL verify peer    | Mark the checkbox to verify the SSL certificate of the web server. The server certificate will be automatically taken from system-wide certificate authority (CA) location. You can override the location of CA files using Zabbix server or proxy configuration parameter SSLCALocation. This sets the CURLOPT_SSL_VERIFYHOST cURL option. |
| SSL verify host    | Mark the checkbox to verify that the Common Name field or the Subject Alternate Name field of the web server certificate matches. This sets the CURLOPT_SSL_VERIFYPEER cURL option. |
| SSL certificate file | Name of the SSL certificate file used for client authentication. The certificate file must be in PEM\(^1\) format. If the certificate file contains also the private key, leave the SSL key file field empty. If the key is encrypted, specify the password in SSL key password field. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSLCertLocation. Supported macros: (HOST.IP), (HOST.CONN), (HOST.DNS), (HOST.HOST), (HOST.NAME), (ITEM.ID), (ITEM.KEY), (ITEM.KEY.ORIG), user macros, low-level discovery macros. This sets the CURLOPT_SSLCERT cURL option. |
| SSL key file       | Name of the SSL private key file used for client authentication. The private key file must be in PEM\(^1\) format. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSLKeyLocation. Supported macros: (HOST.IP), (HOST.CONN), (HOST.DNS), (HOST.HOST), (HOST.NAME), (ITEM.ID), (ITEM.KEY), (ITEM.KEY.ORIG), user macros, low-level discovery macros. This sets the CURLOPT_SSLKEY cURL option. |
| SSL key password   | SSL private key file password. Supported macros: user macros, low-level discovery macros. This sets the CURLOPT_KEYPASSWD cURL option. |
| Enable trapping    | With this checkbox marked, the item will also function as trapper item and will accept data sent to this item by Zabbix sender or using Zabbix sender protocol. |
| Allowed hosts      | Visible only if Enable trapping checkbox is marked. List of comma delimited IP addresses, optionally in CIDR notation, or hostnames. If specified, incoming connections will be accepted only from the hosts listed here. If IPv6 support is enabled then ‘127.0.0.1’, ‘::127.0.0.1’, ‘::ffff:127.0.0.1’ are treated equally and ‘::/0’ will allow any IPv4 or IPv6 address. ‘0.0.0.0/0’ can be used to allow any IPv4 address. Note, that “IPv4-compatible IPv6 addresses” (0000::/96 prefix) are supported but deprecated by RFC4291. Example: Server=127.0.0.1, 192.168.1.0/24, 192.168.3.1-255, 192.168.1-10.1-255, ::1,2001:db8::/32, zabbix.domain Spaces and user macros are allowed in this field. Host macros: (HOST.HOST), (HOST.NAME), (HOST.IP), (HOST.DNS), (HOST.CONN) are allowed in this field. |
If the HTTP proxy field is left empty, another way for using an HTTP proxy is to set proxy-related environment variables.

For HTTP - set the `http_proxy` environment variable for the Zabbix server user. For example:

```
```

For HTTPS - set the `HTTPS_PROXY` environment variable. For example:

```
```

More details are available by running a shell command: `man curl`.

[1] Zabbix supports certificate and private key files in PEM format only. In case you have your certificate and private key data in PKCS #12 format file (usually with extension *.p12 or *.pfx) you may generate the PEM file from it using the following commands:

```
openssl pkcs12 -in ssl-cert.p12 -clcerts -nokeys -out ssl-cert.pem
openssl pkcs12 -in ssl-cert.p12 -nocerts -nodes -out ssl-cert.key
```

**Examples**

**Example 1**

Send simple GET requests to retrieve data from services such as Elasticsearch:

- Create a GET item with URL: `localhost:9200/?pretty`
- Notice the response:

```
{"name": "YQ2VAY-",
"cluster_name": "elasticsearch",
"cluster_uuid": "kH4CYqh5QfqqeTsjh2F9zg",
"version": {
  "number": "6.1.3",
  "build_hash": "af51318",
  "build_date": "2018-01-26T18:22:55.523Z",
  "build_snapshot": false,
  "lucene_version": "7.1.0",
  "minimum_wire_compatibility_version": "5.6.0",
  "minimum_index_compatibility_version": "5.0.0"
},
"tagline": "You know, for search"
}
```

- Now extract the version number using a JSONPath preprocessing step: `$.version.number`

**Example 2**

Send simple POST requests to retrieve data from services such as Elasticsearch:

- Create a POST item with URL: `http://localhost:9200/str/values/_search?scroll=10s`
- Configure the following POST body to obtain the processor load (1 min average per core)

```
{  
  "query": {  
    "bool": {  
      "must": [{  
        "match": {  
          "itemid": 28275  
        }  
      }],  
      "filter": [{  
        "range": {  
          "clock": {  
            "gt": 1517565836,  
            "lte": 1517566137  
          }  
        }  
      }]}  
  }  
}
```

- Received:
Now use a JSONPath preprocessing step to get the item value: $.hits.hits[0]._source.value

Example 3

Checking if Zabbix API is alive, using apiinfo.version.

- Item configuration:
Note the use of the POST method with JSON data, setting request headers and asking to return headers only:

- Item value preprocessing with regular expression to get HTTP code:

- Checking the result in Latest data:
Example 4

Retrieving weather information by connecting to the Openweathermap public service.

- Configure a master item for bulk data collection in a single JSON:

<table>
<thead>
<tr>
<th>Item</th>
<th>Tags</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Name</td>
<td>Get weather</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>HTTP agent</td>
<td></td>
</tr>
<tr>
<td>* Key</td>
<td>get_weather.http</td>
<td></td>
</tr>
<tr>
<td>Type of information</td>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>* URL</td>
<td><a href="http://api.openweathermap.org/data/2.5/weather">http://api.openweathermap.org/data/2.5/weather</a></td>
<td></td>
</tr>
<tr>
<td>Query fields</td>
<td>Name</td>
<td>Value</td>
</tr>
<tr>
<td>units</td>
<td></td>
<td>metric</td>
</tr>
<tr>
<td>lat</td>
<td></td>
<td>$(LAT)</td>
</tr>
<tr>
<td>lon</td>
<td></td>
<td>$(LON)</td>
</tr>
<tr>
<td>APPID</td>
<td></td>
<td>$(WEATHER_APIKEY)</td>
</tr>
<tr>
<td>lang</td>
<td></td>
<td>$(WEATHER_LANG)</td>
</tr>
</tbody>
</table>

Note the usage of macros in query fields. Refer to the Openweathermap API for how to fill them.

Sample JSON returned in response to HTTP agent:

```json
{
    "body": {
        "coord": {
            "lon": 40.01,
            "lat": 29.76
        },
        "weather": [
            {
                "id": 800,
                "main": "Clear",
                "description": "clear sky",
                "icon": "h01"
            }
        ],
        "base": "stations",
        "main": {
            "temp": 293.15,
            "feels_like": 292.74,
            "temp_min": 291.25,
            "temp_max": 293.15,
            "pressure": 1013.25,
            "humidity": 100
        },
        "visibility": 10000,
        "wind": {
            "speed": 1.5,
            "deg": 180
        },
        "clouds": {
            "all": 0
        },
        "dt": 1569282841,
        "sys": {
            "type": 1,
            "id": 2034188,
            "country": "US",
            "sunrise": 1569271002,
            "sunset": 1569283427
        },
        "timezone": -28800,
        "id": 531118,
        "name": "San Francisco",
        "cod": 200
    }
}
```
"lat": 56.11,
"weather": [{
  "id": 801,
  "main": "Clouds",
  "description": "few clouds",
  "icon": "02n"
}],
"base": "stations",
"main": {
  "temp": 15.14,
  "pressure": 1012.6,
  "humidity": 66,
  "temp_min": 15.14,
  "temp_max": 15.14,
  "sea_level": 1030.91,
  "grnd_level": 1012.6
},
"wind": {
  "speed": 1.86,
  "deg": 246.001
},
"clouds": {
  "all": 20
},
"dt": 1526509427,
"sys": {
  "message": 0.0035,
  "country": "RU",
  "sunrise": 1526432608,
  "sunset": 1526491828
},
"id": 487837,
"name": "Stavrovo",
"cod": 200
}

The next task is to configure dependent items that extract data from the JSON.

- Configure a sample dependent item for humidity:

<table>
<thead>
<tr>
<th>Item</th>
<th>Tags</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Name</td>
<td></td>
<td>Humidity</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td>Dependent item</td>
</tr>
<tr>
<td>* Key</td>
<td></td>
<td>humidity</td>
</tr>
<tr>
<td>Type of information</td>
<td></td>
<td>Numeric (float)</td>
</tr>
<tr>
<td>* Master item</td>
<td></td>
<td>Apache: Get weather</td>
</tr>
</tbody>
</table>

Other weather metrics such as 'Temperature' are added in the same manner.

- Sample dependent item value preprocessing with JSONPath:
Check the result of weather data in Latest data:

Example 5

Connecting to Nginx status page and getting its metrics in bulk.

- Configure Nginx following the official guide.

- Configure a master item for bulk data collection:

Sample Nginx stub status output:
The next task is to configure dependent items that extract data.

- Configure a sample dependent item for requests per second:

```
<table>
<thead>
<tr>
<th>Item</th>
<th>Tags</th>
<th>Preprocessing 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Name</td>
<td>Client requests per second</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Dependent Item</td>
<td></td>
</tr>
<tr>
<td>* Key</td>
<td>nginx_requests_rps</td>
<td></td>
</tr>
<tr>
<td>Type of information</td>
<td>Numeric (unsigned)</td>
<td></td>
</tr>
<tr>
<td>* Master item</td>
<td>Nginx by HTTP: Nginx: Get status page</td>
<td></td>
</tr>
</tbody>
</table>
```

- Sample dependent item value preprocessing with regular expression server accepts handled requests\(\s+(\[0-9]+)\)\(\[0-9]+\)\(\[0-9]+\):

```
<table>
<thead>
<tr>
<th>Item</th>
<th>Tags</th>
<th>Preprocessing 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preprocessing steps</td>
<td>Name</td>
<td>Parameters</td>
</tr>
<tr>
<td>1: Regular expression</td>
<td>request(\s+([0-9]+))([0-9]+)([0-9]+)</td>
<td>13</td>
</tr>
<tr>
<td>2: Change per second</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- Check the complete result from stub module in Latest data:

```
<table>
<thead>
<tr>
<th>Host</th>
<th>Name</th>
<th>Last check</th>
<th>Last value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nginx (5 items)</td>
<td>2018-05-10 17:54:53</td>
<td>568</td>
</tr>
<tr>
<td></td>
<td>Accepted client connections</td>
<td>2018-05-10 17:54:53</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Active connections</td>
<td>2018-05-10 17:54:53</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Client requests per second</td>
<td>2018-05-10 17:54:53</td>
<td>0</td>
</tr>
</tbody>
</table>
|      | Get Nginx stub status | 2018-05-10 17:54:53 | HTTP/1.1 300 OK S...
|      | Handled connections per second | 2018-05-10 17:54:53 | 0 |
|      | Reading | 2018-05-10 17:54:53 | 0 |
|      | Waiting | 2018-05-10 17:54:53 | 0 |
|      | Writing | 2018-05-10 17:54:53 | 1 |
```

**17 Prometheus checks**

**Overview**

Zabbix can query metrics exposed in the Prometheus line format.

Two steps are required to start gathering Prometheus data:

- an **HTTP master item** pointing to the appropriate data endpoint, e.g. https://<prometheus host>/metrics
- dependent items using a Prometheus preprocessing option to query required data from the metrics gathered by the master item

There are two Prometheus data preprocessing options:

- Prometheus pattern - used in normal items to query Prometheus data
• Prometheus to JSON - used in normal items and for low-level discovery. In this case queried Prometheus data are returned in a JSON format.

**Bulk processing**

Bulk processing is supported for dependent items. To enable caching and indexing, the Prometheus pattern preprocessing must be the **first** preprocessing step. When Prometheus pattern is first preprocessing step then the parsed Prometheus data is cached and indexed by the first `<label>==<value>` condition in the Prometheus pattern preprocessing step. This cache is reused when processing other dependent items in this batch. For optimal performance, the first label should be the one with most different values.

If there is other preprocessing to be done before the first step, it should be moved either to the master item or to a new dependent item which would be used as the master item for the dependent items.

**Configuration**

Providing you have the HTTP master item configured, you need to create a dependent item that uses a Prometheus preprocessing step:

• Enter general dependent item parameters in the configuration form
• Go to the Preprocessing tab
• Select a Prometheus preprocessing option (Prometheus pattern or Prometheus to JSON)

<table>
<thead>
<tr>
<th>Preprocessing steps</th>
<th>Name</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prometheus pattern</td>
<td>cpu_usage_system{cpu=&quot;cp&quot;}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value</td>
</tr>
<tr>
<td></td>
<td><strong>Add</strong></td>
<td></td>
</tr>
<tr>
<td>Type of information</td>
<td>Numeric (unsigned)</td>
<td></td>
</tr>
</tbody>
</table>

The following parameters are specific to the Prometheus pattern preprocessing option:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern</td>
<td>To define the required data pattern you may use a query language that is similar to Prometheus query language (see comparison table), e.g.: <code>&lt;metric name&gt;</code> - select by metric name <code>(__name__=&quot;&lt;metric name&gt;&quot;)</code> - select by metric name <code>(__name__=~&quot;&lt;regex&gt;&quot;)</code> - select by metric name matching a regular expression <code>{&lt;label name&gt;=&quot;&lt;label value&gt;&quot;,...} - select by label name </code>{&lt;label name&gt;=~&quot;&lt;regex&gt;&quot;,...} - select by label name matching a regular expression <code>{&lt;label name&gt;=~&quot;&lt;regex&gt;&quot;\=*} ==&lt;value&gt; - select by metric value Or a combination of the above: </code>&lt;metric name&gt;{&lt;label1 name&gt;=&quot;&lt;label1 value&quot;&gt;,&lt;label2 name&gt;=&quot;&lt;regex&gt;&quot;,...} ==&lt;value&gt;`</td>
<td><code>wmi_os_physical_memory_free_bytes</code> <code>cpu_usage_system{cpu=&quot;cp-total&quot;}</code> <code>cpu_usage_system{cpu=&quot;cp~*&quot;}</code> <code>cpu_usage_system{cpu=&quot;cp-total&quot;,host=~&quot;.*&quot;}</code> <code>wmi_service_state{name=&quot;dhcp&quot;}==1</code> <code>wmi_os_timezone{timezone=~&quot;.*&quot;}==1</code></td>
</tr>
</tbody>
</table>

Label value can be any sequence of UTF-8 characters, but the backslash, double-quote and line feed characters have to be escaped as \\, \" and \n respectively; other characters shall not be escaped.
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
</table>
| Result processing | Specify whether to return the value, the label or apply the appropriate function (if the pattern matches several lines and the result needs to be aggregated):  
  - value - return metric value (error if multiple lines matched)  
  - label - return value of the label specified in the Label field (error if multiple metrics are matched)  
  - sum - return the sum of values  
  - min - return the minimum value  
  - max - return the maximum value  
  - avg - return the average value  
  - count - return the count of values  
 | See also examples of using parameters below. | 
| Output          | Define label name (optional). In this case the value corresponding to the label name is returned. This field is only available for the Prometheus pattern option. | 

### Examples of using parameters

1. The most common use case is to return the **value**. To return the value of `/var/db` from:

   ```
   node_disk_usage_bytes{path="/var/cache"} 2.1766144e+09
   node_disk_usage_bytes{path="/var/db"} 20480
   node_disk_usage_bytes{path="/var/dpkg"} 8192
   node_disk_usage_bytes{path="/var/empty"} 4096
   ```

   use the following parameters:
   - Pattern - `node_disk_usage_bytes{path="/var/db"}`
   - Result processing - select 'value'

2. You may also be interested in the **average** value of all `node_disk_usage_bytes` parameters:

   ```
   node_disk_usage_bytes
   ```

   use the following parameters:
   - Pattern - `node_disk_usage_bytes`
   - Result processing - select 'avg'

3. While Prometheus supports only numerical data, it is popular to use a workaround that allows to return the relevant textual description as well. This can be accomplished with a filter and specifying the label. So, to return the value of the 'color' label from:

   ```
   elasticsearch_cluster_health_status{cluster="elasticsearch",color="green"} 1
   elasticsearch_cluster_health_status{cluster="elasticsearch",color="red"} 0
   elasticsearch_cluster_health_status{cluster="elasticsearch",color="yellow"} 0
   ```

   use the following parameters:
   - Pattern - `elasticsearch_cluster_health_status {cluster="elasticsearch"} == 1`
   - Result processing - select 'label'
   - Label - specify 'color'

   The filter (based on the numeric value `1`) will match the appropriate row, while the label will return the health status description (currently 'green'; but potentially also 'red' or 'yellow').

### Prometheus to JSON

Data from Prometheus can be used for low-level discovery. In this case data in JSON format are needed and the Prometheus to JSON preprocessing option will return exactly that.

For more details, see [Discovery using Prometheus data](#).

### Query language comparison

The following table lists differences and similarities between PromQL and Zabbix Prometheus preprocessing query language.

<table>
<thead>
<tr>
<th>PromQL instant vector selector</th>
<th>Zabbix Prometheus preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differences</td>
<td></td>
</tr>
</tbody>
</table>
PromQL instant vector selector

| Query target | Prometheus server | Plain text in Prometheus exposition format |
| Returns | Instant vector | Metric or label value (Prometheus pattern) |
| | | Array of metrics for single value in JSON (Prometheus to JSON) |
| Label matching operators | =, !=, =~, !~ | =, !=, =~, !~ |
| Regular expression used in label or metric name matching | RE2 | PCRE |
| Comparison operators | See list | Only == (equal) is supported for value filtering |

Similarities

Selecting by metric name that equals string

| <metric name> or | __name__="<metric name>" |
Selecting by metric name that matches regular expression

| __name__=~"<regex>" |
Selecting by <label name> value that equals string

| <label name>="<label value>" |
Selecting by <label name> value that matches regular expression

| <label name>=~"<regex>" |
Selecting by value that equals string

| __name__=~".*" == <value> |

18 Script items

Overview

Script items can be used to collect data by executing a user-defined JavaScript code with the ability to retrieve data over HTTP/HTTPS. In addition to the script, an optional list of parameters (pairs of name and value) and timeout can be specified.

This item type may be useful in data collection scenarios that require multiple steps or complex logic. As an example, a Script item can be configured to make an HTTP call, then process the data received in the first step in some way, and pass transformed value to the second HTTP call.

Script items are processed by Zabbix server or proxy pollers.

Configuration

In the Type field of item configuration form select Script then fill out required fields.

All mandatory input fields are marked with a red asterisk.

The fields that require specific information for Script items are:
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Enter a unique key that will be used to identify the item.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Specify the variables to be passed to the script as the attribute and value pairs. User macros are supported. To see which built-in macros are supported, do a search for “Script-type item” in the supported macro table.</td>
</tr>
<tr>
<td>Script</td>
<td>Enter JavaScript code in the block that appears when clicking in the parameter field (or on the view/edit button next to it). This code must provide the logic for returning the metric value.</td>
</tr>
<tr>
<td>Timeout</td>
<td>JavaScript execution timeout (1-60s, default 3s); exceeding it will return error. Time suffixes are supported, e.g. 30s, 1m. Depending on the script it might take longer for the timeout to trigger.</td>
</tr>
</tbody>
</table>

Examples

Simple data collection

Collect the content of https://www.example.com/release_notes:

- Create an item with type “Script”.
  - In the Script field, enter:
    ```javascript
    var request = new HttpRequest();
    return request.get("https://www.example.com/release_notes");
    ```

Data collection with parameters

Get the content of a specific page and make use of parameters:

- Create an item with type Script and two parameters:
  - url : {$DOMAIN} (the user macro {$DOMAIN} should be defined, preferably on a host level)
  - subpage : /release_notes
  - In the script field, enter:
    ```javascript
    var obj = JSON.parse(value); var url = obj.url; var subpage = obj.subpage; var request = new HttpRequest(); return request.get(url+subpage);
    ```

Multiple HTTP requests

Collect the content of both https://www.example.com and https://www.example.com/release_notes:

- Create an item with type “Script”.
  - In the Script field, enter:
    ```javascript
    var request = new HttpRequest();
    return request.get("https://www.example.com") + request.get("https://www.example.com/release_notes");
    ```
Logging

Add the "Log test" entry to the Zabbix server log and receive the item value "1" in return:

- Create an item with type “Script”.
- In the Script field, enter:

```python
Zabbix.log(3, 'Log test');
return 1;
```

### 4 History and trends

**Overview**

History and trends are the two ways of storing collected data in Zabbix.

Whereas history keeps each collected value, trends keep averaged information on hourly basis and therefore are less resource-hungry.

**Keeping history**

You can set for how many days history will be kept:

- in the item properties form
- when mass-updating items
- when setting up housekeeper tasks

Any older data will be removed by the housekeeper.

The general strong advice is to keep history for the smallest possible number of days and that way not to overload the database with lots of historical values.

Instead of keeping a long history, you can keep longer data of trends. For example, you could keep history for 14 days and trends for 5 years.

You can get a good idea of how much space is required by history versus trends data by referring to the database sizing page.

While keeping shorter history, you will still be able to review older data in graphs, as graphs will use trend values for displaying older data.

If history is set to ‘0’, the item will update only dependent items and inventory. No trigger functions will be evaluated because trigger evaluation is based on history data only.

As an alternative way to preserve history consider to use history export functionality of loadable modules.

**Keeping trends**

Trends is a built-in historical data reduction mechanism which stores minimum, maximum, average and the total number of values per every hour for numeric data types.

You can set for how many days trends will be kept:

- in the item properties form
- when mass-updating items
- when setting up Housekeeper tasks

Trends usually can be kept for much longer than history. Any older data will be removed by the housekeeper.

Zabbix server accumulates trend data in runtime in the trend cache, as the data flows in. Server flushes previous hour trends of every item into the database (where frontend can find them) in these situations:

- server receives the first current hour value of the item
- 5 or less minutes of the current hour left and still no current hour values of the item
- server stops

To see trends on a graph you need to wait at least to the beginning of the next hour (if item is updated frequently) and at most to the end of the next hour (if item is updated rarely), which is 2 hours maximum.

When server flushes trend cache and there are already trends in the database for this hour (for example, server has been restarted mid-hour), server needs to use update statements instead of simple inserts. Therefore on a bigger installation if restart is needed it is desirable to stop server in the end of one hour and start in the beginning of the next hour to avoid trend data overlap.

History tables do not participate in trend generation in any way.
If trends are set to ‘0’, Zabbix server does not calculate or store trends at all.

The trends are calculated and stored with the same data type as the original values. As a result the average value calculations of unsigned data type values are rounded and the less the value interval is the less precise the result will be. For example if item has values 0 and 1, the average value will be 0, not 0.5.

Also restarting server might result in the precision loss of unsigned data type average value calculations for the current hour.

5 User parameters

Overview

Sometimes you may want to run an agent check that does not come predefined with Zabbix. This is where user parameters come to help.

You may write a command that retrieves the data you need and include it in the user parameter in the agent configuration file (‘UserParameter’ configuration parameter).

A user parameter has the following syntax:

UserParameter=<key>,<command>

As you can see, a user parameter also contains a key. The key will be necessary when configuring an item. Enter a key of your choice that will be easy to reference (it must be unique within a host).

Restart the agent or use the agent runtime control option to pick up the new parameter, e. g.:

zabbix_agentd -R userparameter_reload

Then, when configuring an item, enter the key to reference the command from the user parameter you want executed.

User parameters are commands executed by Zabbix agent. Up to 512KB of data can be returned before item preprocessing steps. Note, however, that the text value that can be eventually stored in database is limited to 64KB on MySQL (see info on other databases in the table).

/bin/sh is used as a command line interpreter under UNIX operating systems. User parameters obey the agent check timeout; if timeout is reached the forked user parameter process is terminated.

See also:

• Step-by-step tutorial on making use of user parameters
• Command execution

Examples of simple user parameters

A simple command:

UserParameter=ping,echo 1

The agent will always return ‘1’ for an item with ‘ping’ key.

A more complex example:

UserParameter=mysql.ping,mysqladmin -uroot ping | grep -c alive

The agent will return ‘1’, if MySQL server is alive, ‘0’ - otherwise.

Flexible user parameters

Flexible user parameters accept parameters with the key. This way a flexible user parameter can be the basis for creating several items.

Flexible user parameters have the following syntax:

UserParameter=key[*],command

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Unique item key. The [*] defines that this key accepts parameters within the brackets. Parameters are given when configuring the item.</td>
</tr>
</tbody>
</table>
**Parameter Description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command</strong></td>
<td>Command to be executed to evaluate value of the key. For flexible user parameters only: You may use positional references $1…$9 in the command to refer to the respective parameter in the item key. Zabbix parses the parameters enclosed in [ ] of the item key and substitutes $1,...,$9 in the command accordingly. $0 will be substituted by the original command (prior to expansion of $0,...,$9) to be run. Positional references are interpreted regardless of whether they are enclosed between double ('') or single ('') quotes. To use positional references unaltered, specify a double dollar sign - for example, awk '{print $$2}'. In this case $$2 will actually turn into $2 when executing the command.</td>
</tr>
</tbody>
</table>

Positional references with the $ sign are searched for and replaced by Zabbix agent only for flexible user parameters. For simple user parameters, such reference processing is skipped and, therefore, any $ sign quoting is not necessary.

Certain symbols are not allowed in user parameters by default. See UnsafeUserParameters documentation for a full list.

**Example 1**

Something very simple:

UserParameter=ping[*],echo $1

We may define unlimited number of items for monitoring all having format ping[something].

- ping[0] - will always return '0'
- ping[aaa] - will always return 'aaa'

**Example 2**

Let's add more sense!

UserParameter=mysql.ping[*],mysqladmin -u$1 -p$2 ping | grep -c alive

This parameter can be used for monitoring availability of MySQL database. We can pass user name and password:

mysql.ping[zabbix,our_password]

**Example 3**

How many lines matching a regular expression in a file?

UserParameter=wc[*],grep -c "$2" $1

This parameter can be used to calculate number of lines in a file.

wc[/etc/passwd,root]  
wcf[/etc/services,zabbix]

**Command result**

The return value of the command is standard output together with standard error.

A text (character, log or text type of information) item will not become unsupported in case of standard error output.

User parameters that return text (character, log, text type of information) can return whitespace. In case of invalid result the item will become unsupported.

**1 Extending Zabbix agents**

This tutorial provides step-by-step instructions on how to extend the functionality of Zabbix agent with the use of a user parameter.

**Step 1**

Write a script or command line to retrieve required parameter.

For example, we may write the following command in order to get total number of queries executed by a MySQL server:

mysqladmin -uroot status | cut -f4 -d:" | cut -f1 -d"S"
When executed, the command returns total number of SQL queries.

Step 2

Add the command to zabbix_agentd.conf:

```
UserParameter=mysql.questions,mysqladmin -u root status | cut -f4 -d":" | cut -f1 -d"S"
```

`mysql.questions` is a unique identifier. It can be any valid key identifier, for example, queries.

Test this parameter by using Zabbix agent with `-t` flag (if running under root, however, note that the agent may have different permissions when launched as a daemon):

```
zabbix_agentd -t mysql.questions
```

Step 3

Reload user parameters from the configuration file by running:

```
zabbix_agentd -R userparameter_reload
```

You may also restart the agent instead of the runtime control command.

Test the parameter by using `zabbix_get` utility.

Step 4

Add new item with Key=mysql.questions to the monitored host. Type of the item must be either Zabbix Agent or Zabbix Agent (active).

Be aware that type of returned values must be set correctly on Zabbix server. Otherwise Zabbix won't accept them.

### 6 Loadable modules

1 Overview

Loadable modules offer a performance-minded option for extending Zabbix functionality.

There already are ways of extending Zabbix functionality by way of:

- **user parameters** (agent metrics)
- **external checks** (agent-less monitoring)
- **system.run[]** Zabbix agent item.

They work very well, but have one major drawback, namely fork(). Zabbix has to fork a new process every time it handles a user metric, which is not good for performance. It is not a big deal normally, however it could be a serious issue when monitoring embedded systems, having a large number of monitored parameters or heavy scripts with complex logic or long startup time.

Support of loadable modules offers ways for extending Zabbix agent, server and proxy without sacrificing performance.

A loadable module is basically a shared library used by Zabbix daemon and loaded on startup. The library should contain certain functions, so that a Zabbix process may detect that the file is indeed a module it can load and work with.

Loadable modules have a number of benefits. Great performance and ability to implement any logic are very important, but perhaps the most important advantage is the ability to develop, use and share Zabbix modules. It contributes to trouble-free maintenance and helps to deliver new functionality easier and independently of the Zabbix core code base.

Module licensing and distribution in binary form is governed by the GPL license (modules are linking with Zabbix in runtime and are using Zabbix headers; currently the whole Zabbix code is licensed under GPL license). Binary compatibility is not guaranteed by Zabbix.

Module API stability is guaranteed during one Zabbix LTS (Long Term Support) release cycle. Stability of Zabbix API is not guaranteed (technically it is possible to call Zabbix internal functions from a module, but there is no guarantee that such modules will work).

2 Module API

In order for a shared library to be treated as a Zabbix module, it should implement and export several functions. There are currently six functions in the Zabbix module API, only one of which is mandatory and the other five are optional.

2.1 Mandatory interface

The only mandatory function is `zbx_module_api_version()`:

```c
int zbx_module_api_version(void);
```
This function should return the API version implemented by this module and in order for the module to be loaded this version must match module API version supported by Zabbix. Version of module API supported by Zabbix is ZBX_MODULE_API_VERSION. So this function should return this constant. Old constant ZBX_MODULE_API_VERSION_ONE used for this purpose is now defined to equal ZBX_MODULE_API_VERSION to preserve source compatibility, but it’s usage is not recommended.

2.2 Optional interface

The optional functions are `zbx_module_init()`, `zbx_module_item_list()`, `zbx_module_item_timeout()`, `zbx_module_history_write_cbs()` and `zbx_module_uninit()`:

```c
int zbx_module_init(void);
```

This function should perform the necessary initialization for the module (if any). If successful, it should return ZBX_MODULE_OK. Otherwise, it should return ZBX_MODULE_FAIL. In the latter case Zabbix will not start.

```c
ZBX_METRIC *zbx_module_item_list(void);
```

This function should return a list of items supported by the module. Each item is defined in a ZBX_METRIC structure, see the section below for details. The list is terminated by a ZBX_METRIC structure with “key” field of NULL.

```c
void zbx_module_item_timeout(int timeout);
```

If module exports `zbx_module_item_list()` then this function is used by Zabbix to specify the timeout settings in Zabbix configuration file that the item checks implemented by the module should obey. Here, the “timeout” parameter is in seconds.

```c
ZBX_HISTORY_WRITE_CBS zbx_module_history_write_cbs(void);
```

This function should return callback functions Zabbix server will use to export history of different data types. Callback functions are provided as fields of ZBX_HISTORY_WRITE_CBS structure, fields can be NULL if module is not interested in the history of certain type.

```c
int zbx_module_uninit(void);
```

This function should perform the necessary uninitialization (if any) like freeing allocated resources, closing file descriptors, etc.

All functions are called once on Zabbix startup when the module is loaded, with the exception of `zbx_module_uninit()`, which is called once on Zabbix shutdown when the module is unloaded.

2.3 Defining items

Each item is defined in a ZBX_METRIC structure:

```c
typedef struct
{
    char     *key;
    unsigned flags;
    int       (*function)();
    char     *test_param;
} ZBX_METRIC;
```

Here, `key` is the item key (e.g., “dummy.random”), `flags` is either CF_HAVEPARAMS or 0 (depending on whether the item accepts parameters or not), `function` is a C function that implements the item (e.g., “zbx_module_dummy_random”), and `test_param` is the parameter list to be used when Zabbix agent is started with the “-p” flag (e.g., “1,1000”, can be NULL). An example definition may look like this:

```c
static ZBX_METRIC keys[] =
{
    { "dummy.random", CF_HAVEPARAMS, zbx_module_dummy_random, "1,1000" },
    { NULL }
}
```

Each function that implements an item should accept two pointer parameters, the first one of type AGENT_REQUEST and the second one of type AGENT_RESULT:

```c
int zbx_module_dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    ...
    SET_UI64_RESULT(result, from + rand() % (to - from + 1));
}
return SYSINFO_RET_OK;
}

These functions should return SYSINFO_RET_OK, if the item value was successfully obtained. Otherwise, they should return SYSINFO_RET_FAIL. See example “dummy” module below for details on how to obtain information from AGENT_REQUEST and how to set information in AGENT_RESULT.

2.4 Providing history export callbacks

History export via module is no longer supported by Zabbix proxy since Zabbix 4.0.0.

Module can specify functions to export history data by type: Numeric (float), Numeric (unsigned), Character, Text and Log:

typedef struct
{
    void (*history_float_cb)(const ZBX_HISTORY_FLOAT *history, int history_num);
    void (*history_integer_cb)(const ZBX_HISTORY_INTEGER *history, int history_num);
    void (*history_string_cb)(const ZBX_HISTORY_STRING *history, int history_num);
    void (*history_text_cb)(const ZBX_HISTORY_TEXT *history, int history_num);
    void (*history_log_cb)(const ZBX_HISTORY_LOG *history, int history_num);
}
ZBX_HISTORY_WRITE_CBS;

Each of them should take “history” array of “history_num” elements as arguments. Depending on history data type to be exported, “history” is an array of the following structures, respectively:

typedef struct
{
    zbx_uint64_t itemid;
    int clock;
    int ns;
    double value;
}
ZBX_HISTORY_FLOAT;

typedef struct
{
    zbx_uint64_t itemid;
    int clock;
    int ns;
    zbx_uint64_t value;
}
ZBX_HISTORY_INTEGER;

typedef struct
{
    zbx_uint64_t itemid;
    int clock;
    int ns;
    const char *value;
}
ZBX_HISTORY_STRING;

typedef struct
{
    zbx_uint64_t itemid;
    int clock;
    int ns;
    const char *value;
}
ZBX_HISTORY_TEXT;
Callbacks will be used by Zabbix server history syncer processes in the end of history sync procedure after data is written into Zabbix database and saved in value cache.

In case of internal error in history export module it is recommended that module is written in such a way that it does not block whole monitoring until it recovers but discards data instead and allows Zabbix server to continue running.

2.5 Building modules

Modules are currently meant to be built inside Zabbix source tree, because the module API depends on some data structures that are defined in Zabbix headers.

The most important header for loadable modules is `include/module.h`, which defines these data structures. Other necessary system headers that help `include/module.h` to work properly are `stdlib.h` and `stdint.h`.

With this information in mind, everything is ready for the module to be built. The module should include `stdlib.h`, `stdint.h` and `module.h`, and the build script should make sure that these files are in the include path. See example “dummy” module below for details.

Another useful header is `include/log.h`, which defines `zabbix_log()` function, which can be used for logging and debugging purposes.

3 Configuration parameters

Zabbix agent, server and proxy support two parameters to deal with modules:

- `LoadModulePath` - full path to the location of loadable modules
- `LoadModule` - module(s) to load at startup. The modules must be located in a directory specified by `LoadModulePath` or the path must precede the module name. If the preceding path is absolute (starts with '/') then `LoadModulePath` is ignored. It is allowed to include multiple `LoadModule` parameters.

For example, to extend Zabbix agent we could add the following parameters:

```
LoadModulePath=/usr/local/lib/zabbix/agent/
LoadModule=mariadb.so
LoadModule=apache.so
LoadModule=kernel.so
LoadModule=/usr/local/lib/zabbix/dummy.so
```

Upon agent startup it will load the mariadb.so, apache.so and kernel.so modules from the `/usr/local/lib/zabbix/agent` directory while dummy.so will be loaded from `/usr/local/lib/zabbix`. It will fail if a module is missing, in case of bad permissions or if a shared library is not a Zabbix module.

4 Frontend configuration

Loadable modules are supported by Zabbix agent, server and proxy. Therefore, item type in Zabbix frontend depends on where the module is loaded. If the module is loaded into the agent, then the item type should be “Zabbix agent” or “Zabbix agent (active)”.

If the module is loaded into server or proxy, then the item type should be “Simple check”.

History export through Zabbix modules does not need any frontend configuration. If the module is successfully loaded by server and provides `zbx_module_history_write_cbs()` function which returns at least one non-NULL callback function then history export will be enabled automatically.

5 Dummy module

Zabbix includes a sample module written in C language. The module is located under `src/modules/dummy`:

```
alex@alex:~trunk/src/modules/dummy$ ls -l
-rw-rw-r-- 1 alex alex 9019 Apr 24 17:54 dummy.c
-rw-rw-r-- 1 alex alex 67 Apr 24 17:54 Makefile
-rw-rw-r-- 1 alex alex 245 Apr 24 17:54 README
```

The module is well documented, it can be used as a template for your own modules.

After `./configure` has been run in the root of Zabbix source tree as described above, just run `make` in order to build `dummy.so`. 
/**
 * Zabbix
 * Copyright (C) 2001-2020 Zabbix SIA
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
 **/
* Function: zbx_module_item_timeout

* Purpose: set timeout value for processing of items

* Parameters: timeout - timeout in seconds, 0 - no timeout set

******************************************************************************/

void zbx_module_item_timeout(int timeout)
{
    item_timeout = timeout;
}

/****************************************************************************
* Function: zbx_module_item_list
* Purpose: returns list of item keys supported by the module
* Return value: list of item keys
******************************************************************************/

ZBX_METRIC *zbx_module_item_list(void)
{
    return keys;
}

static int dummy_ping(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    SET_UI64_RESULT(result, 1);
    return SYSINFO_RET_OK;
}

static int dummy_echo(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    char *param;
    if (1 != request→nparam)
    {
        /* set optional error message */
        SET_MSG_RESULT(result, strdup("Invalid number of parameters."));
        return SYSINFO_RET_FAIL;
    }
    param = get_rparam(request, 0);
    SET_STR_RESULT(result, strdup(param));
    return SYSINFO_RET_OK;
}

/****************************************************************************
* Function: dummy_random
* Purpose: a main entry point for processing of an item
* Parameters: request - structure that contains item key and parameters
* request→key - item key without parameters
* request→nparam - number of parameters
* request→params[N-1] - pointers to item key parameters
* request→types[N-1] - item key parameters types:
******************************************************************************/
static int dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    char *param1, *param2;
    int from, to;

    if (2 != request->nparam)
    {
        /* set optional error message */
        SET_MSG_RESULT(result, strdup("Invalid number of parameters."));
        return SYSINFO_RET_FAIL;
    }

    param1 = get_rparam(request, 0);
    param2 = get_rparam(request, 1);

    /* there is no strict validation of parameters and types for simplicity sake */
    from = atoi(param1);
    to = atoi(param2);

    if (from > to)
    {
        SET_MSG_RESULT(result, strdup("Invalid range specified."));
        return SYSINFO_RET_FAIL;
    }

    SET_UI64_RESULT(result, from + rand() % (to - from + 1));

    return SYSINFO_RET_OK;
}

/*****************************************************************************/

int zbx_module_init(void)
{

}
/* initialization for dummy.random */
srand(time(NULL));

    return ZBX_MODULE_OK;
}

/*****************************************************************************
* *
* Function: zbx_module_uninit
* *
* Purpose: the function is called on agent shutdown
* It should be used to cleanup used resources if there are any
* *
* Return value: ZBX_MODULE_OK - success
* ZBX_MODULE_FAIL - function failed
* *
*********************************************************************************/
int zbx_module_uninit(void)
{
    return ZBX_MODULE_OK;
}

/*****************************************************************************
* *
* Functions: dummy_history_float_cb
* dummy_history_integer_cb
* dummy_history_string_cb
* dummy_history_text_cb
* dummy_history_log_cb
* *
* Purpose: callback functions for storing historical data of types float, integer, string, text and log respectively in external storage
* *
* Parameters: history - array of historical data
* history_num - number of elements in history array
* *
*********************************************************************************/
static void dummy_history_float_cb(const ZBX_HISTORY_FLOAT *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
}

static void dummy_history_integer_cb(const ZBX_HISTORY_INTEGER *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
}

static void dummy_history_string_cb(const ZBX_HISTORY_STRING *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    { /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
static void dummy_history_text_cb(const ZBX_HISTORY_TEXT *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
}

static void dummy_history_log_cb(const ZBX_HISTORY_LOG *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
}

/ZBX_HISTORY_WRITE_CBS zbx_module_history_write_cbs(void)
{
    static ZBX_HISTORY_WRITE_CBS dummy_callbacks =
    {
        dummy_history_float_cb,
        dummy_history_integer_cb,
        dummy_history_string_cb,
        dummy_history_text_cb,
        dummy_history_log_cb,
    };

    return dummy_callbacks;
}

The module exports three new items:
- `dummy.ping` - always returns '1'
- `dummy.echo[param1]` - returns the first parameter as it is, for example, `dummy.echo[ABC]` will return ABC
- `dummy.random[param1, param2]` - returns a random number within the range of param1-param2, for example, `dummy.random[1,1000000]

6 Limitations

Support of loadable modules is implemented for the Unix platform only. It means that it does not work for Windows agents.

In some cases a module may need to read module-related configuration parameters from zabbix_agentd.conf. It is not supported currently. If you need your module to use some configuration parameters you should probably implement parsing of a module-specific configuration file.
7 Windows performance counters

Overview

You can effectively monitor Windows performance counters using the perf_counter[] key.

For example:

\texttt{perf\_counter["\Processor(0)\Interrupts/sec"]}

or

\texttt{perf\_counter["\Processor(0)\Interrupts/sec", 10]}

For more information on using this key or its English-only equivalent \texttt{perf\_counter\_en}, see Windows-specific item keys.

In order to get a full list of performance counters available for monitoring, you may run:

\texttt{typeperf -qx}

You may also use low-level discovery to discover multiple \texttt{object\ instances} of Windows performance counters and automate the creation of perf_counter items for multiple instance objects.

Numeric representation

Windows maintains numeric representations (indexes) for object and performance counter names. Zabbix supports these numeric representations as parameters to the \texttt{perf\_counter}, \texttt{perf\_counter\_en} item keys and in \texttt{PerfCounter}, \texttt{PerfCounterEn} configuration parameters.

However, it’s not recommended to use them unless you can guarantee your numeric indexes map to correct strings on specific hosts. If you need to create portable items that work across different hosts with various localized Windows versions, you can use the \texttt{perf\_counter\_en} key or \texttt{PerfCounterEn} configuration parameter which allow to use English names regardless of system locale.

To find out the numeric equivalents, run \texttt{regedit}, then find HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\009.

The registry entry contains information like this:

\begin{verbatim}
1
1847
2
System
4
Memory
6
% Processor Time
10
File Read Operations/sec
12
File Write Operations/sec
14
File Control Operations/sec
16
File Read Bytes/sec
18
File Write Bytes/sec
....
\end{verbatim}

Here you can find the corresponding numbers for each string part of the performance counter, like in \texttt{"\System\% Processor Time"}:

\begin{verbatim}
System → 2
% Processor Time → 6
\end{verbatim}

Then you can use these numbers to represent the path in numbers:

\begin{verbatim}
\2\6
\end{verbatim}

Performance counter parameters

You can deploy some PerfCounter parameters for the monitoring of Windows performance counters.

For example, you can add these to the Zabbix agent configuration file:
PerfCounter=UserPerfCounter1,"\Memory\Page Reads/sec",30
or
PerfCounter=UserPerfCounter2,"\4\24",30

With such parameters in place, you can then simply use UserPerfCounter1 or UserPerfCounter2 as the keys for creating the respective items.

Remember to restart Zabbix agent after making changes to the configuration file.

8 Mass update

Overview

Sometimes you may want to change some attribute for a number of items at once. Instead of opening each individual item for editing, you may use the mass update function for that.

Using mass update

To mass-update some items, do the following:

- Mark the checkboxes of the items to update in the list
- Click on Mass update below the list
- Navigate to the tab with required attributes (Item, Tags or Preprocessing)
- Mark the checkboxes of the attributes to update
- Enter new values for the attributes
The Tags option allows to:

- **Add** - add specified tags to the items (tags with the same name, but different values are not considered ‘duplicates’ and can be added to the same host).
- **Replace** - remove the specified tags and add tags with new values
- **Remove** - remove specified tags from the items

User macros, {INVENTORY.*} macros, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros are supported in tags.

When done, click on Update.

**9 Value mapping**

**Overview**

For a more "human" representation of received values, you can use value maps that contain the mapping between numeric/string values and string representations.

Value mappings can be used in both the Zabbix frontend and notifications sent by media types.

For example, an item which has value ‘0’ or ‘1’ can use value mapping to represent the values in a human-readable form:

- ‘0’ => ‘Not Available’
- ‘1’ => ‘Available’

Or, a backup related value map could be:

- ‘F’ => ‘Full’
- ‘D’ => ‘Differential’
- ‘I’ => ‘Incremental’
In another example, value ranges for voltage may be mapped:

- ‘<=209’ => ‘Low’
- ‘210-230’ => ‘OK’
- ‘>=231’ => ‘High’

Value mappings are defined on template or host level. Once defined they become available for all items of the respective template or host. There is no value map inheritance - a template item on a host still uses the value map from the template; linking a template with value maps to the host does not make the host inherit the value maps.

When configuring items you can use a value map to “humanize” the way an item value will be displayed. To do that, you refer to the name of a previously defined value map in the Value mapping field.

Value mapping can be used with items having Numeric (unsigned), Numeric (float) and Character type of information.

Value mappings can be exported/imported with the respective template or host.

Value mappings can be mass updated. Both host and template mass update forms have a Value mapping tab for mass updating value maps.

Configuration

To define a value map:

- Open a host or template configuration form
- Go to the Value mapping tab
- Click on Add to add a new map
- Click on the name of an existing map to edit it

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Unique name of a set of value mappings.</td>
</tr>
<tr>
<td>Mappings</td>
<td>Individual mapping rules for mapping numeric/string values to string representations. Mapping is applied according to the order of mapping rules. It is possible to reorder mappings by dragging. Only numeric value types are supported for mapping ranges (is greater than or equals, is less than or equals, in range mapping types).</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| **Type**  | Mapping type:  
* **equals** - equal values will be mapped  
* **is greater than or equals** - equal or greater values will be mapped  
* **is less than or equals** - equal or smaller values will be mapped  
* **in range** - values in range will be mapped; the range is expressed as `<number1>-<number2>`, or `<number>`. Multiple ranges are supported (e.g. 1-10,101-110,201)  
* **regexp** - values corresponding to the regular expression will be mapped (global regular expressions are not supported)  
* **default** - all outstanding values will be mapped, other than those with specific mappings |
| **Value** | Incoming value.  
Depending on the mapping type, may also contain a range or regular expression. |
| **Mapped to** | String representation for the incoming value. |

All mandatory input fields are marked with a red asterisk.

When the value map is displayed in the list, only the first three mappings of it are visible, while three dots indicate that more mappings exist.

How this works

For example, one of the predefined agent items ‘Zabbix agent ping’ uses a template-level value map called ‘Zabbix agent ping status’ to display its values.

In the item **configuration form** you can see a reference to this value map in the Value mapping field:

In Monitoring → Latest data the mapping is put to use to display ‘Up’ (with the raw value in parentheses).
In the Latest data section displayed values are shortened to 20 symbols. If value mapping is used, this shortening is not applied to the mapped value, but only to the raw value separately (displayed in parentheses).

A value being displayed in a human-readable form is also easier to understand when receiving notifications.

Without a predefined value map you would only get this:

<table>
<thead>
<tr>
<th>□ Host</th>
<th>Name</th>
<th>Last check</th>
<th>Last value</th>
</tr>
</thead>
<tbody>
<tr>
<td>□ Zabbix server</td>
<td>Monitoring agent (1 Item)</td>
<td>02/23/2021 06:00:07 PM</td>
<td>1</td>
</tr>
</tbody>
</table>

So in this case you would either have to guess what the ‘1’ stands for or do a search of documentation to find out.

### 10 Queue

**Overview**

The queue displays items that are waiting for a refresh. The queue is just a **logical** representation of data. There is no IPC queue or any other queue mechanism in Zabbix.

Items monitored by proxies are also included in the queue - they will be counted as queued for the proxy history data update period.

Only items with scheduled refresh times are displayed in the queue. This means that the following item types are excluded from the queue:

- log, logrt and event log active Zabbix agent items
- SNMP trap items
- trapper items
- web monitoring items
- dependent items

Statistics shown by the queue is a good indicator of the performance of Zabbix server.

The queue is retrieved directly from Zabbix server using JSON protocol. The information is available only if Zabbix server is running.

**Reading the queue**

To read the queue, go to Administration → Queue.

The picture here is generally “ok” so we may assume that the server is doing fine.

The queue shows some items waiting up to 30 seconds. It would be great to know what items these are.

To do just that, select Queue details in the title dropdown. Now you can see a list of those delayed items.
With these details provided it may be possible to find out why these items might be delayed.

With one or two delayed items there perhaps is no cause for alarm. They might get updated in a second. However, if you see a bunch of items getting delayed for too long, there might be a more serious problem.

**Queue item**

A special internal item `zabbix[queue,<from>,<to>]` can be used to monitor the health of the queue in Zabbix. It will return the number of items delayed by the set amount of time. For more information see [Internal items](#).

### 11 Value cache

**Overview**

To make the calculation of trigger expressions, calculated items and some macros much faster, a value cache option is supported by the Zabbix server.

This in-memory cache can be used for accessing historical data, instead of making direct SQL calls to the database. If historical values are not present in the cache, the missing values are requested from the database and the cache updated accordingly.

To enable the value cache functionality, an optional `ValueCacheSize` parameter is supported by the Zabbix server configuration file.

Two internal items are supported for monitoring the value cache: `zabbix[vcache,buffer,<mode>]` and `zabbix[vcache,cache,<parameter>]`. See more details with [internal items](#).

### 12 Execute now

**Overview**

Checking for a new item value in Zabbix is a cyclic process that is based on configured update intervals. While for many items the update intervals are quite short, there are others (including low-level discovery rules) for which the update intervals are quite long, so in real-life situations there may be a need to check for a new value quicker - to pick up changes in discoverable resources, for example. To accommodate such a necessity, it is possible to reschedule a passive check and retrieve a new value immediately.

This functionality is supported for **passive** checks only. The following item types are supported:
The check must be present in the configuration cache in order to get executed; for more information see CacheUpdateFrequency. Before executing the check, the configuration cache is not updated, thus very recent changes to item/discovery rule configuration will not be picked up. Therefore, it is also not possible to check for a new value for an item/rule that is being created or has been created just now; use the Test option while configuring an item for that.

Configuration

To execute a passive check immediately:

- click on Execute now for selected items in the list of latest data:

Several items can be selected and “executed now” at once.

In latest data this option is available only for hosts with read-write access. Accessing this option for hosts with read-only permissions depends on the user role option called Invoke “Execute now” on read-only hosts.

- click on Execute now in an existing item (or discovery rule) configuration form:
13 Restricting agent checks

Overview

It is possible to restrict checks on the agent side by creating an item blacklist, a whitelist, or a combination of whitelist/blacklist.

To do that use a combination of two agent configuration parameters:

- **AllowKey=<pattern>** - which checks are allowed; <pattern> is specified using a wildcard (*) expression
- **DenyKey=<pattern>** - which checks are denied; <pattern> is specified using a wildcard (*) expression

Note that:

- All system.run[*] items (remote commands, scripts) are disabled by default, even when no deny keys are specified, it should be assumed that DenyKey=system.run[*] is implicitly appended.
- Since Zabbix 5.0.2 the EnableRemoteCommands agent parameter is:
  - deprecated by Zabbix agent
  - unsupported by Zabbix agent2

Therefore, to allow remote commands, specify an AllowKey=system.run[<command>,*] for each allowed command, * stands for wait and nowait mode. It is also possible to specify AllowKey=system.run[*] parameter to allow all commands with wait and nowait modes. To disallow specific remote commands, add DenyKey parameters with system.run[] commands before the AllowKey=system.run[*] parameter.

Important rules

- A whitelist without a deny rule is only allowed for system.run[*] items. For all other items, AllowKey parameters are not allowed without a DenyKey parameter; in this case Zabbix agent **will not start** with only AllowKey parameters.
- The order matters. The specified parameters are checked one by one according to their appearance order in the configuration file:
  - As soon as an item key matches an allow/deny rule, the item is either allowed or denied; and rule checking stops. So if an item matches both an allow rule and a deny rule, the result will depend on which rule comes first.
  - The order affects also EnableRemoteCommands parameter (if used).
- Unlimited numbers of AllowKey/DenyKey parameters is supported.
- AllowKey, DenyKey rules do not affect HostnameItem, HostMetadataItem, HostInterfaceItem configuration parameters.
- Key pattern is a wildcard expression where the wildcard (*) character matches any number of any characters in certain position. It might be used in both the key name and parameters.
- If a specific item key is disallowed in the agent configuration, the item will be reported as unsupported (no hint is given as to the reason):
- Zabbix agent with --print (-p) command line option will not show keys that are not allowed by configuration;
- Zabbix agent with --test (-t) command line option will return "Unsupported item key." status for keys that are not allowed by configuration;
- Denied remote commands will not be logged in the agent log (if LogRemoteCommands=1).
Use cases

Deny specific check

- Blacklist a specific check with DenyKey parameter. Matching keys will be disallowed. All non-matching keys will be allowed, except system.run[] items.

For example:

# Deny secure data access
DenyKey=vfs.file.contents[/etc/passwd,*]

A blacklist may not be a good choice, because a new Zabbix version may have new keys that are not explicitly restricted by the existing configuration. This could cause a security flaw.

Deny specific command, allow others

- Blacklist a specific command with DenyKey parameter. Whitelist all other commands, with the AllowKey parameter.

# Disallow specific command
DenyKey=system.run[ls -l /]

# Allow other scripts
AllowKey=system.run[*]

Allow specific check, deny others

- Whitelist specific checks with AllowKey parameters, deny others with DenyKey=*

For example:

# Allow reading logs:
AllowKey=vfs.file.*[/var/log/*]

# Allow localtime checks
AllowKey=system.localtime[*]

# Deny all other keys
DenyKey=*

Pattern examples

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
<th>Matches</th>
<th>No match</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Matches all possible keys with or without parameters.</td>
<td>Any</td>
<td>None</td>
</tr>
<tr>
<td>vfs.file.contents</td>
<td>Matches vfs.file.contents without parameters.</td>
<td>vfs.file.contents</td>
<td>vfs.file.contents[/etc/passwd]</td>
</tr>
<tr>
<td>vfs.file.contents[]</td>
<td>Matches vfs.file.contents with empty parameters.</td>
<td>vfs.file.contents[]</td>
<td>vfs.file.contents</td>
</tr>
<tr>
<td>vfs.file.contents[*]</td>
<td>Matches vfs.file.contents with any parameters; will not match vfs.file.contents without square brackets.</td>
<td>vfs.file.contents[]</td>
<td>vfs.file.contents</td>
</tr>
<tr>
<td>vfs.file.contents[/etc/passwd]</td>
<td>Matches vfs.file.contents with first parameters matching /etc/passwd and all other parameters having any value (also empty).</td>
<td>vfs.file.contents[/etc/passwd]</td>
<td>vfs.file.contents[/etc/passwd,utf8]</td>
</tr>
<tr>
<td>vfs.file.contents[*passwd]</td>
<td>Matches vfs.file.contents with first parameter matching <em>passwd</em> and no other parameters.</td>
<td>vfs.file.contents[/etc/passwd,utf8]</td>
<td>vfs.file.contents[/etc/passwd,utf8]</td>
</tr>
</tbody>
</table>
| vfs.file.* | Matches any keys starting with vfs.file. without any parameters. | vfs.file.* | vfs.file.*[
| vfs.file.[] | Matches any keys starting with vfs.file. with any parameters. | vfs.file.[] | vfs.file.[]
system.run and AllowKey

A hypothetical script like ‘myscript.sh’ may be executed on a host via Zabbix agent in several ways:

1. As an item key in a passive or active check, for example:
   
   • system.run[myscript.sh]
   • system.run[myscript.sh,wait]
   • system.run[myscript.sh,nowait]

   Here the user may add “wait”, “nowait” or omit the 2nd argument to use its default value in system.run[].

2. As a global script (initiated by user in front end or API).

   A user configures this script in Administration → Scripts, sets “Execute on: Zabbix agent” and puts “myscript.sh” into the script’s “Commands” input field. When invoked from frontend or API the Zabbix server sends to agent:

   • system.run[myscript.sh,wait] - up to Zabbix 5.0.4
   • system.run[myscript.sh] - since 5.0.5

   Here the user does not control the “wait”/”nowait” parameters.

3. As a remote command from an action. The Zabbix server sends to agent:

   • system.run[myscript.sh,nowait]

   Here again the user does not control the “wait”/”nowait” parameters.

What that means is if we set AllowKey like:

```
AllowKey=system.run[myscript.sh]
```

then

```
• system.run[myscript.sh] - will be allowed
• system.run[myscript.sh,wait], system.run[myscript.sh,nowait] will not be allowed - the script will not be run if invoked as a step of action
```

To allow all described variants you may add:

```
AllowKey=system.run[myscript.sh,*]
DenyKey=system.run[*]
```

to the agent/agent2 parameters.

14 Plugins

Overview

Plugins provide an option to extend the monitoring capabilities of Zabbix. Plugins are written in the Go programming language and are supported by Zabbix agent 2 only.

Plugins provide an alternative to loadable modules (written in C), and other methods for extending Zabbix functionality, such as user parameters (agent metrics), external checks (agent-less monitoring), and `system.run[]` Zabbix agent item.

The following features are specific to agent 2 and its plugins:

- support of scheduled and flexible intervals for both passive and active checks;
- task queue management with respect to schedule and task concurrency;
- plugin-level timeouts.

Since Zabbix 6.0.0, plugins don’t have to be integrated into the agent 2 directly and can be added as separate external add-ons (loadable plugins), thus making the creation process of additional plugins for gathering new monitoring metrics easier.

This page lists Zabbix native plugins and describes plugin configuration principles from the user perspective. For instructions about writing your own plugins, please see Plugin development guidelines.

Configuring plugins

This section provides common plugin configuration principles and best practices.
All plugins are configured using Plugins.* parameter, which can either be part of the Zabbix agent 2 configuration file or a plugin’s own configuration file. If a plugin uses a separate configuration file, path to this file should be specified in the Include parameter of Zabbix agent 2 configuration file.

Each plugin parameter should have the following structure:

Plugins.<PluginName>.<Parameter>=<Value>

Parameter names should adhere to the following requirements:

- it is recommended to capitalize the names of your plugins;
- the parameter should be capitalized;
- special characters are not allowed;
- nesting isn’t limited by a maximum level;
- the number of parameters is not limited.

Named sessions

Named sessions represent an additional level of plugin parameters and can be used to define separate sets of authentication parameters for each of the instances being monitored. Each named session parameter should have the following structure:

Plugins.<PluginName>.<SessionName>.<Parameter>=<Value>

A session name can be used as a connString item key parameter instead of specifying a URI, username, and password separately. In item keys, the first parameter can be either a connString or a URI. If the first key parameter matches a session name specified in the configuration file, the check will be executed using named session parameters. If the first key parameter doesn’t match any session name, it will be treated as a URI.

Note, that:

- when providing a connString (session name) in key parameters, key parameters for username and password must be empty;
- passing embedded URI credentials is not supported, consider using named sessions instead;
- in case an authentication parameter is not specified for the named session, a hardcoded default value will be used.

The list of available named session parameters depends on the plugin, see individual plugin configuration files for details.

**Example:** Monitoring of two instances “MySQL1” and “MySQL2” can be configured in the following way:

Plugins.Mysql.Sessions.MySQL1.Password=<PasswordForMySQL1>

Now, these names may be used as connStrings in keys instead of URIs:

mysql.ping[MySQL1]
mysql.ping[MySQL2]

Hardcoded defaults

If a parameter required for authentication is not provided in an item key or in the named session parameters, the plugin will use a hardcoded default value.

Connections

Some plugins support gathering metrics from multiple instances simultaneously. Both local and remote instances can be monitored. TCP and Unix-socket connections are supported.

It is recommended to configure plugins to keep connections to instances in an open state. The benefits are reduced network congestion, latency, and CPU and memory usage due to the lower number of connections. The client library takes care of this.

Time period for which unused connections should remain open can be determined by Plugins.<PluginName>.KeepAlive parameter.

**Example:** Plugins.Memcached.KeepAlive

Plugins supplied out-of-the-box

All metrics supported by Zabbix agent 2 are collected by plugins. The following plugins for Zabbix agent 2 are available out-of-the-box:
<table>
<thead>
<tr>
<th>Plugin name</th>
<th>Description</th>
<th>Supported item keys</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>Metrics of the Zabbix agent being used.</td>
<td>agent.hostname, agent.ping, agent.version</td>
<td>Supported keys have the same parameters as Zabbix agent <strong>keys</strong>.</td>
</tr>
<tr>
<td>Ceph</td>
<td>Ceph monitoring.</td>
<td>ceph.df.details, ceph.osd.stats, ceph.osd.discovery, ceph.osd.dump, ceph.ping, ceph.pool.discovery, ceph.status</td>
<td>Supported <strong>keys</strong> can be used with Zabbix agent 2 only. See also: - Plugin documentation - Configuration parameters</td>
</tr>
<tr>
<td>CPU</td>
<td>System CPU monitoring (number of CPUs/CPU cores, discovered CPUs, utilization percentage)</td>
<td>system.cpu.discovery, system.cpu.num, system.cpu.util</td>
<td>Supported keys have the same parameters as Zabbix agent <strong>keys</strong>.</td>
</tr>
<tr>
<td>Docker</td>
<td>Monitoring of Docker containers.</td>
<td>docker.container_info, docker.container_stats, docker.containers, docker.containers.discovery, docker.data_usage, docker.images, docker.images.discovery, docker.info, docker.ping</td>
<td>Supported <strong>keys</strong> can be used with Zabbix agent 2 only. See also: <strong>Configuration parameters</strong></td>
</tr>
<tr>
<td>File</td>
<td>File metrics collection.</td>
<td>vfs.file.cksum, vfs.file.contents, vfs.file.exists, vfs.file.md5sum, vfs.file.regexp, vfs.file.regmatch, vfs.file.size, vfs.file.time</td>
<td>Supported keys have the same parameters as Zabbix agent <strong>keys</strong>.</td>
</tr>
<tr>
<td>Kernel</td>
<td>Kernel monitoring.</td>
<td>kernel.maxfiles, kernel.maxproc</td>
<td>Supported keys have the same parameters as Zabbix agent <strong>keys</strong>.</td>
</tr>
<tr>
<td>Log</td>
<td>Log file monitoring.</td>
<td>log, log.count, logrt, logrt.count</td>
<td>Supported keys have the same parameters as Zabbix agent <strong>keys</strong>. See also: Plugin configuration parameters (Unix/Windows)</td>
</tr>
<tr>
<td>Memcached</td>
<td>Memcached server monitoring.</td>
<td>memcached.ping, memcached.stats</td>
<td>Supported <strong>keys</strong> can be used with Zabbix agent 2 only. See also: - Plugin documentation - Configuration parameters</td>
</tr>
<tr>
<td>Modbus</td>
<td>Reads Modbus data.</td>
<td>modbus.get</td>
<td>Supported keys have the same parameters as Zabbix agent <strong>keys</strong>. See also: - Plugin documentation - Configuration parameters</td>
</tr>
<tr>
<td>MongoDB</td>
<td>Monitoring of MongoDB servers and clusters (document-based, distributed database).</td>
<td>mongodb.collection.stats, mongodb.collections.discovery, mongodb.collections.usage, mongodb.connpool.stats, mongodb.db.stats, mongodb.db.discovery, mongodb.jumbo_chunks.count, mongodb.oplog.stats, mongodb.ping, mongodb.rs.config, mongodb.rs.status, mongodb.server.status, mongodb.sh.discovery</td>
<td>Supported MongoDB versions: 2.6-5.3 Supported <strong>keys</strong> can be used with Zabbix agent 2 only. Plugins.MongoDB.System.Path variable needs to be set in Zabbix agent 2 configuration file with the path to the MongoDB plugin executable. See also: - Plugin documentation - Configuration parameters</td>
</tr>
<tr>
<td>Plugin name</td>
<td>Description</td>
<td>Supported item keys</td>
<td>Comments</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>---------------------</td>
<td>----------</td>
</tr>
</tbody>
</table>
| MQTT        | Receives published values of MQTT topics. | mqtt.get | Supported keys can be used with Zabbix agent 2 only. See also:  
- Plugin documentation  
- Configuration parameters |
| MySQL       | Monitoring of MySQL and its forks. | mysql.db.discovery, mysql.db.size, mysql.get_status_variables, mysql.ping, mysql.replication.discovery, mysql.replication.get_slave_status, mysql.version | To configure encrypted connection to the database, use named sessions and specify TLS parameters for the named session in the agent configuration file. Currently, TLS parameters cannot be passed as item key parameters.  
Supported keys can be used with Zabbix agent 2 only. See also:  
- Plugin documentation  
- Configuration parameters |
| NetIf       | Monitoring of network interfaces. | net.if.collisions, net.if.discovery, net.if.in, net.if.out, net.if.total | Supported keys have the same parameters as Zabbix agent keys. See also:  
- Plugin documentation  
- Configuration parameters |
| Oracle      | Oracle Database monitoring. | oracle.diskgroups.stats, oracle.diskgroups.discovery, oracle.archive.info, oracle.archive.discovery, oracle.cdb.info, oracle.custom.query, oracle.datafiles.stats, oracle.db.discovery, oracle.fra.stats, oracle.instance.info, oracle.pdb.info, oracle.pdb.discovery, oracle.pga.stats, oracle.ping, oracle.proc.stats, oracle.redolog.info, oracle.sga.stats, oracle.sessions.stats, oracle.sys.metrics, oracle.sys.params, oracle.ts.stats, oracle.ts.discovery, oracle.user.info | Install the Oracle Instant Client before using the plugin.  
Supported keys can be used with Zabbix agent 2 only. See also:  
- Plugin documentation  
- Configuration parameters |
| PostgreSQL  | Monitoring of PostgreSQL and its forks. | pgsql.autovacuum.count, pgsql.archive, pgsql.bgwriter, pgsql.cache.hit, pgsql.connections, pgsql.custom.query, pgsql.dbstat, pgsql.dbstat.sum, pgsql.db.age, pgsql.db.bloating_tables, pgsql.db.discovery, pgsql.db.size, pgsql.locks, pgsql.oldest.xid, pgsql.ping, pgsql.queries, pgsql.replication.count, pgsql.replication.process, pgsql.replication.process.discovery, pgsql.replication.recovery_role, pgsql.replication.status, pgsql.replication_lag.b, pgsql.replication_lag.sec, pgsql.uptime, pgsql.wal.stat | To configure encrypted connection to the database, use named sessions and specify TLS parameters for the named session in the agent configuration file. Currently, TLS parameters cannot be passed as item key parameters.  
Supported keys can be used with Zabbix agent 2 only. It is possible to extend functionality of the plugin with user-defined queries - see Plugin documentation for details. See also: Configuration parameters |
| Proc        | Process CPU utilization percentage. | proc.cpu.util | Supported key has the same parameters as Zabbix agent key. See also:  
- Plugin documentation  
- Configuration parameters |
| Redis       | Redis server monitoring. | redis.config, redis.info, redis.ping, redis.slowlog.count | Supported keys can be used with Zabbix agent 2 only. See also:  
- Plugin documentation  
- Configuration parameters |
<table>
<thead>
<tr>
<th>Plugin name</th>
<th>Description</th>
<th>Supported item keys</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smart</td>
<td>S.M.A.R.T. monitoring.</td>
<td>smart.attribute.discovery, smart.disk.discovery, smart.disk.get</td>
<td><strong>Sudo/root access rights to smartctl are required for the user executing Zabbix agent 2. The minimum required smartctl version is 7.1.</strong> Supported keys can be used with Zabbix agent 2 only on Linux/Windows, both as a passive and active check. See also: <strong>Configuration parameters</strong> Supported key has the same parameters as Zabbix agent <strong>key</strong>.</td>
</tr>
<tr>
<td>Swap</td>
<td>Swap space size in bytes/percentage</td>
<td>system.swap.size</td>
<td>Supported key has the same parameters as Zabbix agent <strong>key</strong>.</td>
</tr>
<tr>
<td>SystemRun</td>
<td>Runs specified command.</td>
<td>system.run</td>
<td>Supported key has the same parameters as Zabbix agent <strong>key</strong>.</td>
</tr>
<tr>
<td>Systemd</td>
<td>Monitoring of systemd services</td>
<td>systemd.unit.discovery, systemd.unit.get, systemd.unit.info</td>
<td>Supported keys can be used with Zabbix agent 2 only.</td>
</tr>
<tr>
<td>TCP</td>
<td>TCP connection availability check.</td>
<td>net.tcp.port</td>
<td>Supported key has the same parameters as Zabbix agent <strong>key</strong>.</td>
</tr>
<tr>
<td>UDP</td>
<td>Monitoring of the UDP services availability and performance.</td>
<td>net.udp.service, net.udp.service.perf</td>
<td>Supported keys have the same parameters as Zabbix agent <strong>keys</strong>.</td>
</tr>
<tr>
<td>Uname</td>
<td>Retrieval of information about the system.</td>
<td>system.hostname, system.sw.arch, system.uname</td>
<td>Supported keys have the same parameters as Zabbix agent <strong>keys</strong>.</td>
</tr>
<tr>
<td>Uptime</td>
<td>System uptime metrics collection.</td>
<td>system.uptime</td>
<td>Supported key has the same parameters as Zabbix agent <strong>key</strong>.</td>
</tr>
<tr>
<td>VFSDev</td>
<td>VFS metrics collection.</td>
<td>vfs.dev.discovery, vfs.dev.read, vfs.dev.write</td>
<td>Supported keys have the same parameters as Zabbix agent <strong>keys</strong>.</td>
</tr>
<tr>
<td>WebCertificate</td>
<td>Monitoring of TLS/SSL website certificates.</td>
<td>web.certificate.get</td>
<td>Supported <strong>key</strong> can be used with Zabbix agent 2 only.</td>
</tr>
<tr>
<td>WebPage</td>
<td>Web page monitoring.</td>
<td>web.page.get, web.page.perf, web.page.regex</td>
<td>Supported keys have the same parameters as Zabbix agent <strong>keys</strong>.</td>
</tr>
<tr>
<td>ZabbixAsync</td>
<td>Asynchronous metrics collection.</td>
<td>net.tcp.listen, net.udp.listen, sensor, system.boottime, system.cpu.intr, system.cpu.load, system.cpu.switches, system.hw.cpu, system.hw.macaddr, system.localtime, system.sw.os, system.swap.in, system.swap.out, vfs.fs.discovery</td>
<td>Supported keys have the same parameters as Zabbix agent <strong>keys</strong>.</td>
</tr>
</tbody>
</table>
### 3 Triggers

Overview

Triggers are logical expressions that "evaluate" data gathered by items and represent the current system state.

While items are used to gather system data, it is highly impractical to follow these data all the time waiting for a condition that is alarming or deserves attention. The job of "evaluating" data can be left to trigger expressions.

Trigger expressions allow to define a threshold of what state of data is "acceptable". Therefore, should the incoming data surpass the acceptable state, a trigger is "fired" - or changes status to PROBLEM.

A trigger may have the following status:

<table>
<thead>
<tr>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>This is a normal trigger state.</td>
</tr>
<tr>
<td>PROBLEM</td>
<td>Normally means that something happened. For example, the processor load is too high.</td>
</tr>
</tbody>
</table>

In a basic trigger we may want to set a threshold for a five minute average of some data, for example, the CPU load. This is accomplished by defining a trigger expression where:

- we apply the 'avg' function to the value received in the item key
- we use a five minute period for evaluation
- we set a threshold of '2'

\[
\text{avg(host/key,5m)} > 2
\]

This trigger will “fire” (become PROBLEM) if the five-minute average is over 2.

In a more complex trigger, the expression may include a combination of multiple functions and multiple thresholds. See also: [Trigger expression](#).

Most trigger functions are evaluated based on history data, while some trigger functions for long-term analytics, e.g. `trendavg()`, `trendcount()`, etc, use trend data.

Calculation time

A trigger is recalculated every time Zabbix server receives a new value that is part of the expression. When a new value is received, each function that is included in the expression is recalculated (not just the one that received the new value).

Additionally, a trigger is recalculated each time when a new value is received and every 30 seconds if time-based functions are used in the expression.

Time-based functions are `nodata()`, `date()`, `dayofmonth()`, `dayofweek()`, `time()`, `now()`; they are recalculated every 30 seconds by the Zabbix history syncer process.

Triggers that reference trend functions only are evaluated once per the smallest time period in the expression. See also trend functions.
Evaluation period

An evaluation period is used in functions referencing the item history. It allows to specify the interval we are interested in. It can be specified as a time period (30s, 10m, 1h) or as a value range (#5 - for five latest values).

The evaluation period is measured up to “now” - where “now” is the latest recalculation time of the trigger (see Calculation time above); “now” is not the “now” time of the server.

The evaluation period specifies either:

- To consider all values between “now-time period” and “now” (or, with time shift, between “now-time shift-time period” and “now-time shift”)
- To consider no more than the num count of values from the past, up to “now”
  - If there are 0 available values for the time period or num count specified - then the trigger or calculated item that uses this function becomes unsupported

Note that:

- If only a single function (referencing data history) is used in the trigger, “now” is always the latest received value. For example, if the last value was received an hour ago, the evaluation period will be regarded as up to the latest value an hour ago.
- A new trigger is calculated as soon as the first value is received (history functions); it will be calculated within 30 seconds for time-based functions. Thus the trigger will be calculated even though perhaps the set evaluation period (for example, one hour) has not yet passed since the trigger was created. The trigger will also be calculated after the first value, even though the evaluation range was set, for example, to ten latest values.

1 Configuring a trigger

Overview

To configure a trigger, do the following:

- Go to: Configuration → Hosts
- Click on Triggers in the row of the host
- Click on Create trigger to the right (or on the trigger name to edit an existing trigger)
- Enter parameters of the trigger in the form

See also general information on triggers and their calculation times.

Configuration

The Trigger tab contains all the essential trigger attributes.
All mandatory input fields are marked with a red asterisk.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$1, $2...$9 macros can be used to refer to the first, second...ninth constant of the expression. Note: $1-$9 macros will resolve correctly if referring to constants in relatively simple, straightforward expressions. For example, the name &quot;Processor load above $1 on {HOST:NAME}&quot; will automatically change to &quot;Processor load above 5 on New host&quot; if the expression is last(/New host/system.cpu.util{percpu,avg1})&gt;5</td>
</tr>
<tr>
<td><strong>Event name</strong></td>
<td>If defined, this name will be used to create the problem event name, instead of the trigger name. The event name may be used to build meaningful alerts containing problem data (see example). The same set of macros is supported as in the trigger name, plus (TIME) and (?EXPRESSION) expression macros. Supported since Zabbix 5.2.0.</td>
</tr>
</tbody>
</table>
### Parameter Description

#### Operational data
Operational data allow to define arbitrary strings along with macros. The macros will resolve dynamically to real time data in Monitoring → Problems. While macros in the trigger name (see above) will resolve to their values at the moment of a problem happening and will become the basis of a static problem name, the macros in the operational data maintain the ability to display the very latest information dynamically. The same set of macros is supported as in the trigger name.

#### Severity
Set the required trigger severity by clicking the buttons.

#### Expression
Logical expression used to define the conditions of a problem. A problem is created after all the conditions included in the expression are met, i.e. the expression evaluates to TRUE. The problem will be resolved as soon as the expression evaluates to FALSE, unless additional recovery conditions are specified in Recovery expression.

The same set of macros is supported as in the trigger name.

#### OK event generation
OK event generation options:
- **Expression** - OK events are generated based on the same expression as problem events;
- **Recovery expression** - OK events are generated if the problem expression evaluates to FALSE and the recovery expression evaluates to TRUE;
- **None** - in this case the trigger will never return to an OK state on its own.

#### Recovery expression
Logical expression (optional) defining additional conditions that have to be met before the problem is resolved, after the original problem expression has already been evaluated as FALSE. Recovery expression is useful for trigger hysteresis. It is not possible to resolve a problem by recovery expression alone if the problem expression is still TRUE.

This field is only available if ‘Recovery expression’ is selected for OK event generation.

#### PROBLEM event generation mode
Mode for generating problem events:
- **Single** - a single event is generated when a trigger goes into the ‘Problem’ state for the first time;
- **Multiple** - an event is generated upon every ‘Problem’ evaluation of the trigger.

#### OK event closes
Select if OK event closes:
- **All problems** - all problems of this trigger
- **All problems if tag values match** - only those trigger problems with matching event tag values

#### Tag for matching
Enter event tag name to use for event correlation.
This field is displayed if ‘All problems if tag values match’ is selected for the OK event closes property and is mandatory in this case.

#### Allow manual close
Check to allow manual closing of problem events generated by this trigger. Manual closing is possible when acknowledging problem events.

#### URL
If not empty, the URL entered here is available as a link in several frontend locations, e.g. when clicking on the problem name in Monitoring → Problems (URL option in the Trigger menu) and Problems dashboard widget.

The same set of macros is supported as in the trigger name, plus {EVENT.ID}, {HOST.ID} and {TRIGGER.ID}. Note that user macros with secret values will not be resolved in the URL.

#### Description
Text field used to provide more information about this trigger. May contain instructions for fixing specific problem, contact detail of responsible staff, etc.

The same set of macros is supported as in the trigger name.

#### Enabled
Unchecking this box will disable the trigger if required.

Problems of a disabled trigger are no longer displayed in the frontend, but are not deleted.

---

The **Tags** tab allows you to define trigger-level **tags**. All problems of this trigger will be tagged with the values entered here.
In addition the Inherited and trigger tags option allows to view tags defined on template level, if the trigger comes from that template. If there are multiple templates with the same tag, these tags are displayed once and template names are separated with commas. A trigger does not “inherit” and display host-level tags.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name/Value</td>
<td>Set custom tags to mark trigger events. Tags are a pair of tag name and value. You can use only the name or pair it with a value. A trigger may have several tags with the same name, but different values. User macros, user macro context, low-level discovery macros and macro functions with {{ITEM.VALUE}}, {{ITEM.LASTVALUE}} and low-level discovery macros are supported in event tags. Low-level discovery macros can be used inside macro context. {TRIGGER.ID} macro is supported in trigger tag values. It may be useful for identifying triggers created from trigger prototypes and, for example, suppressing problems from these triggers during maintenance. If the total length of expanded value exceeds 255, it will be cut to 255 characters. See all macros supported for event tags. Event tags can be used for event correlation, in action conditions and will also be seen in Monitoring → Problems or the Problems widget.</td>
</tr>
</tbody>
</table>

The Dependencies tab contains all the dependencies of the trigger.

Click on Add to add a new dependency.

You can also configure a trigger by opening an existing one, pressing the Clone button and then saving under a different name.

Testing expressions

It is possible to test the configured trigger expression as to what the expression result would be depending on the received value.

The following expression from an official template is taken as an example:

```plaintext
avg(/Cisco IOS SNMPv2/sensor.temp.value[ciscoEnvMonTemperatureValue.{#SNMPINDEX}],5m)>{$TEMP_WARN} or last(/Cisco IOS SNMPv2/sensor.temp.status[ciscoEnvMonTemperatureState.{#SNMPINDEX}])={$TEMP_WARN_STATUS}
```

To test the expression, click on Expression constructor under the expression field.

In the Expression constructor, all individual expressions are listed. To open the testing window, click on Test below the expression list.
In the testing window you can enter sample values ('80', '70', '0', '1' in this example) and then see the expression result, by clicking on the Test button.

The result of the individual expressions as well as the whole expression can be seen.

"TRUE" means that the specified expression is correct. In this particular case A, "80" is greater than the \{$TEMP_WARN\} specified value, "70" in this example. As expected, a "TRUE" result appears.

"FALSE" means that the specified expression is incorrect. In this particular case B, \{$TEMP_WARN_STATUS\} "1" needs to be equal with specified value, "0" in this example. As expected, a "FALSE" result appears.

The chosen expression type is "OR". If at least one of the specified conditions (A or B in this case) is TRUE, the overall result will be TRUE as well. Meaning that the current value exceeds the warning value and a problem has occurred.

2 Trigger expression

Overview

The expressions used in triggers are very flexible. You can use them to create complex logical tests regarding monitored statistics.

A simple expression uses a function that is applied to the item with some parameters. The function returns a result that is compared to the threshold, using an operator and a constant.

The syntax of a simple useful expression is `function(/host/key,parameter)<operator><constant>`.

For example:

```
min(/Zabbix server/net.if.in[eth0,bytes],5m)>100K
```

will trigger if the number of received bytes during the last five minutes was always over 100 kilobytes.

While the syntax is exactly the same, from the functional point of view there are two types of trigger expressions:

- problem expression - defines the conditions of the problem
- recovery expression (optional) - defines additional conditions of the problem resolution

When defining a problem expression alone, this expression will be used both as the problem threshold and the problem recovery threshold. As soon as the problem expression evaluates to TRUE, there is a problem. As soon as the problem expression evaluates to FALSE, the problem is resolved.

When defining both problem expression and the supplemental recovery expression, problem resolution becomes more complex: not only the problem expression has to be FALSE, but also the recovery expression has to be TRUE. This is useful to create hysteresis and avoid trigger flapping.
Functions

Functions allow to calculate the collected values (average, minimum, maximum, sum), find strings, reference current time and other factors.

A complete list of supported functions is available.

Typically functions return numeric values for comparison. When returning strings, comparison is possible with the = and <> operators (see example).

Function parameters

Function parameters allow to specify:

- host and item key (functions referencing the host item history only)
- function-specific parameters
- other expressions (not available for functions referencing the host item history, see other expressions for examples)

The host and item key can be specified as /host/key. The referenced item must be in a supported state (except for nodata() function, which is calculated for unsupported items as well).

While other trigger expressions as function parameters are limited to non-history functions in triggers, this limitation does not apply in calculated items.

Function-specific parameters

Function-specific parameters are placed after the item key and are separated from the item key by a comma. See the supported functions for a complete list of these parameters.

Most of numeric functions accept time as a parameter. You may use seconds or time suffixes to indicate time. Preceded by a hashtag, the parameter has a different meaning:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum(/host/key,10m)</td>
<td>Sum of values in the last 10 minutes.</td>
</tr>
<tr>
<td>sum(/host/key,#10)</td>
<td>Sum of the last ten values.</td>
</tr>
</tbody>
</table>

Parameters with a hashtag have a different meaning with the function last - they denote the Nth previous value, so given the values 3, 7, 2, 6, 5 (from the most recent to the least recent):

- last(/host/key,#2) would return '7'
- last(/host/key,#5) would return '5'

Time shift

An optional time shift is supported with time or value count as the function parameter. This parameter allows to reference data from a period of time in the past.

Time shift starts with now - specifying the current time, and is followed by +N<time unit> or -N<time unit> - to add or subtract N time units.

For example, avg(/host/key,1h:now-1d) will return the average value for an hour one day ago.

**Time shift with absolute time periods**

Absolute time periods are supported in the time shift parameter, for example, midnight to midnight for a day, Monday-Sunday for a week, first day-last day of the month for a month.

Time shift for absolute time periods starts with now - specifying the current time, and is followed by any number of time operations: /<time unit> - defines the beginning and end of the time unit, for example, midnight to midnight for a day, +N<time unit> or -N<time unit> - to add or subtract N time units.

Please note that the value of time shift can be greater or equal to 0, while the time period minimum value is 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1d:now/d</td>
<td>Yesterday</td>
</tr>
<tr>
<td>1d:now/d+1d</td>
<td>Today</td>
</tr>
<tr>
<td>2d:now/d+1d</td>
<td>Last 2 days</td>
</tr>
<tr>
<td>1w:now/w</td>
<td>Last week</td>
</tr>
<tr>
<td>1w:now/w+1w</td>
<td>This week</td>
</tr>
</tbody>
</table>

Other expressions
Function parameters may contain other expressions, as in the following syntax:

\[
\min(\min(/host/key,1h),\min(/host2/key2,1h)\times10)
\]

Note that other expressions may not be used, if the function references item history. For example, the following syntax is not allowed:

\[
\min(/host/key,#5*10)
\]

Operators

The following operators are supported for triggers (in descending priority of execution):

<table>
<thead>
<tr>
<th>Priority</th>
<th>Operator</th>
<th>Definition</th>
<th>Notes for unknown values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Unary minus</td>
<td>-Unknown (\rightarrow) Unknown</td>
</tr>
<tr>
<td>2</td>
<td>not</td>
<td>Logical NOT</td>
<td>not Unknown (\rightarrow) Unknown</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
<td>Multiplication</td>
<td>0 (\times) Unknown (\rightarrow) Unknown (yes, Unknown, not 0 - to not lose Unknown in arithmetic operations) 1.2 (\times) Unknown (\rightarrow) Unknown</td>
</tr>
<tr>
<td></td>
<td>/</td>
<td>Division</td>
<td>Unknown / 0 (\rightarrow) error 1.2 / Unknown (\rightarrow) Unknown 0.0 / Unknown (\rightarrow) Unknown</td>
</tr>
<tr>
<td>4</td>
<td>+</td>
<td>Arithmetical plus</td>
<td>1.2 (+) Unknown (\rightarrow) Unknown</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Arithmetical minus</td>
<td>1.2 (-) Unknown (\rightarrow) Unknown</td>
</tr>
<tr>
<td>5</td>
<td>&lt;</td>
<td>Less than. The operator is defined as:</td>
<td>1.2 (&lt;) Unknown (\rightarrow) Unknown</td>
</tr>
<tr>
<td></td>
<td>&lt;=</td>
<td>Less than or equal to. The operator is defined as:</td>
<td>Unknown (\leq) Unknown (\rightarrow) Unknown</td>
</tr>
<tr>
<td></td>
<td>&gt;</td>
<td>More than. The operator is defined as:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&gt;=</td>
<td>More than or equal to. The operator is defined as:</td>
<td></td>
</tr>
</tbody>
</table>

1. Force cast operand to float.
Priority Operator| Definition | Notes for unknown values
---|---|---
< > | Not equal. The operator is defined as: | No \(^1\)
\[A<>B \equiv (A<B-0.000001)\] or \[(A>B+0.000001)\]
7 | Logical AND | 0 and Unknown \(\rightarrow\) 0
1 and Unknown \(\rightarrow\) Unknown
Unknown and Unknown \(\rightarrow\) Unknown
8 | Logical OR | 1 or Unknown \(\rightarrow\) 1
0 or Unknown \(\rightarrow\) Unknown
Unknown or Unknown \(\rightarrow\) Unknown

\(^1\) String operand is still cast to numeric if:

- another operand is numeric
- operator other than \(=\) or \(< >\) is used on an operand

(If the cast fails - numeric operand is cast to a string operand and both operands get compared as strings.)

not, and and or operators are case-sensitive and must be in lowercase. They also must be surrounded by spaces or parentheses.

All operators, except unary - and not, have left-to-right associativity. Unary - and not are non-associative (meaning \(-\(\cdot\)\) and not (not 1) should be used instead of \(-\) and not 1).

Evaluation result:

- \(<,\leq,\geq,\neq,\equiv,<>\) operators shall yield ‘1’ in the trigger expression if the specified relation is true and ‘0’ if it is false. If at least one operand is Unknown the result is Unknown;
- and for known operands shall yield ‘1’ if both of its operands compare unequal to ‘0’; otherwise, it yields ‘0’; for unknown operands and yields ‘0’ only if one operand compares equal to ‘0’; otherwise, it yields ‘Unknown’;
- or for known operands shall yield ‘1’ if either of its operands compare unequal to ‘0’; otherwise, it yields ‘0’; for unknown operands or yields ‘1’ only if one operand compares unequal to ‘0’; otherwise, it yields ‘Unknown’;
- The result of the logical negation operator not for a known operand is ‘0’ if the value of its operand compares unequal to ‘0’; ‘1’ if the value of its operand compares equal to ‘0’. For unknown operand not yields ‘Unknown’.

Value caching

Values required for trigger evaluation are cached by Zabbix server. Because of this trigger evaluation causes a higher database load for some time after the server restarts. The value cache is not cleared when item history values are removed (either manually or by housekeeper), so the server will use the cached values until they are older than the time periods defined in trigger functions or server is restarted.

Examples of triggers

Example 1

The processor load is too high on Zabbix server.

last(/Zabbix server/system.cpu.load[all,avg1])>5

By using the function ‘last()’, we are referencing the most recent value. /Zabbix server/system.cpu.load[all,avg1] gives a short name of the monitored parameter. It specifies that the host is ‘Zabbix server’ and the key being monitored is ‘system.cpu.load[all,avg1]’. Finally, >5 means that the trigger is in the PROBLEM state whenever the most recent processor load measurement from Zabbix server is greater than 5.

Example 2

www.example.com is overloaded.

last(/www.example.com/system.cpu.load[all,avg1])>5 or min(/www.example.com/system.cpu.load[all,avg1],10m)>2

The expression is true when either the current processor load is more than 5 or the processor load was more than 2 during last 10 minutes.
Example 3

/etc/passwd has been changed.

(last(/www.example.com/vfs.file.cksum[/etc/passwd],#1)<>last(/www.example.com/vfs.file.cksum[/etc/passwd],#2))=1

The expression is true when the previous value of /etc/passwd checksum differs from the most recent one.

Similar expressions could be useful to monitor changes in important files, such as /etc/passwd, /etc/inetd.conf, /kernel, etc.

Example 4

Someone is downloading a large file from the Internet.

Use of function min:

```
min(/www.example.com/net.if.in[eth0,bytes],5m)>100K
```

The expression is true when number of received bytes on eth0 is more than 100 KB within last 5 minutes.

Example 5

Both nodes of clustered SMTP server are down.

Note use of two different hosts in one expression:

```
last(/smtp1.example.com/net.tcp.service[smtp])=0 and last(/smtp2.example.com/net.tcp.service[smtp])=0
```

The expression is true when both SMTP servers are down on both smtp1.example.com and smtp2.example.com.

Example 6

Zabbix agent needs to be upgraded.

Use of function find:

```
find(/example.example.com/agent.version,"like","beta8")=1
```

The expression is true if Zabbix agent has version beta8.

Example 7

Server is unreachable.

```
count(/example.example.com/icmpping,30m,"0")>5
```

The expression is true if host "example.example.com" is unreachable more than 5 times in the last 30 minutes.

Example 8

No heartbeats within last 3 minutes.

Use of function nodata:

```
nodata(/example.example.com/tick,3m)=1
```

To make use of this trigger, ‘tick’ must be defined as a Zabbix trapper item. The host should periodically send data for this item using zabbix_sender. If no data is received within 180 seconds, the trigger value becomes PROBLEM.

Note that ‘nodata’ can be used for any item type.

Example 9

CPU activity at night time.

Use of function time:

```
min(/Zabbix server/system.cpu.load[all,avg1],5m)>2 and time()>000000 and time()<060000
```

The trigger may change its status to true only at night (00:00-06:00).

Example 10

Check if client local time is in sync with Zabbix server time.

Use of function fuzzytime:

```
fuzzytime(/MySQL_DB/system.localtime,10s)=0
```

```
The trigger will change to the problem state in case when local time on server MySQL_DB and Zabbix server differs by more than 10 seconds. Note that 'system.localtime' must be configured as a **passive check**.

**Example 11**
Comparing average load today with average load of the same time yesterday (using time shift as `now-1d`).
\[
\text{avg}(/\text{server/system.cpu.load,1h})/\text{avg}(/\text{server/system.cpu.load,1h:now-1d}) > 2
\]
This expression will fire if the average load of the last hour tops the average load of the same hour yesterday more than two times.

**Example 12**
Using the value of another item to get a trigger threshold:
\[
\text{last}(/\text{Template PfSense/hrStorageFree[{#SNMPVALUE}]} < \text{last}(/\text{Template PfSense/hrStorageSize[{#SNMPVALUE}]})*0.1
\]
The trigger will fire if the free storage drops below 10 percent.

**Example 13**
Using **evaluation result** to get the number of triggers over a threshold:
\[
(\text{last}(/\text{server1/system.cpu.load[all,avg1]} > 5) + (\text{last}(/\text{server2/system.cpu.load[all,avg1]} > 5) + (\text{last}(/\text{server3/system.cpu.load[all,avg1]} > 5) >= 2
\]
The trigger will fire if at least two of the triggers in the expression are over 5.

**Example 14**
Comparing string values of two items - operands here are functions that return strings.
Problem: create an alert if Ubuntu version is different on different hosts
\[
\text{last}(/\text{NY Zabbix server/vfs.file.contents[/etc/os-release]} <> \text{last}(/\text{LA Zabbix server/vfs.file.contents[/etc/os-release]})
\]

**Example 15**
Comparing two string values - operands are:
- a function that returns a string
- a combination of macros and strings
Problem: detect changes in the DNS query
The item key is:
\[
\text{net.dns.record}[8.8.8.8,${WEBSITE_NAME},${DNS_Resource_RECORD_TYPE},2,1]
\]
with macros defined as
\[
{WEBSITE_NAME} = \text{example.com}
{DNS_Resource_RECORD_TYPE} = \text{MX}
\]
and normally returns:
\[
\text{example.com} \quad \text{MX} \quad 0 \quad \text{mail.example.com}
\]
So our trigger expression to detect if the DNS query result deviated from the expected result is:
\[
\text{last}(/\text{Zabbix server/net.dns.record[8.8.8.8,${WEBSITE_NAME},${DNS_Resource_RECORD_TYPE},2,1]} <> {WEBSITE_NAME} {DNS_Resource_RECORD_TYPE} 0 mail.{WEBSITE_NAME}"
\]
Notice the quotes around the second operand.

**Example 16**
Comparing two string values - operands are:
- a function that returns a string
- a string constant with special characters \ and "
Problem: detect if the /tmp/hello file content is equal to:
\[
\" //hello ?\"
\]
Option 1) write the string directly
\[
\text{last}(/\text{Zabbix server/vfs.file.contents[/tmp/hello]} = "\" //hello ?\"
\]
Notice how \ and " characters are escaped when the string gets compared directly.
Option 2) use a macro
{$HELLO_MACRO} = " //hello ?"

in the expression:
last(/Zabbix server/vfs.file.contents[/tmp/hello])={$HELLO_MACRO}

Example 17
Comparing long-term periods.

Problem: Load of Exchange server increased by more than 10% last month

trendavg(/Exchange/system.cpu.load,1M:now/M)>1.1*trendavg(/Exchange/system.cpu.load,1M:now/M-1M)

You may also use the Event name field in trigger configuration to build a meaningful alert message, for example to receive something like

"Load of Exchange server increased by 24% in July (0.69) comparing to June (0.56)"

the event name must be defined as:

Load of {HOST.HOST} server increased by {?100*trendavg(/system.cpu.load,1M:now/M)/trendavg(/system.cpu.load,1M:now/M-1M)}%

It is also useful to allow manual closing in trigger configuration for this kind of problem.

Hysteresis

Sometimes an interval is needed between problem and recovery states, rather than a simple threshold. For example, if we want to define a trigger that reports a problem when server room temperature goes above 20°C and we want it to stay in the problem state until the temperature drops below 15°C, a simple trigger threshold at 20°C will not be enough.

Instead, we need to define a trigger expression for the problem event first (temperature above 20°C). Then we need to define an additional recovery condition (temperature below 15°C). This is done by defining an additional Recovery expression parameter when defining a trigger.

In this case, problem recovery will take place in two steps:

• First, the problem expression (temperature above 20°C) will have to evaluate to FALSE
• Second, the recovery expression (temperature below 15°C) will have to evaluate to TRUE

The recovery expression will be evaluated only when the problem event is resolved first.

The recovery expression being TRUE alone does not resolve a problem if the problem expression is still TRUE!

Example 1
Temperature in server room is too high.

Problem expression:
last(/server/temp)>20

Recovery expression:
last(/server/temp)<=15

Example 2
Free disk space is too low.

Problem expression: it is less than 10GB for last 5 minutes
max(/server/vfs.fs.size[/,free],5m)<10G

Recovery expression: it is more than 40GB for last 10 minutes
min(/server/vfs.fs.size[/,free],10m)>40G

Expressions with unsupported items and unknown values

Versions before Zabbix 3.2 are very strict about unsupported items in a trigger expression. Any unsupported item in the expression immediately renders trigger value to Unknown.

Since Zabbix 3.2 there is a more flexible approach to unsupported items by admitting unknown values into expression evaluation:

• For the nodata() function, the values are not affected by whether an item is supported or unsupported. The function is evaluated even if it refers to an unsupported item.
• Logical expressions with OR and AND can be evaluated to known values in two cases regardless of unknown operands:
  - "1 or Unsupported_item1.some_function() or Unsupported_item2.some_function() or ..." can be evaluated to '1' (True),
“0 and Unsupported_item1.some_function() and Unsupported_item2.some_function() and ...” can be evaluated to ‘0’ (False).

Zabbix tries to evaluate logical expressions taking unsupported items as Unknown values. In the two cases mentioned above a known value will be produced; in other cases trigger value will be Unknown.

- If a function evaluation for supported item results in error, the function value is Unknown and it takes part in further expression evaluation.

Note that unknown values may “disappear” only in logical expressions as described above. In arithmetic expressions unknown values always lead to result Unknown (except division by 0).

If a trigger expression with several unsupported items evaluates to Unknown the error message in the frontend refers to the last unsupported item evaluated.

3 Trigger dependencies

Overview

Sometimes the availability of one host depends on another. A server that is behind some router will become unreachable if the router goes down. With triggers configured for both, you might get notifications about two hosts down - while only the router was the guilty party.

This is where some dependency between hosts might be useful. With dependency set notifications of the dependents could be withheld and only the notification for the root problem sent.

While Zabbix does not support dependencies between hosts directly, they may be defined with another, more flexible method - trigger dependencies. A trigger may have one or more triggers it depends on.

So in our simple example we open the server trigger configuration form and set that it depends on the respective trigger of the router. With such dependency the server trigger will not change state as long as the trigger it depends on is in ‘PROBLEM’ state - and thus no dependent actions will be taken and no notifications sent.

If both the server and the router are down and dependency is there, Zabbix will not execute actions for the dependent trigger.

Actions on dependent triggers will not be executed if the trigger they depend on:

- changes its state from ‘PROBLEM’ to ‘UNKNOWN’
- is closed manually, by correlation or with the help of time-based functions
- is resolved by a value of an item not involved in dependent trigger
- is disabled, has disabled item or disabled item host

Note that “secondary” (dependent) trigger in the above-mentioned cases will not be immediately updated. While parent trigger is in PROBLEM state, it’s dependents may report values, which we cannot trust. Thus, dependent trigger will only be re-evaluated, and change it’s state, only after parent trigger is in OK state and we have received trusty metrics.

Also:

- Trigger dependency may be added from any host trigger to any other host trigger, as long as it wouldn’t result in a circular dependency.
- Trigger dependency may be added from a template to a template. If a trigger from template A depends on a trigger from template B, template A may only be linked to a host (or another template) together with template B, but template B may be linked to a host (or another template) alone.
- Trigger dependency may be added from template trigger to a host trigger. In this case, linking such a template to a host will create a host trigger that depends on the same trigger template trigger was depending on. This allows to, for example, have a template where some triggers depend on router (host) triggers. All hosts linked to this template will depend on that specific router.
- Trigger dependency from a host trigger to a template trigger may not be added.
- Trigger dependency may be added from a trigger prototype to another trigger prototype (within the same low-level discovery rule) or a real trigger. A trigger prototype may not depend on a trigger prototype from a different LLD rule or on a trigger created from trigger prototype. Host trigger prototype cannot depend on a trigger from a template.

Configuration

To define a dependency, open the Dependencies tab in a trigger configuration form. Click on Add in the ‘Dependencies’ block and select one or more triggers that our trigger will depend on.
Click Update. Now the trigger has an indication of its dependency in the list.

Example of several dependencies

For example, a Host is behind a Router2 and the Router2 is behind a Router1.

Zabbix - Router1 - Router2 - Host

If Router1 is down, then obviously Host and Router2 are also unreachable yet we don’t want to receive three notifications about Host, Router1 and Router2 all being down.

So in this case we define two dependencies:

'Host is down' trigger depends on 'Router2 is down' trigger
'Router2 is down' trigger depends on 'Router1 is down' trigger

Before changing the status of the 'Host is down' trigger, Zabbix will check for corresponding trigger dependencies. If found, and one of those triggers is in 'Problem' state, then the trigger status will not be changed and thus actions will not be executed and notifications will not be sent.

Zabbix performs this check recursively. If Router1 or Router2 is unreachable, the Host trigger won’t be updated.

4 Trigger severity

Trigger severity defines how important a trigger is. Zabbix supports the following trigger severities:

<table>
<thead>
<tr>
<th>SEVERITY</th>
<th>DEFINITION</th>
<th>COLOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not classified</td>
<td>Unknown severity.</td>
<td>Gray</td>
</tr>
<tr>
<td>Information</td>
<td>For information purposes.</td>
<td>Light blue</td>
</tr>
<tr>
<td>Warning</td>
<td>Be warned.</td>
<td>Yellow</td>
</tr>
<tr>
<td>Average</td>
<td>Average problem.</td>
<td>Orange</td>
</tr>
<tr>
<td>High</td>
<td>Something important has happened.</td>
<td>Light red</td>
</tr>
<tr>
<td>Disaster</td>
<td>Disaster. Financial losses, etc.</td>
<td>Red</td>
</tr>
</tbody>
</table>

The severities are used for:

- visual representation of triggers. Different colors for different severities.
- audio in global alarms. Different audio for different severities.
- user media. Different media (notification channel) for different severities. For example, SMS - high severity, email - other.
- limiting actions by conditions against trigger severities

It is possible to customize trigger severity names and colors.

5 Customizing trigger severities
Trigger severity names and colors for severity related GUI elements can be configured in Administration → General → Trigger displaying options. Colors are shared among all GUI themes.

Translating customized severity names

If Zabbix frontend translations are used, custom severity names will override translated names by default.

Default trigger severity names are available for translation in all locales. If a severity name is changed, a custom name is used in all locales and additional manual translation is needed.

Custom severity name translation procedure:

- set required custom severity name, for example, ‘Important’
- edit `<frontend_dir>/locale/<required_locale>/LC_MESSAGES/frontend.po`
- add 2 lines:

```
msgid "Important"
msgstr "<translation string>"
```

and save file.

- create .mo files as described in `<frontend_dir>/locale/README`

Here `msgid` should match the new custom severity name and `msgstr` should be the translation for it in the specific language.

This procedure should be performed after each severity name change.

6 Mass update

Overview

With mass update you may change some attribute for a number of triggers at once, saving you the need to open each individual trigger for editing.

Using mass update

To mass-update some triggers, do the following:

- Mark the checkboxes of the triggers you want to update in the list
- Click on Mass update below the list
- Navigate to the tab with required attributes (Trigger, Tags or Dependencies)
- Mark the checkboxes of any attribute to update

---

Mass update

<table>
<thead>
<tr>
<th>Trigger</th>
<th>Tags</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>Not classified</td>
<td>Information</td>
</tr>
<tr>
<td>Allow manual close</td>
<td>Original</td>
<td></td>
</tr>
</tbody>
</table>

---

Mass update

<table>
<thead>
<tr>
<th>Trigger</th>
<th>Tags</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tags</td>
<td>Add</td>
<td>Replace</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>value</td>
</tr>
</tbody>
</table>

Add
The following options are available when selecting the respective button for tag update:

- **Add** - allows to add new tags for the triggers;
- **Replace** - will remove any existing tags from the trigger and replace them with the one(s) specified below;
- **Remove** - will remove specified tags from triggers.

Note, that tags with the same name, but different values are not considered ‘duplicates’ and can be added to the same trigger.

---

### Mass update

<table>
<thead>
<tr>
<th>Trigger</th>
<th>Tags</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Replace dependencies**, **Add**

Zabbix server: Lack of available memory (< 20M of 7.72 GB)

Replace dependencies - will remove any existing dependencies from the trigger and replace them with the one(s) specified.

Click on **Update** to apply the changes.

---

### 7 Predictive trigger functions

#### Overview

Sometimes there are signs of the upcoming problem. These signs can be spotted so that actions may be taken in advance to prevent or at least minimize the impact of the problem.

Zabbix has tools to predict the future behavior of the monitored system based on historic data. These tools are realized through predictive trigger functions.

#### 1 Functions

Two things one needs to know is how to define a problem state and how much time is needed to take action. Then there are two ways to set up a trigger signaling about a potential unwanted situation. First: the trigger must fire when the system after "time to act" is expected to be in a problem state. Second: the trigger must fire when the system is going to reach the problem state in less than "time to act". Corresponding trigger functions to use are **forecast** and **timeleft**. Note that underlying statistical analysis is basically identical for both functions. You may set up a trigger whichever way you prefer with similar results.

#### 2 Parameters

Both functions use almost the same set of parameters. Use the list of supported functions for reference.

##### 2.1 Time interval

First of all, you should specify the historic period Zabbix should analyze to come up with the prediction. You do it in a familiar way by means of the **time period** parameter and optional time shift like you do it with **avg**, **count**, **delta**, **max**, **min** and **sum** functions.

##### 2.2 Forecasting horizon

**forecast** only

Parameter **time** specifies how far in the future Zabbix should extrapolate dependencies it finds in historic data. No matter if you use **time_shift** or not, **time** is always counted starting from the current moment.

##### 2.3 Threshold to reach

**timeleft** only

Parameter **threshold** specifies a value the analyzed item has to reach, no difference if from above or from below. Once we have determined \( f(t) \) (see below) we should solve equation \( f(t) = \text{threshold} \) and return the root which is closer to now and to the right from now or 999999999999.9999 if there is no such root.

When item values approach the threshold and then cross it, **timeleft** assumes that intersection is already in the past and therefore switches to the next intersection with **threshold** level, if any. Best practice should be to use predictions as a complement to ordinary problem diagnostics, not as a substitution.¹

---

¹According to specification these are voltages on chip pins and generally speaking may need scaling.
2.4 Fit functions

Default fit is the linear function. But if your monitored system is more complicated you have more options to choose from.

\[
\text{fit} \quad x = f(t)
\]

- linear \( x = a + b*t \)
- polynomial \( x = a_0 + a_1*t + a_2*t^2 + \ldots + a_n*t^n \)
- exponential \( x = a*\exp(b*t) \)
- logarithmic \( x = a + b*\log(t) \)
- power \( x = a*t^b \)

2.5 Modes

**forecast** only

Every time a trigger function is evaluated it gets data from the specified history period and fits a specified function to the data. So, if the data is slightly different the fitted function will be slightly different. If we simply calculate the value of the fitted function at a specified time in the future you will know nothing about how the analyzed item is expected to behave between now and that moment in the future. For some fit options (like polynomial) a simple value from the future may be misleading.

<table>
<thead>
<tr>
<th>mode</th>
<th>forecast</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>f(now + time)</td>
<td>max (max_{now &lt;= t &lt;= now + time} f(t))</td>
</tr>
<tr>
<td>max</td>
<td>max</td>
<td>(\max_{now &lt;= t &lt;= now + time} f(t))</td>
</tr>
<tr>
<td>min</td>
<td>min</td>
<td>(\min_{now &lt;= t &lt;= now + time} f(t))</td>
</tr>
<tr>
<td>delta</td>
<td>max - min</td>
<td>(\text{max} - \text{min})</td>
</tr>
<tr>
<td>avg</td>
<td>avg</td>
<td>average of (f(t)) (now &lt;= t &lt;= now + time) according to definition</td>
</tr>
</tbody>
</table>

3 Details

To avoid calculations with huge numbers we consider the timestamp of the first value in specified period plus 1 ns as a new zero-time (current epoch time is of order \(10^{12}\), epoch squared is \(10^{24}\), double precision is about \(10^{-16}\)). 1 ns is added to provide all positive time values for logarithmic and power fits which involve calculating \(\log(t)\). Time shift does not affect linear, polynomial, exponential (apart from easier and more precise calculations) but changes the shape of logarithmic and power functions.

4 Potential errors

Functions return -1 in such situations:

- specified evaluation period contains no data;
- result of mathematical operation is not defined\(^3\);
- numerical complications (unfortunately, for some sets of input data range and precision of double-precision floating-point format become insufficient)\(^4\).

No warnings or errors are flagged if chosen fit poorly describes provided data or there is just too few data for accurate prediction.

5 Examples and dealing with errors

To get a warning when you are about to run out of free disk space on your host you may use a trigger expression like this:

\[
\text{timeleft(/host/vfs.fs.size[/,free],1h,0)}<1h
\]

However, error code -1 may come into play and put your trigger in a problem state. Generally it’s good because you get a warning that your predictions don’t work correctly and you should look at them more thoroughly to find out why. But sometimes it’s bad because -1 can simply mean that there was no data about the host free disk space obtained in the last hour. If you are getting too many false positive alerts consider using more complicated trigger expression\(^5\):

\[
\text{timeleft(/host/vfs.fs.size[/,free],1h,0)}<1h \text{ and } \text{timeleft(/host/vfs.fs.size[/,free],1h,0)}<>-1
\]

\(^2\)Secure indicates that the cookie should only be transmitted over a secure HTTPS connection from the client. When set to ‘true’, the cookie will only be set if a secure connection exists.

\(^3\)For example fitting exponential or power functions involves calculating \(\log()\) of item values. If data contains zeros or negative numbers you will get an error since \(\log()\) is defined for positive values only.

\(^4\)For linear, exponential, logarithmic and power fits all necessary calculations can be written explicitly. For polynomial only value can be calculated without any additional steps. Calculating \(\text{avg}\) involves computing polynomial antiderivative (analytically). Computing max and min and delta involves computing polynomial derivative (analytically) and finding its roots (numerically). Solving \(f(t)=0\) involves finding polynomial roots (numerically).

\(^5\)But in this case -1 can cause your trigger to recover from the problem state. To be fully protected use: \(\text{timeleft(/host/vfs.fs.size[/,free],1h,0)}<1h\) and \((\{\text{TRIGGER.VALUE}\}=0 \text{ and } \text{timeleft(/host/vfs.fs.size[/,free],1h,0)}<>-1\) or \(\{\text{TRIGGER.VALUE}\}=1\))
The situation is a bit more difficult with **forecast**. First of all, -1 may or may not put the trigger in a problem state depending on whether you have expression like `forecast(/host/item,(...)<=...` or like `forecast(/host/item,(...)>=...`.

Furthermore, -1 may be a valid forecast if it's normal for the item value to be negative. But probability of this situation in the real world situation is negligible (see how the operator = works). So add ... or: `forecast(/host/item,(...)=-1` and `forecast(/host/item,(...)<>-1` if you want or don't want to treat -1 as a problem respectively.

### 4 Events

**Overview**

There are several types of events generated in Zabbix:

- trigger events - whenever a trigger changes its status (OK→PROBLEM→OK)
- service events - whenever a service changes its status (OK→PROBLEM→OK)
- discovery events - when hosts or services are detected
- autoregistration events - when active agents are auto-registered by server
- internal events - when an item/low-level discovery rule becomes unsupported or a trigger goes into an unknown state

Internal events are supported starting with Zabbix 2.2 version.

Events are time-stamped and can be the basis of actions such as sending notification e-mail etc.

To view details of events in the frontend, go to Monitoring → Problems. There you can click on the event date and time to view details of an event.

More information is available on:

- trigger events
- other event sources

### 1 Trigger event generation

**Overview**

Change of trigger status is the most frequent and most important source of events. Each time the trigger changes its state, an event is generated. The event contains details of the trigger state’s change - when it happened and what the new state is.

Two types of events are created by triggers - Problem and OK.

**Problem events**

A problem event is created:

- when a trigger expression evaluates to TRUE if the trigger is in OK state;
- each time a trigger expression evaluates to TRUE if multiple problem event generation is enabled for the trigger.

**OK events**

An OK event closes the related problem event(s) and may be created by 3 components:

- triggers - based on ‘OK event generation’ and ‘OK event closes’ settings;
- event correlation
- task manager - when an event is manually closed

**Triggers**

Triggers have an ‘OK event generation’ setting that controls how OK events are generated:

- Expression - an OK event is generated for a trigger in problem state when its expression evaluates to FALSE. This is the simplest setting, enabled by default.
- Recovery expression - an OK event is generated for a trigger in problem state when its expression evaluates to FALSE and the recovery expression evaluates to TRUE. This can be used if trigger recovery criteria is different from problem criteria.
- None - an OK event is never generated. This can be used in conjunction with multiple problem event generation to simply send a notification when something happens.

Additionally triggers have an ‘OK event closes’ setting that controls which problem events are closed:

- All problems - an OK event will close all open problems created by the trigger
• All problems if tag values match - an OK event will close open problems created by the trigger and having at least one matching tag value. The tag is defined by 'Tag for matching' trigger setting. If there are no problem events to close then OK event is not generated. This is often called trigger level event correlation.

Event correlation

Event correlation (also called global event correlation) is a way to set up custom event closing (resulting in OK event generation) rules.

The rules define how the new problem events are paired with existing problem events and allow to close the new event or the matched events by generating corresponding OK events.

However, event correlation must be configured very carefully, as it can negatively affect event processing performance or, if misconfigured, close more events than intended (in the worst case even all problem events could be closed). A few configuration tips:

1. always reduce the correlation scope by setting a unique tag for the control event (the event that is paired with old events) and use the 'new event tag' correlation condition
2. don’t forget to add a condition based on the old event when using 'close old event' operation, or all existing problems could be closed
3. avoid using common tag names used by different correlation configurations

Task manager

If the 'Allow manual close' setting is enabled for trigger, then it’s possible to manually close problem events generated by the trigger. This is done in the frontend when updating a problem. The event is not closed directly – instead a 'close event' task is created, which is handled by the task manager shortly. The task manager will generate a corresponding OK event and the problem event will be closed.

2 Other event sources

Service events

Service events are generated only if service actions for these events are enabled. In this case, each service status change creates a new event:

• Problem event - when service status is changed from OK to PROBLEM
• OK event - when service status is changed from PROBLEM to OK

The event contains details of the service state change - when it happened and what the new state is.

Discovery events

Zabbix periodically scans the IP ranges defined in network discovery rules. Frequency of the check is configurable for each rule individually. Once a host or a service is discovered, a discovery event (or several events) are generated.

Zabbix generates the following events:

<table>
<thead>
<tr>
<th>Event</th>
<th>When generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Up</td>
<td>Every time Zabbix detects active service.</td>
</tr>
<tr>
<td>Service Down</td>
<td>Every time Zabbix cannot detect service.</td>
</tr>
<tr>
<td>Host Up</td>
<td>If at least one of the services is UP for the IP.</td>
</tr>
<tr>
<td>Host Down</td>
<td>If all services are not responding.</td>
</tr>
<tr>
<td>Service Discovered</td>
<td>If the service is back after downtime or discovered for the first time.</td>
</tr>
<tr>
<td>Service Lost</td>
<td>If the service is lost after being up.</td>
</tr>
<tr>
<td>Host Discovered</td>
<td>If host is back after downtime or discovered for the first time.</td>
</tr>
<tr>
<td>Host Lost</td>
<td>If host is lost after being up.</td>
</tr>
</tbody>
</table>

Active agent auto-discovery events

Active agent autoregistration creates events in Zabbix.

If configured, active agent autoregistration event is created when a previously unknown active agent asks for checks or if the host metadata has changed. The server adds a new auto-registered host, using the received IP address and port of the agent.

For more information, see the active agent autoregistration page.

Internal events

Internal events happen when:
• an item changes state from 'normal' to 'unsupported'
• an item changes state from 'unsupported' to 'normal'
• a low-level discovery rule changes state from 'normal' to 'unsupported'
• a low-level discovery rule changes state from 'unsupported' to 'normal'
• a trigger changes state from 'normal' to 'unknown'
• a trigger changes state from 'unknown' to 'normal'

Internal events are supported since Zabbix 2.2. The aim of introducing internal events is to allow users to be notified when any internal event takes place, for example, an item becomes unsupported and stops gathering data.

Internal events are only created when internal actions for these events are enabled. To stop generation of internal events (for example, for items becoming unsupported), disable all actions for internal events in Configuration → Actions → Internal actions.

If internal actions are disabled, while an object is in the 'unsupported' state, recovery event for this object will still be created.

If internal actions are enabled, while an object is in the 'unsupported' state, recovery event for this object will be created, even though ‘problem event’ has not been created for the object.

3 Manual closing of problems

Overview

While generally problem events are resolved automatically when trigger status goes from ‘Problem’ to ‘OK’, there may be cases when it is difficult to determine if a problem has been resolved by means of a trigger expression. In such cases, the problem needs to be resolved manually.

For example, syslog may report that some kernel parameters need to be tuned for optimal performance. In this case the issue is reported to Linux administrators, they fix it and then close the problem manually.

Problems can be closed manually only for triggers with the Allow manual close option enabled.

When a problem is “manually closed”, Zabbix generates a new internal task for Zabbix server. Then the task manager process executes this task and generates an OK event, therefore closing problem event.

A manually closed problem does not mean that the underlying trigger will never go into a ‘Problem’ state again. The trigger expression is re-evaluated and may result in a problem:

• When new data arrive for any item included in the trigger expression (note that the values discarded by a throttling preprocessing step are not considered as received and will not cause trigger expression to be re-evaluated);
• When time-based functions are used in the expression. Complete time-based function list can be found on Triggers page.

Configuration

Two steps are required to close a problem manually.

Trigger configuration

In trigger configuration, enable the Allow manual close option.

Problem update window

If a problem arises for a trigger with the Manual close flag, you can open the problem update popup window of that problem and close the problem manually.

To close the problem, check the Close problem option in the form and click on Update.
All mandatory input fields are marked with a red asterisk.

The request is processed by Zabbix server. Normally it will take a few seconds to close the problem. During that process CLOSING is displayed in Monitoring → Problems as the status of the problem.

Verification

It can be verified that a problem has been closed manually:

- in event details, available through Monitoring → Problems;
- by using the {EVENT.UPDATE.HISTORY} macro in notification messages that will provide this information.

5 Event correlation

Overview

Event correlation allows to correlate problem events to their resolution in a manner that is very precise and flexible.

Event correlation can be defined:

- on trigger level - one trigger may be used to relate separate problems to their solution
- globally - problems can be correlated to their solution from a different trigger/polling method using global correlation rules

1 Trigger-based event correlation

Overview

Trigger-based event correlation allows to correlate separate problems reported by one trigger.

While generally an OK event can close all problem events created by one trigger, there are cases when a more detailed approach is needed. For example, when monitoring log files you may want to discover certain problems in a log file and close them individually rather than all together.

This is the case with triggers that have Multiple Problem Event Generation enabled. Such triggers are normally used for log monitoring, trap processing, etc.
It is possible in Zabbix to relate problem events based on **tagging**. Tags are used to extract values and create identification for problem events. Taking advantage of that, problems can also be closed individually based on matching tag.

In other words, the same trigger can create separate events identified by the event tag. Therefore problem events can be identified one-by-one and closed separately based on the identification by the event tag.

**How it works**

In log monitoring you may encounter lines similar to these:

Line1: Application 1 stopped  
Line2: Application 2 stopped  
Line3: Application 1 was restarted  
Line4: Application 2 was restarted  

The idea of event correlation is to be able to match the problem event from Line1 to the resolution from Line3 and the problem event from Line2 to the resolution from Line4, and close these problems one by one:

Line1: Application 1 stopped  
Line3: Application 1 was restarted #problem from Line 1 closed  
Line2: Application 2 stopped  
Line4: Application 2 was restarted #problem from Line 2 closed  

To do this you need to tag these related events as, for example, “Application 1” and “Application 2”. That can be done by applying a regular expression to the log line to extract the tag value. Then, when events are created, they are tagged “Application 1” and “Application 2” respectively and problem can be matched to the resolution.

**Configuration**

**Item**

To begin with, you may want to set up an item that monitors a log file, for example:

```
log[/var/log/syslog]
```

With the item set up, wait a minute for the configuration changes to be picked up and then go to Latest data to make sure that the item has started collecting data.

**Trigger**

With the item working you need to configure the **trigger**. It’s important to decide what entries in the log file are worth paying attention to. For example, the following trigger expression will search for a string like “Stopping” to signal potential problems:

```
find(/My host/log[/var/log/syslog],"regexp","Stopping")=1
```

To make sure that each line containing the string “Stopping” is considered a problem also set the Problem event generation mode in trigger configuration to ‘Multiple’.

Then define a recovery expression. The following recovery expression will resolve all problems if a log line is found containing the string “Starting”:

```
find(/My host/log[/var/log/syslog],"regexp","Starting")=1
```
Since we do not want that it’s important to make sure somehow that the corresponding root problems are closed, not just all problems. That’s where tagging can help.

Problems and resolutions can be matched by specifying a tag in the trigger configuration. The following settings have to be made:

- Problem event generation mode: Multiple
- OK event closes: All problems if tag values match
- Enter the name of the tag for event matching

If configured successfully you will be able to see problem events tagged by application and matched to their resolution in Monitoring → Problems.
Because misconfiguration is possible, when similar event tags may be created for unrelated problems, please review the cases outlined below:

- With two applications writing error and recovery messages to the same log file a user may decide to use two Application tags in the same trigger with different tag values by using separate regular expressions in the tag values to extract the names of, say, application A and application B from the \{ITEM.VALUE\} macro (e.g. when the message formats differ). However, this may not work as planned if there is no match to the regular expressions. Non-matching regexps will yield empty tag values and a single empty tag value in both problem and OK events is enough to correlate them. So a recovery message from application A may accidentally close an error message from application B.

- Actual tags and tag values only become visible when a trigger fires. If the regular expression used is invalid, it is silently replaced with an *UNKNOWN* string. If the initial problem event with an *UNKNOWN* tag value is missed, there may appear subsequent OK events with the same *UNKNOWN* tag value that may close problem events which they shouldn't have closed.

- If a user uses the \{ITEM.VALUE\} macro without macro functions as the tag value, the 255-character limitation applies. When log messages are long and the first 255 characters are non-specific, this may also result in similar event tags for unrelated problems.

2 Global event correlation

Overview

Global event correlation allows to reach out over all metrics monitored by Zabbix and create correlations.

It is possible to correlate events created by completely different triggers and apply the same operations to them all. By creating intelligent correlation rules it is actually possible to save yourself from thousands of repetitive notifications and focus on root causes of a problem!

Global event correlation is a powerful mechanism, which allows you to untie yourself from one-trigger based problem and resolution logic. So far, a single problem event was created by one trigger and we were dependent on that same trigger for the problem resolution. We could not resolve a problem created by one trigger with another trigger. But with event correlation based on event tagging, we can.

For example, a log trigger may report application problems, while a polling trigger may report the application to be up and running. Taking advantage of event tags you can tag the log trigger as Status: Down while tag the polling trigger as Status: Up. Then, in a global correlation rule you can relate these triggers and assign an appropriate operation to this correlation such as closing the old events.

In another use, global correlation can identify similar triggers and apply the same operation to them. What if we could get only one problem report per network port problem? No need to report them all. That is also possible with global event correlation.

Global event correlation is configured in correlation rules. A correlation rule defines how the new problem events are paired with existing problem events and what to do in case of a match (close the new event, close matched old events by generating corresponding OK events). If a problem is closed by global correlation, it is reported in the Info column of Monitoring → Problems.

Configuring global correlation rules is available to Super Admin level users only.

Event correlation must be configured very carefully, as it can negatively affect event processing performance or, if misconfigured, close more events than was intended (in the worst case even all problem events could be closed).

To configure global correlation safely, observe the following important tips:

- Reduce the correlation scope. Always set a unique tag for the new event that is paired with old events and use the New event tag correlation condition;
- Add a condition based on the old event when using the Close old event operation (or else all existing problems could be closed);
- Avoid using common tag names that may end up being used by different correlation configurations;
- Keep the number of correlation rules limited to the ones you really need.

See also: known issues.
• Go to Configuration → Event correlation
• Click on Create correlation to the right (or on the correlation name to edit an existing rule)
• Enter parameters of the correlation rule in the form

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Unique correlation rule name.</td>
</tr>
<tr>
<td>Type of calculation</td>
<td>The following options of calculating conditions are available: And - all conditions must be met, Or - enough if one condition is met, And/Or - AND with different condition types and OR with the same condition type, Custom expression - a user-defined calculation formula for evaluating action conditions. It must include all conditions (represented as uppercase letters A, B, C, ...) and may include spaces, tabs, brackets ( ), and (case sensitive), or (case sensitive), not (case sensitive).</td>
</tr>
<tr>
<td>Conditions</td>
<td>List of conditions. See below for details on configuring a condition.</td>
</tr>
<tr>
<td>Description</td>
<td>Correlation rule description.</td>
</tr>
<tr>
<td>Operations</td>
<td>Mark the checkbox of the operation to perform when event is correlated. The following operations are available: Close old events - close old events when a new event happens. Always add a condition based on the old event when using the Close old events operation or all existing problems could be closed. Close new event - close the new event when it happens</td>
</tr>
<tr>
<td>Enabled</td>
<td>If you mark this checkbox, the correlation rule will be enabled.</td>
</tr>
</tbody>
</table>

All mandatory input fields are marked with a red asterisk.

To configure details of a new condition, click on Add in the Conditions block. A popup window will open where you can edit the condition details.
**New condition**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| New condition | Select a condition for correlating events. Note that if no old event condition is specified, all old events may be matched and closed. Similarly if no new event condition is specified, all new events may be matched and closed. The following conditions are available:  
**Old event tag** - specify the old event tag for matching.  
**New event tag** - specify the new event tag for matching.  
**New event host group** - specify the new event host group for matching.  
**Event tag pair** - specify new event tag and old event tag for matching. In this case there will be a match if the values of the tags in both events match. Tag names need not match. This option is useful for matching runtime values, which may not be known at the time of configuration (see also Example 1).  
**Old event tag value** - specify the old event tag name and value for matching, using the following operators: equals - has the old event tag value  
does not equal - does not have the old event tag value  
contains - has the string in the old event tag value  
does not contain - does not have the string in the old event tag value  
**New event tag value** - specify the new event tag name and value for matching, using the following operators: equals - has the new event tag value  
does not equal - does not have the new event tag value  
contains - has the string in the new event tag value  
does not contain - does not have the string in the new event tag value |

Because misconfiguration is possible, when similar event tags may be created for unrelated problems, please review the cases outlined below:

- Actual tags and tag values only become visible when a trigger fires. If the regular expression used is invalid, it is silently replaced with an *UNKNOWN* string. If the initial problem event with an *UNKNOWN* tag value is missed, there may appear subsequent OK events with the same *UNKNOWN* tag value that may close problem events which they shouldn’t have closed.

- If a user uses the (ITEM.VALUE) macro without macro functions as the tag value, the 255-character limitation applies. When log messages are long and the first 255 characters are non-specific, this may also result in similar event tags for unrelated problems.

**Examples**

Example 1

Stop repetitive problem events from the same network port.
This global correlation rule will correlate problems if Host and Port tag values exist on the trigger and they are the same in the original event and the new one.

The operation will close new problem events on the same network port, keeping only the original problem open.

6 Tagging

Overview

There is an option to tag various entities in Zabbix. Tags can be defined for:

- templates
- hosts
- items
- web scenarios
- triggers
- services
- template items and triggers
- host, item and trigger prototypes

Tags have several uses, most notably, to mark events. If entities are tagged, the corresponding new events get marked accordingly:

- with tagged templates - any host problems created by relevant entities (items, triggers, etc) from this template will be marked
- with tagged hosts - any problem of the host will be marked
- with tagged items, web scenarios - any data/problem of this item or web scenario will be marked
- with tagged triggers - any problem of this trigger will be marked
A problem event inherits all tags from the whole chain of templates, hosts, items, web scenarios, triggers. Completely identical tag:value combinations (after resolved macros) are merged into one rather than being duplicated, when marking the event.

Having custom event tags allows for more flexibility. Importantly, events can be correlated based on event tags. In other uses, actions can be defined based on tagged events. Item problems can be grouped based on tags. Problem tags can also be used to map problems to services.

Tagging is realized as a pair of tag name and value. You can use only the name or pair it with a value:
MySQL, Service:MySQL, Services, Services:Customer, Applications, Application:Java, Priority:High

An entity (template, host, item, web scenario, trigger or event) may be tagged with the same name, but different values - these tags will not be considered ‘duplicates’. Similarly, a tag without value and the same tag with value can be used simultaneously.

**Use cases**

Some use cases for this functionality are as follows:

1. **Mark trigger events in the frontend**
   - Define tags on trigger level;
   - See how all trigger problems are marked with these tags in Monitoring → Problems.

2. **Mark all template-inherited problems**
   - Define a tag on template level, for example 'App=MySQL';
   - See how those host problems that are created by triggers from this template are marked with these tags in Monitoring → Problems.

3. **Mark all host problems**
   - Define a tag on host level, for example 'Service=JIRA';
   - See how all problems of the host triggers are marked with these tags in Monitoring → Problems.

4. **Group related items**
   - Define a tag on item level, for example 'MySQL';
   - See all items tagged as ‘MySQL’ in Latest data by using the tag filter.

5. **Identify problems in a log file and close them separately**
   - Define tags in the log trigger that will identify events using value extraction by the \{ITEM.VALUE<N>.regsub()\} macro;
   - In trigger configuration, have multiple problem event generation mode;
   - In trigger configuration, use event correlation: select the option that OK event closes only matching events and choose the tag for matching;
   - See problem events created with a tag and closed individually.

6. **Use it to filter notifications**
   - Define tags on the trigger level to mark events by different tags;
   - Use tag filtering in action conditions to receive notifications only on the events that match tag data.

7. **Use information extracted from item value as tag value**
   - Use an \{ITEM.VALUE<N>.regsub()\} macro in the tag value;
   - See tag values in Monitoring → Problems as extracted data from item value.

8. **Identify problems better in notifications**
   - Define tags on the trigger level;
   - Use an \{EVENT.TAGS\} macro in the problem notification;
   - Easier identify which application/service the notification belongs to.

9. **Simplify configuration tasks by using tags on the template level**
   - Define tags on the template trigger level;
   - See these tags on all triggers created from template triggers.

10. **Create triggers with tags from low-level discovery (LLD)**
    - Define tags on trigger prototypes;
    - Use LLD macros in the tag name or value;
    - See these tags on all triggers created from trigger prototypes.

**Configuration**

Tags can be entered in a dedicated tab, for example, in trigger configuration:
The following macros may be used in trigger tags:

- \{ITEM.VALUE\}, \{ITEM.LASTVALUE\}, \{HOST.HOST\}, \{HOST.NAME\}, \{HOST.CONN\}, \{HOST.DNS\}, \{HOST.IP\}, \{HOST.PORT\} and \{HOST.ID\} macros can be used to populate the tag name or tag value
- \{INVENTORY.*\} macros can be used to reference host inventory values from one or several hosts in a trigger expression
- User macros and user macro context is supported for the tag name/value. User macro context may include low-level discovery macros
- Low-level discovery macros can be used for the tag name/value in trigger prototypes

The following macros may be used in trigger-based notifications:

- \{EVENT.TAGS\} and \{EVENT.RECOVERY.TAGS\} macros will resolve to a comma separated list of event tags or recovery event tags
- \{EVENT.TAGS\JSON\} and \{EVENT.RECOVERY.TAGS\JSON\} macros will resolve to a JSON array containing event tag objects or recovery event tag objects

The following macros may be used in template, host, item and web scenario tags:

- \{HOST.HOST\}, \{HOST.NAME\}, \{HOST.CONN\}, \{HOST.DNS\}, \{HOST.IP\}, \{HOST.PORT\} and \{HOST.ID\} macros
- \{INVENTORY.*\} macros
- User macros
- Low-level discovery macros can be used in item prototype tags

The following macros may be used in host prototype tags:

- \{HOST.HOST\}, \{HOST.NAME\}, \{HOST.CONN\}, \{HOST.DNS\}, \{HOST.IP\}, \{HOST.PORT\} and \{HOST.ID\} macros
- \{INVENTORY.*\} macros
- User macros
- Low-level discovery macros will be resolved during discovery process and then added to the discovered host

Substring extraction in trigger tags

Substring extraction is supported for populating the tag name or tag value, using a macro function - applying a regular expression to the value obtained by the \{ITEM.VALUE\}, \{ITEM.LASTVALUE\} macro or a low-level discovery macro. For example:

```
{{ITEM.VALUE}.regsub(pattern, output)}
{{ITEM.VALUE}.iregsub(pattern, output)}

{{#LLDMACRO}.regsub(pattern, output)}
{{#LLDMACRO}.iregsub(pattern, output)}
```

Tag name and value will be cut to 255 characters if their length exceeds 255 characters after macro resolution.

See also: Using macro functions in low-level discovery macros for event tagging.

Viewing event tags

Tagging, if defined, can be seen with new events in:
7 Visualization

1 Graphs

Overview

With lots of data flowing into Zabbix, it becomes much easier for the users if they can look at a visual representation of what is going on rather than only numbers.

This is where graphs come in. Graphs allow to grasp the data flow at a glance, correlate problems, discover when something started or make a presentation of when something might turn into a problem.

Zabbix provides users with:

- built-in simple graphs of one item data
- the possibility to create more complex customized graphs
- access to a comparison of several items quickly in ad-hoc graphs
- modern customizable vector graphs

1 Simple graphs

Overview

Simple graphs are provided for the visualization of data gathered by items.

No configuration effort is required on the user part to view simple graphs. They are freely made available by Zabbix.

Just go to Monitoring → Latest data and click on the Graph link for the respective item and a graph will be displayed.
Simple graphs are provided for all numeric items. For textual items, a link to History is available in Monitoring → Latest data.

Time period selector

Take note of the time period selector above the graph. It allows to select often required periods with one mouse click.

Note that such options as Today, This week, This month, This year display the whole period, including the hours/days in the future. Today so far, in contrast, only displays the hours passed.

Once a period is selected, it can be moved back and forth in time by clicking on the arrow buttons. The Zoom out button allows to zoom out the period two times or by 50% in each direction. Zoom out is also possible by double-clicking in the graphs. The whole time period selector can be collapsed by clicking on the tab label containing the selected period string.

The From/To fields display the selected period in either:

- absolute time syntax in format Y-m-d H:i:s
- relative time syntax, e.g.: now-1d

A date in relative format can contain one or several mathematical operations (- or +), e.g. now-1d or now-1d-2h+5m. For relative time the following abbreviations are supported:

- now
- s (seconds)
- m (minutes)
- h (hours)
- d (days)
- w (weeks)
- M (months)
- y (years)

Precision is supported in the time filter (e. g., an expression like now-1d/M). Details of precision:

<table>
<thead>
<tr>
<th>Precision</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>Y-m-d H:m:00</td>
<td>Y-m-d H:m:59</td>
</tr>
<tr>
<td>h</td>
<td>Y-m-d H:00:00</td>
<td>Y-m-d H:59:59</td>
</tr>
<tr>
<td>d</td>
<td>Y-m-d 00:00:00</td>
<td>Y-m-d 23:59:59</td>
</tr>
<tr>
<td>w</td>
<td>Monday of the week 00:00:00</td>
<td>Sunday of the week 23:59:59</td>
</tr>
<tr>
<td>M</td>
<td>First day of the month 00:00:00</td>
<td>Last day of the month 23:59:59</td>
</tr>
<tr>
<td>y</td>
<td>1st of January of the year 00:00:00</td>
<td>31st of December of the year 23:59:59</td>
</tr>
</tbody>
</table>
For example:

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Selected period</th>
</tr>
</thead>
<tbody>
<tr>
<td>now/d</td>
<td>now/d</td>
<td>00:00 - 23:59 today</td>
</tr>
<tr>
<td>now/d</td>
<td>now/d+1d</td>
<td>00:00 today - 23:59 tomorrow</td>
</tr>
<tr>
<td>now/w</td>
<td>now/w</td>
<td>Monday 00:00:00 - Sunday 23:59:59 this week</td>
</tr>
<tr>
<td>now-1y/w</td>
<td>now-1y/w</td>
<td>The week of Monday 00:00:00 - Sunday 23:59:59 one year ago</td>
</tr>
</tbody>
</table>

**Datepicker**

It is possible to pick a specific start/end date by clicking on the calendar icon next to the From/To fields. In this case, the date picker pop up will open.

Within the date picker, it is possible to navigate between the blocks of year/month/date using Tab and Shift+Tab. Keyboard arrows or arrow buttons allow to select the desired value. Pressing Enter (or clicking on the desired value) activates the choice.

Another way of controlling the displayed time is to highlight an area in the graph with the left mouse button. The graph will zoom into the highlighted area once you release the left mouse button.

In case no time value is specified or field is left blank, time value will be set to “00:00:00”. This doesn’t apply to today’s date selection: in that case time will be set to current value.

**Recent data vs longer periods**

For very recent data a **single** line is drawn connecting each received value. The single line is drawn as long as there is at least one horizontal pixel available for one value.

For data that show a longer period **three lines** are drawn - a dark green one shows the average, while a light pink and a light green line shows the maximum and minimum values at that point in time. The space between the highs and the lows is filled with yellow background.

**Working time (working days)** is displayed in graphs as a white background, while non-working time is displayed in gray (with the Original blue default frontend theme).

**Working time is always displayed in simple graphs, whereas displaying it in custom graphs is a user preference.**
Working time is not displayed if the graph shows more than 3 months.

Trigger lines

Simple triggers are displayed as lines with black dashes over trigger severity color -- take note of the blue line on the graph and the trigger information displayed in the legend. Up to 3 trigger lines can be displayed on the graph; if there are more triggers then the triggers with lower severity are prioritized. Triggers are always displayed in simple graphs, whereas displaying them in custom graphs is a user preference.

Generating from history/trends

Graphs can be drawn based on either item history or trends.

For the users who have frontend debug mode activated, a gray, vertical caption is displayed at the bottom right of a graph indicating where the data come from.

Several factors influence whether history of trends is used:

- longevity of item history. For example, item history can be kept for 14 days. In that case, any data older than the fourteen days will be coming from trends.
- data congestion in the graph. If the amount of seconds to display in a horizontal graph pixel exceeds 3600/16, trend data are displayed (even if item history is still available for the same period).
- if trends are disabled, item history is used for graph building - if available for that period. This is supported starting with Zabbix 2.2.1 (before, disabled trends would mean an empty graph for the period even if item history was available).

Absence of data

For items with a regular update interval, nothing is displayed in the graph if item data are not collected.

However, for trapper items and items with a scheduled update interval (and regular update interval set to 0), a straight line is drawn leading up to the first collected value and from the last collected value to the end of graph; the line is on the level of the first/last value respectively.

Switching to raw values

A dropdown on the upper right allows to switch from the simple graph to the Values/500 latest values listings. This can be useful for viewing the numeric values making up the graph.

The values represented here are raw, i.e. no units or postprocessing of values is used. Value mapping, however, is applied.

Known issues

See known issues for graphs.
2 Custom graphs

Overview

Custom graphs, as the name suggests, offer customization capabilities.

While simple graphs are good for viewing data of a single item, they do not offer configuration capabilities.

Thus, if you want to change graph style or the way lines are displayed or compare several items, for example, incoming and outgoing traffic in a single graph, you need a custom graph.

Custom graphs are configured manually.

They can be created for a host or several hosts or for a single template.

Configuring custom graphs

To create a custom graph, do the following:

- Go to Configuration → Hosts (or Templates)
- Click on Graphs in the row next to the desired host or template
- In the Graphs screen click on Create graph
- Edit graph attributes

All mandatory input fields are marked with a red asterisk.

Graph attributes:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| Name            | Unique graph name. Expression **macros** are supported in this field, but only with avg, last, min and max functions, with time as parameter (for example, `{avg(/host/key,1h})`). {HOST.HOST<1-9>} macros are supported for the use within this macro, referencing the first, second, third, etc. host in the graph, for example `{avg(/{HOST.HOST2}/key,1h})`. Note that referencing the first host with this macro is redundant, as the first host can be referenced implicitly, for example `{avg(/key,1h})`.
<p>| Width            | Graph width in pixels (for preview and pie/exploded graphs only).           |
| Height           | Graph height in pixels.                                                     |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph type</strong></td>
<td><strong>Normal</strong> - normal graph, values displayed as lines&lt;br&gt;<strong>Stacked</strong> - stacked graph, filled areas displayed&lt;br&gt;<strong>Pie</strong> - pie graph&lt;br&gt;<strong>Exploded</strong> - “exploded” pie graph, portions displayed as “cut out” of the pie</td>
</tr>
<tr>
<td>Show legend</td>
<td>Checking this box will set to display the graph legend.</td>
</tr>
<tr>
<td>Show working time</td>
<td>If selected, non-working hours will be shown with a gray background. Not available for pie and exploded pie graphs.</td>
</tr>
<tr>
<td>Show triggers</td>
<td>If selected, simple triggers will be displayed as lines with black dashes over trigger severity color. Not available for pie and exploded pie graphs.</td>
</tr>
<tr>
<td>Percentile line (left)</td>
<td>Display percentile for left Y-axis. If, for example, 95% percentile is set, then the percentile line will be at the level where 95 percent of the values fall under. Displayed as a bright green line. Only available for normal graphs.</td>
</tr>
<tr>
<td>Percentile line (right)</td>
<td>Display percentile for right Y-axis. If, for example, 95% percentile is set, then the percentile line will be at the level where 95 percent of the values fall under. Displayed as a bright red line. Only available for normal graphs.</td>
</tr>
<tr>
<td>Y axis MIN value</td>
<td><strong>Calculated</strong> - Y axis minimum value will be automatically calculated&lt;br&gt;<strong>Fixed</strong> - fixed minimum value for Y-axis. Not available for pie and exploded pie graphs. Item - last value of the selected item will be the minimum value</td>
</tr>
<tr>
<td>Y axis MAX value</td>
<td><strong>Calculated</strong> - Y axis maximum value will be automatically calculated&lt;br&gt;<strong>Fixed</strong> - fixed maximum value for Y-axis. Not available for pie and exploded pie graphs. Item - last value of the selected item will be the maximum value</td>
</tr>
<tr>
<td>3D view</td>
<td>Enable 3D style. For pie and exploded pie graphs only.</td>
</tr>
<tr>
<td>Items</td>
<td>Items, data of which are to be displayed in this graph. Click on Add to select items. You can also select various displaying options (function, draw style, left/right axis display, color).</td>
</tr>
<tr>
<td>Sort order (0→100)</td>
<td>Draw order. 0 will be processed first. Can be used to draw lines or regions behind (or in front of) another. You can drag and drop items by the arrow at the beginning of a line to set the sort order or which item is displayed in front of the other.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the selected item is displayed as a link. Clicking on the link opens the list of other available items.</td>
</tr>
<tr>
<td>Type</td>
<td><strong>Simple</strong> - the value of the item is represented proportionally on the pie&lt;br&gt;<strong>Graph sum</strong> - the value of the item represents the whole pie Note that coloring of the “graph sum” item will only be visible to the extent that it is not taken up by “proportional” items.</td>
</tr>
<tr>
<td>Function</td>
<td>Select what values will be displayed when more than one value exists per vertical graph pixel for an item:&lt;br&gt;<strong>all</strong> - display all possible values (minimum, maximum, average) in the graph. Note that for shorter periods this setting has no effect; only for longer periods, when data congestion in a vertical graph pixel increases, ‘all’ starts displaying minimum, maximum, and average values. This function is only available for Normal graph type. See also: Generating graphs from history/trends.<strong>avg</strong> - display the average values&lt;br&gt;<strong>last</strong> - display the latest values. This function is only available if either Pie/Exploded pie is selected as graph type.&lt;br&gt;<strong>max</strong> - display the maximum values&lt;br&gt;<strong>min</strong> - display the minimum values</td>
</tr>
<tr>
<td>Draw style</td>
<td>Select the draw style (only available for normal graphs; for stacked graphs filled region is always used) to apply to the item data - Line, Bold line, Filled region, Dot, Dashed line, Gradient line.</td>
</tr>
<tr>
<td>Y axis side</td>
<td>Select the Y axis side to show the item data - Left, Right.</td>
</tr>
<tr>
<td>Color</td>
<td>Select the color to apply to the item data.</td>
</tr>
</tbody>
</table>
Graph preview

In the Preview tab, a preview of the graph is displayed so you can immediately see what you are creating.

Note that the preview will not show any data for template items.

In this example, pay attention to the dashed bold line displaying the trigger level and the trigger information displayed in the legend.

No more than 3 trigger lines can be displayed. If there are more triggers then the triggers with lower severity are prioritized for display.

If graph height is set as less than 120 pixels, no trigger will be displayed in the legend.

3 Ad-hoc graphs

Overview

While a simple graph is great for accessing data of one item and custom graphs offer customization options, none of the two allow to quickly create a comparison graph for multiple items with little effort and no maintenance.

To address this issue, since Zabbix 2.4 it is possible to create ad-hoc graphs for several items in a very quick way.

Configuration

To create an ad-hoc graph, do the following:

- Go to Monitoring → Latest data
- Use filter to display items that you want
- Mark checkboxes of the items you want to graph
• Click on Display stacked graph or Display graph buttons

### Latest data

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Last check</th>
<th>Last val</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>New host</td>
<td>CPU load average</td>
<td>05/24/2021 10:46:5... 0.96</td>
</tr>
<tr>
<td>□</td>
<td>Zabbix server</td>
<td>Load average (1m avg)</td>
<td>05/24/2021 10:47:1... 0.73</td>
</tr>
<tr>
<td>□</td>
<td>Zabbix server</td>
<td>Load average (15m avg)</td>
<td>05/24/2021 10:47:1... 0.93</td>
</tr>
<tr>
<td>✓</td>
<td>Zabbix server</td>
<td>Load average (5m avg)</td>
<td>05/24/2021 10:47:1... 0.93</td>
</tr>
</tbody>
</table>

2 selected  | Display stacked graph  | Display graph |

Your graph is created instantly:

**Note that to avoid displaying too many lines in the graph, only the average value for each item is displayed (min/max value lines are not displayed). Triggers and trigger information is not displayed in the graph.**

In the created graph window you have the **time period selector** available and the possibility to switch from the "normal" line graph to a stacked one (and back).

### 4 Aggregation in graphs
Overview

The aggregation functions, available in the graph widget of the dashboard, allow displaying an aggregated value for the chosen interval (5 minutes, an hour, a day), instead of all values.

The aggregation options are as follows:

- min
- max
- avg
- count
- sum
- first (first value displayed)
- last (last value displayed)

The most exciting use of data aggregation is the possibility to create nice side-by-side comparisons of data for some period:

![Graph with aggregation options](image)

When hovering over a point in time in the graph, date and time is displayed, in addition to items and their aggregated values. Items are displayed in parentheses, prefixed by the aggregation function used. Note that this is the date and time of the point in the graph, not of the actual values.

Configuration

The options for aggregation are available in data set settings when configuring a graph widget.
You may pick the aggregation function and the time interval. As the data set may comprise several items, there is also another option allowing to show aggregated data for each item separately or for all data set items as one aggregated value.

Use cases

Average request count to Nginx server

View the average request count per second per day to the Nginx server:

- add the request count per second item to the data set
- select the aggregate function `avg` and specify interval `1d`
- a bar graph is displayed, where each bar represents the average number of requests per second per day

Minimum weekly disk space among clusters

View the lowest disk space among clusters over a week.

- add to the data set: `hosts cluster*`, key “Free disk space on /data”
- select the aggregate function `min` and specify interval `1w`
- a bar graph is displayed, where each bar represents the minimum disk space per week for each /data volume of the cluster

2 Network maps

Overview

If you have a network to look after, you may want to have an overview of your infrastructure somewhere. For that purpose, you can create maps in Zabbix - of networks and of anything you like.

All users can create network maps. The maps can be public (available to all users) or private (available to selected users).

Proceed to configuring a network map.

1 Configuring a network map

Overview

Configuring a map in Zabbix requires that you first create a map by defining its general parameters and then you start filling the actual map with elements and their links.

You can populate the map with elements that are a host, a host group, a trigger, an image, or another map.

Icons are used to represent map elements. You can define the information that will be displayed with the icons and set that recent problems are displayed in a special way. You can link the icons and define information to be displayed on the links.

You can add custom URLs to be accessible by clicking on the icons. Thus you may link a host icon to host properties or a map icon to another map.

Maps are managed in Monitoring → Maps, where they can be configured, managed and viewed. In the monitoring view, you can click on the icons and take advantage of the links to some scripts and URLs.

Network maps are based on vector graphics (SVG) since Zabbix 3.4.

Public and private maps

All users in Zabbix (including non-admin users) can create network maps. Maps have an owner - the user who created them. Maps can be made public or private.

- Public maps are visible to all users, although to see it the user must have read access to at least one map element. Public maps can be edited in case a user/ user group has read-write permissions for this map and at least read permissions to all elements of the corresponding map including triggers in the links.

- Private maps are visible only to their owner and the users/user groups the map is shared with by the owner. Regular (non-Super admin) users can only share with the groups and users they are members of. Admin level users can see private maps regardless of being the owner or belonging to the shared user list. Private maps can be edited by the owner of the map and in case a user/ user group has read-write permissions for this map and at least read permissions to all elements of the corresponding map including triggers in the links.

Map elements that the user does not have read permission to are displayed with a grayed-out icon and all textual information on the element is hidden. However, the trigger label is visible even if the user has no permission to the trigger.

To add an element to the map the user must also have at least read permission to it.

Creating a map
To create a map, do the following:

- Go to Monitoring ➔ Maps
- Go to the view with all maps
- Click on Create map

You can also use the Clone and Full clone buttons in the configuration form of an existing map to create a new map. Clicking on Clone will retain general layout attributes of the original map, but no elements. Full clone will retain both the general layout attributes and all elements of the original map.

The **Map** tab contains general map attributes:
All mandatory input fields are marked with a red asterisk.

General map attributes:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td>Admin (Zabbix Administrator)</td>
</tr>
<tr>
<td>Name</td>
<td>Local network</td>
</tr>
<tr>
<td>Width</td>
<td>680</td>
</tr>
<tr>
<td>Height</td>
<td>600</td>
</tr>
<tr>
<td>Background image</td>
<td>No image</td>
</tr>
<tr>
<td>Automatic icon mapping</td>
<td>&lt;manual&gt;</td>
</tr>
<tr>
<td>Icon highlight</td>
<td>Yes</td>
</tr>
<tr>
<td>Mark elements on trigger status change</td>
<td>Yes</td>
</tr>
<tr>
<td>Display problems</td>
<td>Expand single problem</td>
</tr>
<tr>
<td>Advanced labels</td>
<td>Yes</td>
</tr>
<tr>
<td>Host group label type</td>
<td>Label</td>
</tr>
<tr>
<td>Host label type</td>
<td>Label</td>
</tr>
<tr>
<td>Trigger label type</td>
<td>Status only</td>
</tr>
<tr>
<td>Map label type</td>
<td>Label</td>
</tr>
<tr>
<td>Image label type</td>
<td>Nothing</td>
</tr>
<tr>
<td>Map element label location</td>
<td>Bottom</td>
</tr>
<tr>
<td>Problem display</td>
<td>All</td>
</tr>
<tr>
<td>Minimum severity</td>
<td>Not classified</td>
</tr>
<tr>
<td>Show suppressed problems</td>
<td>No</td>
</tr>
</tbody>
</table>

**URLs**

<table>
<thead>
<tr>
<th>Name</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latest data</td>
<td><a href="https://localhost/zabbix/latest.php">https://localhost/zabbix/latest.php</a></td>
</tr>
</tbody>
</table>

**Add**  **Cancel**
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td>Name of map owner.</td>
</tr>
<tr>
<td>Name</td>
<td>Unique map name.</td>
</tr>
<tr>
<td>Width</td>
<td>Map width in pixels.</td>
</tr>
<tr>
<td>Height</td>
<td>Map height in pixels.</td>
</tr>
<tr>
<td>Background image</td>
<td>Use background image:</td>
</tr>
<tr>
<td></td>
<td><strong>No image</strong> - no background image (white background)</td>
</tr>
<tr>
<td></td>
<td><strong>Image</strong> - selected image to be used as a background image. No scaling is performed. You may use a geographical map or any other image to enhance your map.</td>
</tr>
<tr>
<td>Automatic icon mapping</td>
<td>You can set to use an automatic icon mapping, configured in Administration → General → Icon mapping. Icon mapping allows mapping certain icons against certain host inventory fields.</td>
</tr>
<tr>
<td>Icon highlighting</td>
<td>If you check this box, map elements will receive highlighting. Elements with an active trigger will receive a round background, in the same color as the highest severity trigger. Moreover, a thick green line will be displayed around the circle, if all problems are acknowledged. Elements with “disabled” or “in maintenance” status will get a square background, gray and orange respectively.</td>
</tr>
<tr>
<td>Mark elements on trigger status change</td>
<td>A recent change of trigger status (recent problem or resolution) will be highlighted with markers (inward-pointing red triangles) on the three sides of the element icon that are free of the label. Markers are displayed for 30 minutes.</td>
</tr>
<tr>
<td>Display problems</td>
<td>Select how problems are displayed with a map element:</td>
</tr>
<tr>
<td></td>
<td><strong>Expand single problem</strong> - if there is only one problem, the problem name is displayed. Otherwise, the total number of problems is displayed.</td>
</tr>
<tr>
<td></td>
<td><strong>Number of problems</strong> - the total number of problems is displayed.</td>
</tr>
<tr>
<td></td>
<td><strong>Number of problems and expand most critical one</strong> - the name of the most critical problem and the total number of problems is displayed.</td>
</tr>
<tr>
<td></td>
<td>‘Most critical’ is determined based on problem severity and, if equal, problem event ID (higher ID or later problem displayed first). For a trigger map element it is based on problem severity and if equal, trigger position in the trigger list. In case of multiple problems of the same trigger, the most recent one will be displayed.</td>
</tr>
<tr>
<td>Advanced labels</td>
<td>If you check this box you will be able to define separate label types for separate element types.</td>
</tr>
<tr>
<td>Map element label type</td>
<td>Label type used for map elements:</td>
</tr>
<tr>
<td></td>
<td><strong>Label</strong> - map element label</td>
</tr>
<tr>
<td></td>
<td><strong>IP address</strong> - IP address</td>
</tr>
<tr>
<td></td>
<td><strong>Element name</strong> - element name (for example, host name)</td>
</tr>
<tr>
<td></td>
<td><strong>Status only</strong> - status only (OK or PROBLEM)</td>
</tr>
<tr>
<td></td>
<td><strong>Nothing</strong> - no labels are displayed</td>
</tr>
<tr>
<td>Map element label location</td>
<td>Label location in relation to the map element:</td>
</tr>
<tr>
<td></td>
<td><strong>Bottom</strong> - beneath the map element</td>
</tr>
<tr>
<td></td>
<td><strong>Left</strong> - to the left</td>
</tr>
<tr>
<td></td>
<td><strong>Right</strong> - to the right</td>
</tr>
<tr>
<td></td>
<td><strong>Top</strong> - above the map element</td>
</tr>
<tr>
<td>Problem display</td>
<td>Display problem count as:</td>
</tr>
<tr>
<td></td>
<td><strong>All</strong> - full problem count will be displayed</td>
</tr>
<tr>
<td></td>
<td><strong>Separated</strong> - unacknowledged problem count will be displayed separated as a number of the total problem count</td>
</tr>
<tr>
<td></td>
<td><strong>Unacknowledged only</strong> - only the unacknowledged problem count will be displayed</td>
</tr>
<tr>
<td>Minimum trigger severity</td>
<td>Problems below the selected minimum severity level will not be displayed on the map.</td>
</tr>
<tr>
<td></td>
<td>For example, with Warning selected, changes with Information and Not classified level triggers will not be reflected in the map.</td>
</tr>
<tr>
<td></td>
<td>This parameter is supported starting with Zabbix 2.2.</td>
</tr>
<tr>
<td>Show suppressed problems</td>
<td>Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance.</td>
</tr>
<tr>
<td>URLs</td>
<td>URLs for each element type can be defined (with a label). These will be displayed as links when a user clicks on the element in the map viewing mode.</td>
</tr>
<tr>
<td></td>
<td>Macros can be used in map URL names and values. For a full list, see supported macros and search for ‘map URL names and values’.</td>
</tr>
</tbody>
</table>

Sharing

The **Sharing** tab contains the map type as well as sharing options (user groups, users) for private maps:
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Select map type:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Private</strong> - map is visible only to selected user groups and users</td>
</tr>
<tr>
<td></td>
<td>- <strong>Public</strong> - map is visible to all</td>
</tr>
<tr>
<td><strong>List of user group shares</strong></td>
<td>Select user groups that the map is accessible to.</td>
</tr>
<tr>
<td></td>
<td>You may allow read-only or read-write access.</td>
</tr>
<tr>
<td><strong>List of user shares</strong></td>
<td>Select users that the map is accessible to.</td>
</tr>
<tr>
<td></td>
<td>You may allow read-only or read-write access.</td>
</tr>
</tbody>
</table>

When you click on Add to save this map, you have created an empty map with a name, dimensions, and certain preferences. Now you need to add some elements. For that, click on Constructor in the map list to open the editable area.

**Adding elements**

To add an element, click on Add next to **Map element**. The new element will appear at the top left corner of the map. Drag and drop it wherever you like.

Note that with the Grid option "On", elements will always align to the grid (you can pick various grid sizes from the dropdown, also hide/show the grid). If you want to put elements anywhere without alignment, turn the option to "Off". (Random elements can later again be aligned to the grid with the Align map elements button.)

Now that you have some elements in place, you may want to start differentiating them by giving names, etc. By clicking on the element, a form is displayed and you can set the element type, give a name, choose a different icon, etc.
Map element attributes:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Type of the element:</td>
</tr>
<tr>
<td>Host</td>
<td>- icon representing status of all triggers of the selected host</td>
</tr>
<tr>
<td>Map</td>
<td>- icon representing status of all elements of a map</td>
</tr>
<tr>
<td>Trigger</td>
<td>- icon representing status of one or more triggers</td>
</tr>
<tr>
<td>Host group</td>
<td>- icon representing status of all triggers of all hosts belonging to the selected group</td>
</tr>
<tr>
<td>Image</td>
<td>- an icon, not linked to any resource</td>
</tr>
<tr>
<td><strong>Label</strong></td>
<td>Icon label, any string.</td>
</tr>
<tr>
<td><strong>Label location</strong></td>
<td>Label location in relation to the icon:</td>
</tr>
<tr>
<td>Default</td>
<td>- map’s default label location</td>
</tr>
<tr>
<td>Bottom</td>
<td>- beneath the icon</td>
</tr>
<tr>
<td>Left</td>
<td>- to the left</td>
</tr>
<tr>
<td>Right</td>
<td>- to the right</td>
</tr>
<tr>
<td>Top</td>
<td>- above the icon</td>
</tr>
<tr>
<td>Expression macros</td>
<td>supported macros and search for 'map element labels'.</td>
</tr>
</tbody>
</table>

Macros and multi-line strings can be used.

For a full list of supported macros, see supported macros.
### Parameter Description

**Host**
Enter the host if the element type is ‘Host’. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. Scroll down to select. Click on ‘x’ to remove the selected.

**Map**
Select the map, if the element type is ‘Map’.

**Triggers**
If the element type is ‘Trigger’, select one or more triggers in the New triggers field below and click on Add.

The order of selected triggers can be changed, but only within the same severity of triggers.

Multiple trigger selection also affects {HOST.*} macro resolution both in the construction and view modes.

// 1 In construction mode // the first displayed {HOST.*} macros will be resolved depending on the first trigger in the list (based on trigger severity).

// 2 View mode // depends on the Display problems parameter in General map attributes.

* If Expand single problem mode is chosen the first displayed {HOST.*} macros will be resolved depending on the latest detected problem trigger (not mattering the severity) or the first trigger in the list (in case no problem detected);

* If Number of problems and expand most critical one mode is chosen the first displayed {HOST.*} macros will be resolved depending on the trigger severity.

**Host group**
Enter the host group if the element type is ‘Host group’. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Scroll down to select. Click on ‘x’ to remove the selected.

**Tags**
Specify tags to limit the number of problems displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive.

There are several operators available for each condition:

- **Exists** - include the specified tag names
- **Equals** - include the specified tag names and values (case-sensitive)
- **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)
- **Does not exist** - exclude the specified tag names
- **Does not equal** - exclude the specified tag names and values (case-sensitive)
- **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)

There are two calculation types for conditions:

- **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition
- **Or** - enough if one condition is met

This field is available for host and host group element types.

**Automatic icon selection**
In this case an icon mapping will be used to determine which icon to display.

**Icons**
You can choose to display different icons for the element in these cases: default, problem, maintenance, disabled.

**Coordinate X**
X coordinate of the map element.

**Coordinate Y**
Y coordinate of the map element.

**URLs**
Element-specific URLs can be set for the element. These will be displayed as links when a user clicks on the element in the map viewing mode. If the element has its own URLs and there are map level URLs for its type defined, they will be combined in the same menu.

Macros can be used in map element names and values. For a full list, see supported macros and search for ‘map URL names and values’.

Added elements are not automatically saved. If you navigate away from the page, all changes may be lost.

Therefore it is a good idea to click on the Update button in the top right corner. Once clicked, the changes are saved regardless of what you choose in the following popup.

Selected grid options are also saved with each map.

Selecting elements

To select elements, select one and then hold down Ctrl to select the others.

You can also select multiple elements by dragging a rectangle in the editable area and selecting all elements in it.

Once you select more than one element, the element property form shifts to the mass-update mode so you can change attributes of selected elements in one go. To do so, mark the attribute using the checkbox and enter a new value for it. You may use macros
Linking elements

Once you have put some elements on the map, it is time to start linking them. To link two elements you must first select them. With the elements selected, click on Add next to Link.

With a link created, the single element form now contains an additional Links section. Click on Edit to edit link attributes.
Link attributes:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Label that will be rendered on top of the link. Expression <strong>macros</strong> are supported in this field, but only with <strong>avg</strong>, <strong>last</strong>, <strong>min</strong> and <strong>max</strong> functions, with time as parameter (for example, `{avg(/host/key,1h))}.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Connect to</td>
<td>The element that the link connects to.</td>
</tr>
<tr>
<td>Type (OK)</td>
<td>Default link style:&lt;br&gt;- Line - single line&lt;br&gt;- Bold line - bold line&lt;br&gt;- Dot - dots&lt;br&gt;- Dashed line - dashed line</td>
</tr>
<tr>
<td>Color (OK)</td>
<td>Default link color.</td>
</tr>
<tr>
<td>Link indicators</td>
<td>List of triggers linked to the link. In case a trigger has status PROBLEM, its style is applied to the link.</td>
</tr>
</tbody>
</table>

Moving and copy-pasting elements

Several selected elements can be **moved** to another place in the map by clicking on one of the selected elements, holding down the mouse button, and moving the cursor to the desired location.

One or more elements can be **copied** by selecting the elements, then clicking on a selected element with the right mouse button and selecting Copy from the menu.

To paste the elements, click on a map area with the right mouse button and select Paste from the menu. The Paste without external links option will paste the elements retaining only the links that are between the selected elements.

Copy-pasting works within the same browser window. Keyboard shortcuts are not supported.

Adding shapes

In addition to map elements, it is also possible to add some shapes. Shapes are not map elements; they are just a visual representation. For example, a rectangle shape can be used as a background to group some hosts. Rectangle and ellipse shapes can be added.

To add a shape, click on Add next to Shape. The new shape will appear at the top left corner of the map. Drag and drop it wherever you like.

A new shape is added with default colors. By clicking on the shape, a form is displayed and you can customize the way a shape looks, add text, etc.
To select shapes, select one and then hold down Ctrl to select the others. With several shapes selected, common properties can be mass updated, similarly as with elements.

Text can be added in the shapes. Expression macros are supported in the text, but only with \texttt{avg}, \texttt{last}, \texttt{min} and \texttt{max} functions, with time as parameter (for example, \texttt{{?avg(/host/key,1h)}}).

To display text only the shape can be made invisible by removing the shape border (select 'None' in the Border field). For example, take note of how the \texttt{(MAP.NAME)} macro, visible in the screenshot above, is actually a rectangle shape with text, which can be seen when clicking on the macro:
{MAP.NAME} resolves to the configured map name when viewing the map.

If hyperlinks are used in the text, they become clickable when viewing the map.

Line wrapping for text is always “on” within shapes. However, within an ellipse, the lines are wrapped as though the ellipse were a rectangle. Word wrapping is not implemented, so long words (words that do not fit the shape) are not wrapped, but are masked (constructor page) or clipped (other pages with maps).

Adding lines

In addition to shapes, it is also possible to add some lines. Lines can be used to link elements or shapes in a map.

To add a line, click on Add next to Shape. A new shape will appear at the top left corner of the map. Select it and click on Line in the editing form to change the shape into a line. Then adjust line properties, such as line type, width, color, etc.
Ordering shapes and lines

To bring one shape in front of the other (or vice versa) click on the shape with the right mouse button bringing up the map shape menu.
2 Host group elements

Overview

This section explains how to add a “Host group” type element when configuring a network map.

Configuration

All mandatory input fields are marked with a red asterisk.

This table consists of parameters typical for Host group element type:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Select Type of the element:</td>
</tr>
<tr>
<td>Host group</td>
<td>- icon representing the status of all triggers of all hosts belonging to the</td>
</tr>
<tr>
<td></td>
<td>selected group</td>
</tr>
<tr>
<td>Show</td>
<td>Show options:</td>
</tr>
<tr>
<td>Host group</td>
<td>- selecting this option will result as one single icon displaying corresponding</td>
</tr>
<tr>
<td></td>
<td>information about the certain host group</td>
</tr>
<tr>
<td>Host group elements</td>
<td>- selecting this option will result as multiple icons displaying corresponding</td>
</tr>
<tr>
<td></td>
<td>information about every single element (host) of the certain host group</td>
</tr>
</tbody>
</table>
Parameter | Description
--- | ---
Area type | This setting is available if the “Host group elements” parameter is selected:
  - **Fit to map** - all host group elements are equally placed within the map
  - **Custom size** - a manual setting of the map area for all the host group elements to be displayed

Area size | This setting is available if “Host group elements” parameter and “Area type” parameter are selected:
  - **Width** - numeric value to be entered to specify map area width
  - **Height** - numeric value to be entered to specify map area height

Placing algorithm | **Grid** - only available option of displaying all the host group elements

Label | Icon label, any string.
  - **Macros** and multi-line strings can be used in labels.
  - If the type of the map element is “Host group” specifying certain macros has an impact on the map view displaying corresponding information about every single host. For example, if \{HOST.IP\} macro is used, the edit map view will only display the macro \{HOST.IP\} itself while map view will include and display each host’s unique IP address

**Viewing host group elements**

This option is available if the "Host group elements" show option is chosen. When selecting "Host group elements" as the show option, you will at first see only one icon for the host group. However, when you save the map and then go to the map view, you will see that the map includes all the elements (hosts) of the certain host group:

<table>
<thead>
<tr>
<th>Map editing view</th>
<th>Map view</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Network maps" /></td>
<td><img src="image2.png" alt="Maps" /></td>
</tr>
</tbody>
</table>

Notice how the \{HOST.NAME\} macro is used. In map editing, the macro name is unresolved, while in map view all the unique names of the hosts are displayed.

**3 Link indicators**

**Overview**

You can assign some triggers to a link between elements in a network map. When these triggers go into a problem state, the link can reflect that.
When you configure a link, you set the default link type and color. When you assign triggers to a link, you can assign different link types and colors with these triggers.

Should any of these triggers go into a problem state, their link style and color will be displayed on the link. So maybe your default link was a green line. Now, with the trigger in the problem state, your link may become bold red (if you have defined it so).

Configuration

To assign triggers as link indicators, do the following:

- select a map element
- click on Edit in the Links section for the appropriate link
- click on Add in the Link indicators block and select one or more triggers

All mandatory input fields are marked with a red asterisk.

Added triggers can be seen in the Link indicators list.

You can set the link type and color for each trigger directly from the list. When done, click on Apply, close the form and click on Update to save the map changes.

Display

In Monitoring → Maps the respective color will be displayed on the link if the trigger goes into a problem state.
If multiple triggers go into a problem state, the problem with the highest severity will determine the link style and color. If multiple triggers with the same severity are assigned to the same map link, the one with the lowest ID takes precedence. Note also that:

1. Minimum trigger severity and Show suppressed problem settings from map configuration affect which problems are taken into account.
2. In the case of triggers with multiple problems (multiple problem generation), each problem may have a severity that differs from trigger severity (changed manually), may have different tags (due to macros), and may be suppressed.

3 Dashboards

Dashboards and their widgets provide a strong visualization platform with such tools as modern graphs, maps, slideshows, and many more.

4 Host dashboards
Overview

Host dashboards look similar to global dashboards, however, host dashboards display data about the host only. Host dashboards have no owner.

Host dashboards are configured on the template level and then are generated for a host, once the template is linked to the host. Widgets of host dashboards can only be copied to host dashboards of the same template. Widgets from global dashboards cannot be copied onto host dashboards.

Host dashboards cannot be configured or directly accessed in the Monitoring → Dashboard section, which is reserved for global dashboards. The ways to access host dashboards are listed below in this section.

Zabbix server health

When viewing host dashboards you may switch between the configured dashboards using the dropdown in the upper right corner. To switch to Monitoring→Hosts section, click All hosts navigation link below the dashboard name in the upper left corner.

Widgets of the host dashboards cannot be edited.

Note that host dashboards used to be host screens before Zabbix 5.2. When importing an older template containing screens, the screen import will be ignored.

Accessing host dashboards

Access to host dashboards is provided:

- From the host menu that is available in many frontend locations:
  - click on the host name and then select Dashboards from the drop-down menu
8 Templates and template groups

Overview

The use of templates is an excellent way of reducing one’s workload and streamlining the Zabbix configuration. A template is a set of entities that can be conveniently applied to multiple hosts.

The entities may be:

- items
- triggers
- graphs
- dashboards
- low-level discovery rules
- web scenarios

As many hosts in real life are identical or fairly similar so it naturally follows that the set of entities (items, triggers, graphs,...) you have created for one host, may be useful for many. Of course, you could copy them to each new host, but that would be a lot of manual work. Instead, with templates you can copy them to one template and then apply the template to as many hosts as needed.

When a template is linked to a host, all entities (items, triggers, graphs,...) of the template are added to the host. Templates are assigned to each individual host directly (and not to a host group).

Templates are often used to group entities for particular services or applications (like Apache, MySQL, PostgreSQL, Postfix...) and then applied to hosts running those services.

Another benefit of using templates is when something has to be changed for all the hosts. Changing something on the template level once will propagate the change to all the linked hosts.

Templates are organized in template groups.
Proceed to creating and configuring a template.

**9 Templates out of the box**

**Overview**

Zabbix strives to provide a growing list of useful out-of-the-box templates. Out-of-the-box templates come preconfigured and thus are a useful way for speeding up the deployment of monitoring jobs.

The templates are available:

- In new installations - in Configuration → Templates;
- If you are upgrading from previous versions, you can find these templates in the templates directory of the downloaded latest Zabbix version. While in Configuration → Templates you can import them manually from this directory.
- It is also possible to download the template from Zabbix git repository directly (make sure the template is compatible with your Zabbix version).

Please use the sidebar to access information about specific template types and operation requirements.

See also:

- Template import
- Linking a template

**HTTP template operation**

Steps to ensure correct operation of templates that collect metrics with HTTP agent:

1. Create a host in Zabbix and specify an IP address or DNS name of the monitoring target as the main interface. This is needed for the (HOSTCONN) macro to resolve properly in the template items.
2. Link the template to the host created in step 1 (if the template is not available in your Zabbix installation, you may need to import the template’s .xml file first - see Templates out-of-the-box section for instructions).
3. Adjust the values of mandatory macros as needed.
4. Configure the instance being monitored to allow sharing data with Zabbix - see instructions in the Additional steps/comments column.

This page contains only a minimum set of macros and setup steps that are required for proper template operation. A detailed description of a template, including the full list of macros, items and triggers, is available in the template’s Readme.md file (accessible by clicking on a template name).

<table>
<thead>
<tr>
<th>Template</th>
<th>Mandatory macros</th>
<th>Additional steps/comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Apache by HTTP</strong></td>
<td><strong>${APACHE.STATUS.HOST}</strong> - the hostname or IP address of Apache status page (default: 127.0.0.1).&lt;br&gt;<strong>${APACHE.STATUS.PATH}</strong> - the URL path (default: server-status?auto).&lt;br&gt;<strong>${APACHE.STATUS.PORT}</strong> - the port of Apache status page (default: 80).&lt;br&gt;<strong>${APACHE.STATUS.SCHEME}</strong> - the request scheme. Supported: http (default), https.</td>
<td>Apache module mod_status should be set (see Apache documentation for details).&lt;br&gt;To check availability, run: http -HM 2&gt;/dev/null | grep status_module. Apache configuration example: &lt;Location &quot;/server-status&quot;&gt; SetHandler server-status Require host example.com &lt;/Location&gt;</td>
</tr>
</tbody>
</table>
| **Asterisk by HTTP** | **${AMI.PORT}** - AMI port number for checking service availability (default: 8088).<br>**${AMI.SECRET}** - the Asterisk Manager secret (default: zabbix).<br>**${AMI.URL}** - the Asterisk Manager API URL in the format <scheme>://<host>:<port>/<prefix>/rawman (default: http://asterisk:8088/asterisk/rawman).<br>**${AMI.USERNAME}** - the Asterisk Manager name. | 1. Enable the mini-HTTP Server.<br>2. Add the option webenabled=yes to the general section of manager.conf file.<br>3. Create Asterisk Manager user in the Asterisk instance.
<table>
<thead>
<tr>
<th>Template</th>
<th>Mandatory macros</th>
<th>Additional steps/comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AWS EC2 by HTTP</strong></td>
<td>{$AWS.ACCESS.KEY.ID} - access key ID (default: not set).</td>
<td>In the AWS account, create an IAM policy for the Zabbix role with required permissions (see template’s Readme.md for details).</td>
</tr>
<tr>
<td></td>
<td>{$AWS.REGION} - Amazon EC2 Region code (default: us-west-1).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$AWS.EC2.INSTANCE.ID} - EC2 instance ID (default: not set).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$AWS.SECRET.ACCESS.KEY} - secret access key (default: not set).</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>ClickHouse by HTTP</strong></td>
<td>Create a ClickHouse user with a ‘web’ profile and permission to view databases (see ClickHouse documentation for details).</td>
</tr>
<tr>
<td></td>
<td>{$CLICKHOUSE.PORT} - the port of ClickHouse HTTP endpoint (default: 8123).</td>
<td>See template’s Readme.md file for a ready-to-use zabbix.xml file configuration.</td>
</tr>
<tr>
<td></td>
<td>{$CLICKHOUSE.SCHEME} - the request scheme. Supported: http (default), https.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$CLICKHOUSE.USER}, {$CLICKHOUSE.PASSWORD} - ClickHouse login credentials (default username: zabbix, password: zabbix_pass).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you don’t need authentication, remove headers from HTTP agent type items.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Cloudflare by HTTP</strong></td>
<td>Cloudflare API Tokens are available in the Cloudflare account under My Profile → API Tokens. Zone ID is available in the Cloudflare account under Account Home → Site.</td>
</tr>
<tr>
<td></td>
<td>{$CLOUDFLARE.API.TOKEN} - Cloudflare API token value (default: &lt;change&gt;).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$CLOUDFLARE.ZONE_ID} - Cloudflare Site Zone ID (default: &lt;change&gt;).</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>CockroachDB by HTTP</strong></td>
<td>Internal node metrics are collected from Prometheus /_status/vars endpoint. Node health metrics are collected from /health and /health?ready=1 endpoints.</td>
</tr>
<tr>
<td></td>
<td>{$COCKROACHDB.API.PORT} - the port of CockroachDB API and Prometheus endpoint. (default: 8080).</td>
<td>The template doesn’t require usage of session token.</td>
</tr>
<tr>
<td></td>
<td>{$COCKROACHDB.API.SCHEME} - the request scheme. Supported: http (default), https.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>DELL PowerEdge R720 by HTTP, DELL PowerEdge R740 by HTTP, DELL PowerEdge R820 by HTTP, DELL PowerEdge R840 by HTTP</strong></td>
<td>Depending on your CockroachDB version and configuration, some metrics may not be collected. In the Dell iDRAC interface of your server: 1. Enable Redfish API. 2. Create a user with monitoring with read-only permissions.</td>
</tr>
<tr>
<td></td>
<td>{$API.URL} - Dell iDRAC Redfish API URL in the format &lt;scheme&gt;://&lt;host&gt;:&lt;port&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(default: &lt;Put your URL here&gt;)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$API.USER}, {$API.PASSWORD} - Dell iDRAC login credentials (default: not set).</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Elasticsearch Cluster by HTTP</strong></td>
<td>Depending on your Envoy Proxy instance version and configuration, some metrics may not be collected.</td>
</tr>
<tr>
<td></td>
<td>{$ELASTICSEARCH.PORT} - the port of the Elasticsearch host (default: 9200).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$ELASTICSEARCH.SCHEME} - the request scheme. Supported: http (default), https.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$ELASTICSEARCH.USERNAME}, {$ELASTICSEARCH.PASSWORD} - login credentials, required only if used for Elasticsearch authentication.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Envoy Proxy by HTTP</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$SENVOY.METRICS.PATH} - the path from where to retrieve metrics in the Prometheus format (default: /stats/prometheus).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$SENVOY.URL} - Envoy Proxy instance URL (default: <a href="http://localhost:9901">http://localhost:9901</a>)</td>
<td></td>
</tr>
<tr>
<td>Template</td>
<td>Mandatory macros</td>
<td>Additional steps/comments</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Etcd by HTTP</strong></td>
<td>{$ETCD.PORT} - the port used by Etcd API endpoint (default: 2379).</td>
<td>Metrics are collected from /metrics endpoint; to specify the endpoint’s location use --listen-metrics-urls flag (see Etcd documentation for details).</td>
</tr>
<tr>
<td></td>
<td>{$ETCD.SCHEME} - the request scheme. Supported: http (default), https.</td>
<td>To verify, whether Etcd is configured to allow metric collection, run:</td>
</tr>
<tr>
<td></td>
<td>{$ETCD.USER}, {$ETCD.PASSWORD} - login credentials, required only if used for Etcd authentication.</td>
<td>curl -L <a href="http://localhost:2379/metrics">http://localhost:2379/metrics</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>To check, if Etcd is accessible from Zabbix proxy or Zabbix server run:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>curl -L http://%&lt;etcd_node_address&gt;%:2379/metrics%</td>
</tr>
<tr>
<td><strong>GitLab by HTTP</strong></td>
<td>{$GITLAB.PORT} - the port of GitLab web endpoint (default: 80)</td>
<td>The template should be added to each node with Etcd.</td>
</tr>
<tr>
<td></td>
<td>{$GITLAB.URL} - GitLab instance URL (default: localhost)</td>
<td>This template works with self-hosted GitLab instances; metrics are collected from the /metrics endpoint.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To access the metrics, the client IP address must be explicitly allowed (see GitLab documentation for details).</td>
</tr>
<tr>
<td><strong>Hadoop by HTTP</strong></td>
<td>{$HADOOP.NAMENODE.HOST} - the Hadoop NameNode host IP address or FQDN (default: NameNode).</td>
<td>Note, that certain metrics may not be available for a particular GitLab instance version and configuration.</td>
</tr>
<tr>
<td></td>
<td>{$HADOOP.NAMENODE.PORT} - the Hadoop NameNode web-UI port (default: 9870).</td>
<td>Metrics are collected by polling the Hadoop API remotely using an HTTP agent and JSONPath preprocessing. Zabbix server (or proxy) executes direct requests to ResourceManager, NodeManagers, NameNode, DataNodes APIs.</td>
</tr>
<tr>
<td></td>
<td>{$HADOOP.RESOURCEMANAGER.HOST} - the Hadoop ResourceManager host IP address or FQDN (default: ResourceManager).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$HADOOP.RESOURCEMANAGER.PORT} - the Hadoop ResourceManager web-UI port (default: 8088).</td>
<td></td>
</tr>
<tr>
<td><strong>HAProxy by HTTP</strong></td>
<td>{$HAPROXY.STATS.PATH} - the path of HAProxy Stats page (default: stats).</td>
<td>HAProxy Stats page should be set up (see HAProxy blog post for details or template's Readme.md for configuration example).</td>
</tr>
<tr>
<td></td>
<td>{$HAPROXY.STATS.PORT} - the port of the HAProxy Stats host or container (default: 8404).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$HAPROXY.STATS.SCHEME} - the request scheme. Supported: http (default), https.</td>
<td></td>
</tr>
<tr>
<td><strong>HashiCorp Consul Node by HTTP</strong></td>
<td>{$CONSUL.NODE.API.URL} - Consul instance URL (default: <a href="http://localhost:8500">http://localhost:8500</a>).</td>
<td>Internal service metrics are collected from /v1/agent/metrics endpoint.</td>
</tr>
<tr>
<td></td>
<td>{$CONSUL.TOKEN} - Consul authorization token (default: &lt;PUT YOUR AUTH TOKEN&gt;).</td>
<td>Prometheus format must be enabled for export metrics. See Consul documentation for details.</td>
</tr>
<tr>
<td><strong>HashiCorp Consul Cluster by HTTP</strong></td>
<td>{$CONSUL.API.PORT} - Consul API port, used in node LLD (default: 8500).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$CONSUL.API.SCHEME} - Consul API scheme, used in node LLD (default: http).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$CONSUL.CLUSTER.URL} - Consul cluster URL (default: <a href="http://localhost:8500">http://localhost:8500</a>).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{$CONSUL.TOKEN} - Consul authorization token (default: &lt;PUT YOUR AUTH TOKEN&gt;).</td>
<td></td>
</tr>
<tr>
<td>Template</td>
<td>Mandatory macros</td>
<td>Additional steps/comments</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| HashiCorp Vault by HTTP              | {$VAULT.API.PORT} - the port on which the Vault listens for API requests (default: 8200).  
   {$VAULT.API.SCHEME} - the API request scheme. Supported: http (default), https.  
   {$VAULT.HOST} - Vault host name (default: <PUT YOUR VAULT HOST>).  
   {$VAULT.TOKEN} - Vault authorization token (default: <PUT YOUR AUTH TOKEN>). | 1. Configure the Vault API (see official documentation for details).  
2. Create a Vault service token, then copy and paste it into {$VAULT.TOKEN} macro value in Zabbix.  

| Hikvision camera by HTTP             | {$HIKVISION.ISAPI_PORT} - ISAPI port on a device (default: 80).  
   {$USER}, {$PASSWORD} - camera login credentials (default username: admin, password: 1234). |                                                                                                                                                                                                                          |
| InfluxDB by HTTP                     | {$INFLUXDB.API.TOKEN} - InfluxDB API authorization token (default: ").  
See InfluxDB documentation for details.                                                                                                                                                                                                                                                                 |
| HPE MSA 2040 Storage by HTTP/HPE MSA 2060 Storage by HTTP | {$HPE.MSA.API.PORT} - connection port for API (default: 443)  
   {$HPE.MSA.API.SCHEME} - connection scheme for API. Supported: http, https (default)  
   {$HPE.MSA.API.USERNAME}, {$HPE.MSA.API.PASSWORD} - API credentials (default username: zabbix, password:”). | Create a separate Storage user (for example, zabbix) with monitor role and specify username and password in template macros.                                                                                                                                                                                                                                                                 |
| HPE Primera by HTTP                  | {$HPE.PRIMERA.API.PORT} - connection port for API (default: 443)  
   {$HPE.PRIMERA.API.SCHEME} - connection scheme for WSAPI. Supported: http, https (default)  
   {$HPE.PRIMERA.API.USERNAME}, {$HPE.PRIMERA.API.PASSWORD} - WSAPI credentials (default username: zabbix, password:”). | Create a separate Storage user (for example, zabbix) with browse role and enable it for all domains.  
To start WSAPI server, log in to the CLI as a user whose role has the wsapi_set right, then run:  
startwsapi  
To check WSAPI state, run:  
showwsapi                                                                                                                                                                                                 |
| HPE Synergy by HTTP                  | {$HPE.SYNERGY.API.PORT} - connection port for API (default: 443)  
   {$HPE.SYNERGY.API.SCHEME} - connection scheme for API. Supported: http, https (default)  
   {$HPE.SYNERGY.API.USERNAME}, {$HPE.SYNERGY.API.PASSWORD} - API credentials (default username: zabbix, password:”). |                                                                                                                                                                                                                          |
<table>
<thead>
<tr>
<th>Template</th>
<th>Mandatory macros</th>
<th>Additional steps/comments</th>
</tr>
</thead>
</table>
| Jenkins by HTTP        | `{$JENKINS.API.KEY}` - API key to access Metrics Servlet; required for common metrics (default: "").<br>`{$JENKINS.API.TOKEN}` - API token for HTTP BASIC authentication; required for monitoring computers and builds (default: ").<br>`{$JENKINS.URL}` - Jenkins URL in the format <scheme>://<host>:<port>; required for monitoring computers and builds (default: ").<br>`{$JENKINS.USER}` - username for HTTP BASIC authentication; required for monitoring computers and builds (default: zabbix). | Metrics are collected by requests to Metrics API.  
For common metrics: install and configure Metrics plugin parameters according to the official documentation. Issue an API key for access to the Metrics Servlet, then use it as `{$JENKINS.API.KEY}` macro value.  
For monitoring computers and builds: create an API token for the Jenkins user that will be used for monitoring, then use it as `{$JENKINS.API.TOKEN}` macro value. See Jenkins documentation for details. |
| Kubernetes API server by HTTP | `{$KUBE.API.SERVER.URL}` - instance URL (default: http://localhost:8086/metrics).<br>`{$KUBE.API.TOKEN}` - API authorization token (default: "). | The template requires Zabbix Helm Chart to be installed in your Kubernetes cluster.  
Internal metrics are collected from the /metrics endpoint.  
Use bearer API token for authorization. See Kubernetes documentation for details. |
| Kubernetes Controller manager by HTTP | `{$KUBE.CONTROLLER.SERVER.URL}` - instance URL (default: http://localhost:10252/metrics).<br>`{$KUBE.CONTROLLER.TOKEN}` - API authorization token (default: "). | The template requires Zabbix Helm Chart to be installed in your Kubernetes cluster.  
Internal metrics are collected from the /metrics endpoint.  
Use bearer API token for authorization. See Kubernetes documentation for details. |
| Kubernetes kubelet by HTTP | `{$KUBE.KUBELET.URL}` - instance URL (default: https://localhost:10250).<br>`{$KUBE.API.TOKEN}` - API authorization token (default: "). | The template requires Zabbix Helm Chart to be installed in your Kubernetes cluster.  
Internal metrics are collected from the /metrics endpoint.  
Use bearer API token for authorization. See Kubernetes documentation for details. |
| Kubernetes nodes by HTTP | `{$KUBE.API.ENDPOINT}` - Kubernetes API endpoint in the format <scheme>://<host>:<port>/api (default: not set).<br>`{$KUBE.API.TOKEN}` - API authorization token (default: "). | The template requires Zabbix Helm Chart to be installed in your Kubernetes cluster.  
To generate a service account token, run:  
kubectl get secret zabbix-service-account -n zabbix -o jsonpath={.data.token} | base64 -d  
See Kubernetes documentation for details. |
| Kubernetes Scheduler by HTTP | `{$KUBE.SCHEDULER.SERVER.URL}` - instance URL (default: http://localhost:10251/metrics).<br>`{$KUBE.SCHEDULER.TOKEN}` - Scheduler API authorization token (default: "). | The template contains additional macros, which can be used to filter out certain metrics of discovered worker nodes.  
The template requires Zabbix Helm Chart to be installed in your Kubernetes cluster.  
Internal metrics are collected from the /metrics endpoint.  
Use bearer API token for authorization. See Kubernetes documentation for details. |
<table>
<thead>
<tr>
<th>Template</th>
<th>Mandatory macros</th>
<th>Additional steps/comments</th>
</tr>
</thead>
</table>
| Kubernetes cluster state by HTTP    | {$KUBE.API.HOST} - Kubernetes API host (default: not set).  
    {$KUBE.API.PORT} - Kubernetes API port (default:6443).  
    {$KUBE.API.TOKEN} - API authorization token (default: "). | The template requires Zabbix Helm Chart to be installed in your Kubernetes cluster.  
Internal service metrics are collected from the kube-state-metrics endpoint.  
Use bearer API token for authorization. See Kubernetes documentation for details. |
| Microsoft SharePoint by HTTP         | {$SHAREPOINT.URL} - portal page URL, for example  
    http://sharepoint.companyname.local/ (default: ").  
    {$SHAREPOINT.ROOT} - a root directory; only the specified directory and all its subfolders will be monitored (default: /Shared Documents)  
    {$SHAREPOINT.USER}, {$SHAREPOINT.PASSWORD} - SharePoint login credentials (default: not set). | The template contains additional macros, which can be used to filter out certain metrics of discovered worker nodes.  
The template contains additional macros, which can be used to filter out certain dictionaries and types during LLD process (see template’s Readme.md for the description of available filter macros). |
| NetApp AFF A700 by HTTP              | {$URL} - AFF700 cluster URL address (default: ' ').  
    {$USERNAME}, {$PASSWORD} - AFF700 login credentials (default: not set). | Create a host for AFF A700 with cluster management IP as the Zabbix agent interface.                                                                                                                                 |
| NGiNX by HTTP                       | {$NGINX.STUB_STATUS.HOST} - the hostname or IP address of NGiNX stub_status host or container (default: localhost).  
    {$NGINX.STUB_STATUS.PATH} - the path of NGiNX stub_status page (default: basic_status).  
    {$NGINX.STUB_STATUS.PORT} - the port of NGiNX stub_status host or container (default: 80).  
    {$NGINX.STUB_STATUS.SCHEME} - the request scheme. Supported: http (default), https. | 'ngx_http_stub_status_module should be set up (see NGiNX documentation for details or template’s Readme.md for configuration example).  
To check availability, run:  
    nginx -V 2>&1 \| grep -o with-http_stub_status_module  
1. Enable NGINX Plus API (see NGINX documentation for details).  
2. Set the macro {$NGINX.API.ENDPOINT}  
3. If required, use other template macros to filter out discovery operations and discover only required zones and upstreams. |
| NGiNX Plus by HTTP                  | {$NGINX.API.ENDPOINT} - NGINX Plus API URL in the format <scheme>://<host>:<port>/<location>/ (default: '). | For instructions on getting the API key, see OpenWeatherMap documentation.  
geo coordinates - for example, 56.95,24.0833  
location name - for example, Chicago  
OpenWeatherMap location ID - download the ID list zip/postal code with a country code - for example, 94040,us  
To specify multiple locations, use the | delimiter.  
Example:  
    43.81821,7.76115\|Riga\|2643743\|94040,us |
<table>
<thead>
<tr>
<th>Template</th>
<th>Mandatory macros</th>
<th>Additional steps/comments</th>
</tr>
</thead>
</table>
| **PHP-FPM by HTTP** | **{$PHP_FPM.HOST}** - a hostname or an IP of PHP-FPM status host or container (default: localhost).  
**{$PHP_FPM.PING.PAGE}** - PHP-FPM ping page path (default: ping).  
**{$PHP_FPM.PORT}** - the port of PHP-FPM status host or container (default: 80).  
**{$PHP_FPM.PROCESS_NAME}** - PHP-FPM process name (default: php-fpm).  
**{$PHP_FPM.SCHEME}** - the request scheme. Supported: http (default), https.  
**{$PHP_FPM.STATUS.PAGE}** - PHP-FPM status page path (default: status). | 1. Open the php-fpm configuration file and enable the status page:  
```
pm.status_path = /status
ping.path = /ping
```
2. Validate the syntax: $ php-fpm7 -t  
3. Reload the php-fpm service.  
4. In the Nginx Server Block (virtual host) configuration file, add (see template’s Readme.md for an expanded example with comments):  
```
location ~ ^/(status|ping)$ {
    access_log off;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_index index.php;
    include fastcgi_params;
    fastcgi_pass 127.0.0.1:9000;
}
```
5. Check the syntax: $ nginx -t  
6. Reload Nginx  
7. Verify: curl -L 127.0.0.1/status
Create a separate user for monitoring, then generate an API token for this user.  
Grant the following access levels to the token and user:  
Check: ["perm","/","["Sys.Audit"]"]  
Check: ["perm","/nodes/{node}",["Sys.Audit"]]  
Check: ["perm","/vms/{vmid}",["VM.Audit"]]

| **Proxmox VE by HTTP** | **{$PVE.TOKEN.ID}** - API token that allows stateless access to most parts of the REST API (default: not set).  
**{$PVE.TOKEN.SECRET}** - secret key (default: not set).  
**{$PVE.URL.PORT}** - the port the server listens to (default: 8006). |  |  
|------------------------|---------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| **RabbitMQ cluster by HTTP** | **{$RABBITMQ.API.CLUSTER_HOST}** - the hostname or IP address of RabbitMQ cluster API endpoint (default: 127.0.0.1).  
**{$RABBITMQ.API.SCHEME}** - the request scheme. Supported: http (default), https.  
**{$RABBITMQ.API.USER},  
**{$RABBITMQ.API.PASSWORD}** - RabbitMQ login credentials (default username: zbx_monitor, password: zabbix). | Enable RabbitMQ management plugin (see RabbitMQ documentation).  
To create a RabbitMQ user with necessary permissions for monitoring, run:  
```
"rabbitmqctl add_user zbx_monitor <PASSWORD>"  
rabbitmqctl set_permissions -p / zbx_monitor  
```
If the cluster consists of several nodes, it is recommended to assign the cluster template to a separate balancing host. In case of a single-node installation, the cluster template can be assigned to the host with a node template. This template works with RabbitMQ cluster.  
Internal service metrics are collected from RabbitMQ management plugin.

| **TiDB by HTTP** | **{$TIDB.PORT}** - The port of TiDB server metrics web endpoint (default: 10080)  
**{$TIDB.URL}** - TiDB server URL (default: localhost). | This template works with TiDB server of PingCAP TiDB cluster.  
Internal service metrics are collected from TiDB /metrics endpoint and TiDB monitoring API.  

| **TiDB PD by HTTP** | **{$TIDB.PORT}** - The port of TiDB server metrics web endpoint (default: 2379)  
**{$TIDB.URL}** - TiDB server URL (default: localhost). | This template works with PD server of PingCAP TiDB cluster.  
Internal service metrics are collected from PD /metrics endpoint and TiDB monitoring API. |
Template | Mandatory macros | Additional steps/comments
--- | --- | ---
**TiDB TiKV by HTTP** | {$TIDB.PORT} - The port of TiDB server metrics web endpoint (default: 20180)  
{$TIDB.URL} - TiDB server URL (default: localhost). | This template works with TiKV server of PingCAP TiDB cluster. Internal service metrics are collected from TiKV /metrics endpoint.

**Travis CI by HTTP** | {$TRAVIS.API.TOKEN} - Travis API Token (default: not set)  
{$TRAVIS.API.URL} - Travis API URL (default: api.travis-ci.com). | Travis API authentication token can be found in the User → Settings → API authentication section.  
{$TRAVIS.API.URL} format for a private project is api.travis-ci.com.  
{$TRAVIS.API.URL} format for an enterprise project is api.example.com (replace example.com with the domain Travis CI is running on).

**VMWare SD-WAN VeloCloud by HTTP** | {$VELOCLOUD.TOKEN} - VMware SD-WAN Orchestrator API Token (default: "),  
{$VELOCLOUD.URL} - VMware SD-WAN Orchestrator URL, for example, velocloud.net (default: "). | API token should be created in the VMware SD-WAN Orchestrator (see VMware documentation for details).

**ZooKeeper by HTTP** | {$ZOOKEEPER.COMMAND_URL} - admin.commandURL; the URL for listing and issuing commands relative to the root URL (default: commands).  
{$ZOOKEEPER.PORT} - admin.serverPort; the port the embedded jetty server listens on (default: 8080).  
{$ZOOKEEPER.SCHEME} - the request scheme. Supported: http (default), https. | Metrics are collected from each ZooKeeper node by requests to AdminServer (enabled by default). See ZooKeeper documentation to enable or configure AdminServer.

---

**IPMI template operation**

IPMI templates do not require any specific setup. To start monitoring, link the template to a target host (if the template is not available in your Zabbix installation, you may need to import the template’s .xml file first - see Templates out-of-the-box section for instructions).

This page contains only a minimum set of macros and setup steps that are required for proper template operation. A detailed description of a template, including the full list of macros, items and triggers, is available in the template’s Readme.md file (accessible by clicking on a template name).

<table>
<thead>
<tr>
<th>Template</th>
<th>Mandatory macros</th>
<th>Additional steps/comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Chassis by IPMI</strong></td>
<td>{$IPMI.USER}, {$IPMI.PASSWORD} - credentials for access to BMC (default: none)</td>
<td>-</td>
</tr>
</tbody>
</table>

---

**JMX template operation**

Steps to ensure correct operation of templates that collect metrics by JMX:

1. Make sure Zabbix Java gateway is installed and set up properly.
2. Link the template to the target host. The host should have JMX interface set up.  
If the template is not available in your Zabbix installation, you may need to import the template’s .xml file first - see Templates out-of-the-box section for instructions.
3. Adjust the values of mandatory macros as needed.
4. Configure the instance being monitored to allow sharing data with Zabbix - see instructions in the Additional steps/comments column.

This page contains only a minimum set of macros and setup steps that are required for proper template operation. A detailed description of a template, including the full list of macros, items, and triggers, is available in the template’s Readme.md file.
<table>
<thead>
<tr>
<th>Template</th>
<th>Mandatory macros</th>
<th>Additional steps/comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache ActiveMQ by JMX</td>
<td>{$ACTIVEMQ.PORT} - port for JMX (default: 1099). &lt;br&gt;{$ACTIVEMQ.USERNAME}. &lt;br&gt;{$ACTIVEMQ.PASSWORD} - login credentials for JMX (default username: admin, password: activemq).</td>
<td>JMX access to Apache ActiveMQ should be enabled and configured per instructions in the official documentation.</td>
</tr>
<tr>
<td>Apache Cassandra by JMX</td>
<td>{$CASSANDRA.USER}. &lt;br&gt;{$CASSANDRA.PASSWORD} - Apache Cassandra login credentials (default username: admin, password: activemq).</td>
<td>JMX access to Apache Cassandra should be enabled and configured per instructions in the official documentation.</td>
</tr>
<tr>
<td>Apache Kafka by JMX</td>
<td>{$KAFKA.USER}. &lt;br&gt;{$KAFKA.PASSWORD} - Apache Kafka login credentials (default username: zabbix, password: zabbix)</td>
<td>JMX access to Apache Kafka should be enabled and configured per instructions in the official documentation.</td>
</tr>
<tr>
<td>Apache Tomcat by JMX</td>
<td>{$TOMCAT.USER}. &lt;br&gt;{$TOMCAT.PASSWORD} - Apache Tomcat login credentials; leave blank if Tomcat installation does not require authentication (default: not set).</td>
<td>JMX access to Apache Tomcat should be enabled and configured per instructions in the official documentation (choose the correct version).</td>
</tr>
<tr>
<td>GridGain by JMX</td>
<td>{$GRIDGAIN.USER}. &lt;br&gt;{$GRIDGAIN.PASSWORD} - GridGain login credentials (default username: zabbix, password: zabbix).</td>
<td>JMX access to GridGain In-Memory Computing Platform should be enabled and configured per instructions in the documentation.</td>
</tr>
<tr>
<td>Ignite by JMX</td>
<td>{$IGNITE.USER}. &lt;br&gt;{$IGNITE.PASSWORD} - Apache Ignite login credentials (default username: zabbix, password: &lt;secret&gt;).</td>
<td>Enable and configure JMX access to Apache Ignite.</td>
</tr>
<tr>
<td>WildFly Domain by JMX</td>
<td>{$WILDFLY.JMX.PROTOCOL} - JMX scheme (default: remote+http) &lt;br&gt;{$WILDFLY.USER}. &lt;br&gt;{$WILDFLY.PASSWORD} - WildFly login credentials (default username: zabbix, password: zabbix).</td>
<td>JMX tree hierarchy contains ClassLoader by default. Adding the following Java Virtual Machine option -DIGNITE_MBEAN_APPEND_CLASS_LOADER_ID=false will exclude one level with ClassLoader name. &lt;br&gt;Cache and Data Region metrics can be configured as needed - see Ignite documentation for details.</td>
</tr>
</tbody>
</table>

**ODBC template operation**

Steps to ensure correct operation of templates that collect metrics via **ODBC monitoring**:
1. Make sure that required ODBC driver is installed on Zabbix server or proxy.
2. **Link** the template to a target host (if the template is not available in your Zabbix installation, you may need to import the template’s .xml file first - see **Templates out-of-the-box** section for instructions).
3. Adjust the values of mandatory macros as needed.
4. Configure the instance being monitored to allow sharing data with Zabbix - see instructions in the Additional steps/comments column.

This page contains only a minimum set of macros and setup steps that are required for proper template operation. A detailed description of a template, including the full list of macros, items and triggers, is available in the template’s Readme.md file (accessible by clicking on a template name).

<table>
<thead>
<tr>
<th>Template</th>
<th>Mandatory macros</th>
<th>Additional steps/comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MSSQL by ODBC</strong></td>
<td>{$MSSQL.DSN} - the system data source name (default: &lt;Put your DSN here&gt;)</td>
<td>Create a Microsoft SQL user for monitoring and grant the user the following permissions: View Server State; View Any Definition (see Microsoft SQL documentation for details).</td>
</tr>
<tr>
<td></td>
<td>{$MSSQL.PORT} - the TCP port of Microsoft SQL Server (default: 1433)</td>
<td>The “Service’s TCP port state” item uses {HOST.CONN} and {$MSSQL.PORT} macros to check the availability of the Microsoft SQL instance.</td>
</tr>
<tr>
<td></td>
<td>{$MSSQL.USER}, {$MSSQL.PASSWORD} - Microsoft SQL login credentials (default: not set)</td>
<td></td>
</tr>
<tr>
<td><strong>MySQL by ODBC</strong></td>
<td>{$MYSQL.DSN} - the system data source name (default: &lt;Put your DSN here&gt;)</td>
<td>To grant required privileges to MySQL user that will be used for monitoring, run:</td>
</tr>
<tr>
<td></td>
<td>{$MYSQL.USER}, {$MYSQL.PASSWORD} - MySQL login credentials; password can be blank (default: not set)</td>
<td>GRANT USAGE,REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON %* % TO '&lt;username&gt;'@'%';%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See <a href="#">MYSQL documentation</a> for details.</td>
</tr>
</tbody>
</table>
Oracle by ODBC

**{$ORACLE.DSN}** - the system data source name (default: <Put your DSN here>)

**{$ORACLE.PORT}** - the TCP port of Oracle DB (default: 1521)

**{$ORACLE.USER}, ** **{$ORACLE.PASSWORD}** - Oracle login credentials (default: not set)

1. To create an Oracle user for monitoring, run:
   ```sql
   CREATE USER zabbix_mon IDENTIFIED BY <PASSWORD>;
   ```
   -- Grant access to the zabbix_mon user.
   ```sql
   GRANT CONNECT, CREATE SESSION TO zabbix_mon;
   GRANT SELECT ON V_$instance TO zabbix_mon;
   GRANT SELECT ON V_$database TO zabbix_mon;
   GRANT SELECT ON V_$sysmetric TO zabbix_mon;
   GRANT SELECT ON v$recovery_file_dest TO zabbix_mon;
   GRANT SELECT ON v$active_session_history TO zabbix_mon;
   GRANT SELECT ON v$osstat TO zabbix_mon;
   GRANT SELECT ON v$restore_point TO zabbix_mon;
   GRANT SELECT ON v$process TO zabbix_mon;
   GRANT SELECT ON v$datafile TO zabbix_mon;
   GRANT SELECT ON v$pgastat TO zabbix_mon;
   GRANT SELECT ON v$sgastat TO zabbix_mon;
   GRANT SELECT ON v$log TO zabbix_mon;
   GRANT SELECT ON v$archive_dest TO zabbix_mon;
   GRANT SELECT ON v$asm_diskgroup TO zabbix_mon;
   GRANT SELECT ON sys.dba_data_files TO zabbix_mon;
   GRANT SELECT ON DBA_TABLESPACES TO zabbix_mon;
   GRANT SELECT ON DBA_TABLESPACE_USAGE_METRICS TO zabbix_mon;
   GRANT SELECT ON DBA_USERS TO zabbix_mon;
   ```

2. Make sure, that ODBC connects to Oracle with session parameter NLS_NUMERIC_CHARACTERS= ',,'.

3. Add a new record to odbc.ini:
   ```ini
   [$ORACLE.DSN]
   Driver = Oracle 19 ODBC driver
   Servername = $ORACLE.DSN
   DSN = $ORACLE.DSN
   ```

4. Check the connection via isql:
   ```
isql $TNS_NAME $DB_USER $DB_PASSWORD
   ```

5. Configure Zabbix server or Zabbix proxy for Oracle ENV Usage. Edit or add a new file: /etc/sysconfig/zabbix-server, or for the proxy: /etc/sysconfig/zabbix-proxy. Then add the following lines to the file:
   ```
   export ORACLE_HOME=/usr/lib/oracle/19.6/client64
   export PATH=$PATH:$ORACLE_HOME/bin
   export LD_LIBRARY_PATH=$ORACLE_HOME/lib:/usr/lib64:/usr/lib:$ORACLE_HOME/lib
   export TNS_ADMIN=$ORACLE_HOME/network/admin
   ```

6. Restart Zabbix server or proxy.

---

### Standardized templates for network devices

**Overview**

In order to provide monitoring for network devices such as switches and routers, we have created two so-called models: for the
network device itself (its chassis basically) and for network interface.

Since Zabbix 3.4 templates for many families of network devices are provided. All templates cover (where possible to get these items from the device):

- Chassis fault monitoring (power supplies, fans and temperature, overall status)
- Chassis performance monitoring (CPU and memory items)
- Chassis inventory collection (serial numbers, model name, firmware version)
- Network interface monitoring with IF-MIB and EtherLike-MIB (interface status, interface traffic load, duplex status for Ethernet)

These templates are available:

- In new installations - in Configuration → Templates;
- If you are upgrading from previous versions, you can find these templates in the templates directory of the downloaded latest Zabbix version. While in Configuration → Templates you can import them manually from this directory.

If you are importing the new out-of-the-box templates, you may want to also update the @Network interfaces for discovery global regular expression to:

Result is FALSE: `^Software Loopback Interface`
Result is FALSE: `^(In)?[lL]oop[0-9_.]*$`
Result is FALSE: `^[0-9_.]*$`
Result is FALSE: `^[sS]ystem$`
Result is FALSE: `^[Nu0-9_.]*$`

...to filter out loopbacks and null interfaces on most systems.

Devices

List of device families for which templates are available:

<table>
<thead>
<tr>
<th>Template name</th>
<th>Vendor</th>
<th>Device family</th>
<th>Known models</th>
<th>OS</th>
<th>MIbs used</th>
<th>Tags</th>
</tr>
</thead>
</table>
| Alcatel Time
tra TimOS SNMP | Alcatel       | Alcatel Time
tra | ALCATEL SR 7750 | TimOS    | TIMETRA-SYSTEM-MIB, TIMETRA-CHASSIS-MIB        | Certified             |
<p>| Brocade FC SNMP | Brocade      | Brocade FC switches | Brocade 300 SAN Switch- | -        | SW-MIB, ENTITY-MIB                            | Performance, Fault    |
| Brocade_Foundry Stackable SNMP | Brocade, Foundry | Brocade ICX | Brocade ICX6610, Brocade ICX7250-48, Brocade ICX7450-48F | -        | FOUNDRY-SN-AGENT-MIB, FOUNDRY-SN-STACKING-MIB | Certified             |
| Brocade_Foundry Nonstackable SNMP | Brocade, Foundry | Brocade MLX, Foundry | Brocade MLXe, Foundry FLS648, Foundry FWSX424 | -        | FOUNDRY-SN-AGENT-MIB                          | Performance, Fault    |
| Cisco IOS SNMP | Cisco | Cisco IOS ver &gt; 12.2 3.5 | Cisco IOS C2950 | IOS      | CISCO-PROCESS-MIB, CISCO-MEMORY-POOL-MIB, CISCO-ENVMON-MIB | Certified             |</p>
<table>
<thead>
<tr>
<th>Template name</th>
<th>Vendor</th>
<th>Device family</th>
<th>Known models</th>
<th>OS</th>
<th>MIBs used</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS versions</td>
<td>Cisco</td>
<td>Cisco</td>
<td>12.0 3 T and 12.2 3.5</td>
<td>IOS</td>
<td>CISCO-PROCESS-MIB, CISCO-MEMORY-POOL-MIB, CISCO-ENVMON-MIB, OLD-CISCO-CPU-MIB, CISCO-MEMORY-POOL-MIB</td>
<td>Certified</td>
</tr>
<tr>
<td>Cisco IOS prior to 12.0 3 T SNMP</td>
<td>Cisco</td>
<td>Cisco IOS 12.0 3 T</td>
<td>-</td>
<td>IOS</td>
<td>DLINK-AGENT-MIB, EQUIPMENT-MIB, ENTITY-MIB</td>
<td>D-Link DES_DGS Switch SNMP</td>
</tr>
<tr>
<td>D-Link DES 7200 SNMP</td>
<td>D-Link</td>
<td>DES/DGX switches</td>
<td>D-Link DES-xxxx/DGS-xxxx/DLINK DGS-3420-26SC</td>
<td>-</td>
<td>D-Link DES 7206</td>
<td>D-Link DES 7200 SNMP</td>
</tr>
<tr>
<td>Huawei VRP SNMP</td>
<td>Huawei</td>
<td>Huawei VRP</td>
<td>S2352P-EI</td>
<td>-</td>
<td>-</td>
<td>Certified</td>
</tr>
<tr>
<td>Intel Qlogic Infiniband SNMP</td>
<td>Intel/QLogic</td>
<td>Intel/QLogic Infiniband devices</td>
<td>12300</td>
<td>-</td>
<td>ICS-CHASSIS-MIB</td>
<td>Fault Inventory</td>
</tr>
<tr>
<td>Juniper SNMP</td>
<td>Juniper</td>
<td>MX,SRX,EX models</td>
<td>Juniper MX240, Juniper EX4200-24F SX1036</td>
<td>JunOS</td>
<td>JUNIPER-MIB</td>
<td>Certified</td>
</tr>
<tr>
<td>Mellanox SNMP</td>
<td>Mellanox</td>
<td>Mellanox Infiniband devices</td>
<td>MLNX-OS</td>
<td>-</td>
<td>HOST-RESOURCES-MIB, ENTITY-MIB, ENTITY-SENSOR-MIB, MELLANOX-MIB</td>
<td>Certified</td>
</tr>
<tr>
<td>Template name</td>
<td>Vendor</td>
<td>Device family</td>
<td>Known models</td>
<td>OS</td>
<td>MIBs used</td>
<td>Tags</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>---------------</td>
<td>--------------</td>
<td>----</td>
<td>-----------</td>
<td>------</td>
</tr>
<tr>
<td>MikroTik CCR&lt;device model&gt; SNMP</td>
<td>MikroTik</td>
<td>MikroTik Cloud Routers (CCR series)</td>
<td>Separate dedicated templates are available for MikroTik CCR1009-7G-1C-1S+, MikroTik CCR1009-7G-1C-1S+PC, MikroTik CCR1009-7G-1C-PC, MikroTik CCR1016-12G, MikroTik CCR1016-12S-1S+, MikroTik CCR1036-12G-4S-EM, MikroTik CCR1036-12G-4S, MikroTik CCR1036-8G-2S+, MikroTik CCR1036-8G-2S+EM, MikroTik CCR1072-1G-8S+, MikroTik CCR2004-16G-2S+, MikroTik CCR2004-1G-12S+2XS</td>
<td>RouterOS</td>
<td>MIKROTIK-MIB, HOST-RESOURCES-MIB</td>
<td>Certified</td>
</tr>
<tr>
<td>Template name</td>
<td>Vendor</td>
<td>Device family</td>
<td>Known models</td>
<td>OS</td>
<td>MIBs used</td>
<td>Tags</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>---------------</td>
<td>--------------</td>
<td>----</td>
<td>-----------</td>
<td>------</td>
</tr>
</tbody>
</table>

MikroTik CSS<device model> SNMP

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Device family</th>
<th>Known models</th>
<th>OS</th>
<th>MIBs used</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>MikroTik</td>
<td>MikroTik Cloud Smart Switches (CSS series)</td>
<td>Separate dedicated templates are available for MikroTik CSS326-24G-25+RM, MikroTik CSS610-8G-25+IN</td>
<td>RouterOS</td>
<td>MIKROTIK-MIB, HOST-RESOURCES-MIB</td>
<td>Certified</td>
</tr>
</tbody>
</table>

MikroTik FiberBox SNMP

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Device family</th>
<th>Known models</th>
<th>OS</th>
<th>MIBs used</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>MikroTik</td>
<td>MikroTik FiberBox</td>
<td></td>
<td>RouterOS</td>
<td>MIKROTIK-MIB, HOST-RESOURCES-MIB</td>
<td>Certified</td>
</tr>
</tbody>
</table>

MikroTik hEX<device model> SNMP

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Device family</th>
<th>Known models</th>
<th>OS</th>
<th>MIBs used</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>MikroTik</td>
<td>MikroTik hEX</td>
<td>Separate dedicated templates are available for MikroTik hEX, MikroTik hEX lite, MikroTik hEX PoE, MikroTik hEX PoE lite, MikroTik hEX S</td>
<td>RouterOS</td>
<td>MIKROTIK-MIB, HOST-RESOURCES-MIB</td>
<td>Certified</td>
</tr>
<tr>
<td>Template name</td>
<td>Vendor</td>
<td>Device family</td>
<td>Known models</td>
<td>OS</td>
<td>MIBs used</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>MikroTik netPower &lt;device model&gt; SNMP</td>
<td>MikroTik</td>
<td>MikroTik netPower</td>
<td>Separate dedicated templates are available for MikroTik netPower 15FR, MikroTik netPower 16P SNMP, MikroTik netPower Lite 7R</td>
<td>RouterOS/SwitchOS Lite</td>
<td>MIKROTIK-MIB, HOST-RESOURCES-MIB</td>
</tr>
<tr>
<td>MikroTik PowerBox &lt;device model&gt; SNMP</td>
<td>MikroTik</td>
<td>MikroTik PowerBox</td>
<td>Separate dedicated templates are available for MikroTik PowerBox, MikroTik PowerBox Pro</td>
<td>RouterOS</td>
<td>MIKROTIK-MIB, HOST-RESOURCES-MIB</td>
</tr>
<tr>
<td>MikroTik RB&lt;device model&gt; SNMP</td>
<td>MikroTik</td>
<td>MikroTik RB series routers</td>
<td>Separate dedicated templates are available for MikroTik RB1100AHx4, MikroTik RB1100AHx4 Dude Edition, MikroTik RB2011iL-IN, MikroTik RB2011iL-RM, MikroTik RB2011iLS-IN, MikroTik RB2011UiAS-IN, MikroTik RB2011UiAS-RM, MikroTik RB260GS, MikroTik RB3011UiAS-RM, MikroTik RB4011iGS+RM, MikroTik RB5009UJG+S+IN</td>
<td>RouterOS</td>
<td>MIKROTIK-MIB, HOST-RESOURCES-MIB</td>
</tr>
<tr>
<td>MikroTik SNMP</td>
<td>MikroTik</td>
<td>MikroTik RouterOS devices</td>
<td>MikroTik CCR1016-12G, MikroTik RB2011UAS-2HnD, MikroTik 912UAG-5HPnD, MikroTik 941-2nD, MikroTik 951G-2HnD, MikroTik 1100AHx2</td>
<td>RouterOS</td>
<td>MIKROTIK-MIB, HOST-RESOURCES-MIB</td>
</tr>
<tr>
<td>QTech QSW SNMP</td>
<td>QTech</td>
<td>Qtech devices</td>
<td>QTECH-MIB, ENTITY-MIB</td>
<td>-</td>
<td>QTECH-MIB, ENTITY-MIB</td>
</tr>
<tr>
<td>Template name</td>
<td>Vendor</td>
<td>Device family</td>
<td>Known models</td>
<td>OS</td>
<td>MIBs used</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
<td>---------------------</td>
<td>--------------------------------</td>
<td>----------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Ubiquiti AirOS</td>
<td>Ubiquiti</td>
<td>Ubiquiti AirOS</td>
<td>NanoBridge, NanoStation</td>
<td>AirOS</td>
<td>FROGFOOT-RESOURCES-MIB,IEEE802dot11-MIB</td>
</tr>
<tr>
<td>SNMP</td>
<td></td>
<td>Wireless devices</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HP Comware</td>
<td>HP</td>
<td>HP (H3C) Comware</td>
<td>A5500-24G-45FP, Hi Switch,</td>
<td></td>
<td>HH3C-ENTITY-EXT-MIB, ENTITY-MIB</td>
</tr>
<tr>
<td>HH3C SNMP</td>
<td></td>
<td>HP ProCurve</td>
<td>I4900B Switch, 2626, HP J9728A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HP Enterprise Switch</td>
<td>HP</td>
<td>HP Enterprise Switch</td>
<td>2920-48G Switch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNMP</td>
<td></td>
<td>HP ProCurve</td>
<td>J4900B Switch, 2626, HP J9728A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TP-LINK SNMP</td>
<td>TP-LINK</td>
<td>TP-LINK</td>
<td>T2600G-28TS v2.0</td>
<td></td>
<td>TPLINK-SYSMONITOR-MIB, TPLINK-SYSINFO-MIB</td>
</tr>
<tr>
<td>Netgear Fastpath</td>
<td>Netgear</td>
<td>TP-LINK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNMP</td>
<td></td>
<td>TP-LINK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Netgear Fastpath</td>
<td></td>
<td>Netgear Fastpath</td>
<td>M5300-28G</td>
<td></td>
<td>FASTPATH-SWITCHING-MIB, FASTPATH-BOXSERVICES-PRIVATE-MIB</td>
</tr>
</tbody>
</table>

Template design

Templates were designed with the following in mind:

- User macros are used as much as possible so triggers can be tuned by the user;
- Low-level discovery is used as much as possible to minimize the number of unsupported items;
- All templates depend on Template ICMP Ping so all devices are also checked by ICMP;
- Items don’t use any MIBs - SNMP OIDs are used in items and low-level discoveries. So it’s not necessary to load any MIBs into Zabbix for templates to work;
- Loopback network interfaces are filtered when discovering as well as interfaces with ifAdminStatus = down(2);
- 64bit counters are used from IF-MIB::ifXTable where possible. If it is not supported, default 32bit counters are used instead;
- All discovered network interfaces have a trigger that controls its operational status (link).
  - If you do not want to monitor this condition for a specific interface create a user macro with context with the value 0.
  - For example:

```plaintext
where Gi0/0 is (#IFNAME). That way the trigger is not used any more for this specific interface.

* You can also change the default behavior for all triggers not to fire and activate this trigger only to limited number of interfaces like uplinks 400
```
Tags

- Performance - device family MIBs provide a way to monitor CPU and memory items;
- Fault - device family MIBs provide a way to monitor at least one temperature sensor;
- Inventory - device family MIBs provide a way to collect at least the device serial number and model name;
- Certified - all three main categories above are covered.

Zabbix agent 2 template operation

Steps to ensure correct operation of templates that collect metrics with Zabbix agent 2:

1. Make sure that the agent 2 is installed on the host, and that the installed version contains the required plugin. In some cases, you may need to upgrade the agent 2 first.
2. Link the template to a target host (if the template is not available in your Zabbix installation, you may need to import the template’s import file first - see Templates out-of-the-box section for instructions).
3. Adjust the values of mandatory macros as needed. Note, that user macros can be used to override configuration parameters.
4. Configure the instance being monitored to allow sharing data with Zabbix - see instructions in the Additional steps/comments column.

Zabbix agent 2 templates work in conjunction with the plugins. While the basic configuration can be done by simply adjusting user macros, the deeper customization can be achieved by configuring the plugin itself. For example, if a plugin supports named sessions, it is possible to monitor several entities of the same kind (e.g. MySQL1 and MySQL2) by specifying named session with own URI, username and password for each entity in the configuration file.

This page contains only a minimum set of macros and setup steps that are required for proper template operation. A detailed description of a template, including the full list of macros, items and triggers, is available in the template’s Readme.md file (accessible by clicking on a template name).
<table>
<thead>
<tr>
<th>Template name</th>
<th>Mandatory macros</th>
<th>Additional steps/comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ceph by Zabbix agent 2</td>
<td><code>{$CEPH.API.KEY}</code> - the API key (default: zabbix_pass). Required, if</td>
<td>Works with Ceph plugin; named sessions are supported.</td>
</tr>
<tr>
<td></td>
<td><code>{$CEPH.CONNSTRING}</code> is a URI. Must be empty, if</td>
<td>1. Configure the Ceph RESTful Module according to documentation.</td>
</tr>
<tr>
<td></td>
<td><code>{$CEPH.CONNSTRING}</code> is a session name.</td>
<td>2. Make sure a RESTful API endpoint is available for connection.</td>
</tr>
<tr>
<td></td>
<td><code>{$CEPH.USER}</code> - user to be used for monitoring (default: zabbix). Required, if</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>{$CEPH.CONNSTRING}</code> is a URI. Must be empty, if</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>{$CEPH.CONNSTRING}</code> is a session name.</td>
<td></td>
</tr>
<tr>
<td>Docker</td>
<td>-</td>
<td>Works with Docker plugin; named sessions are not supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To set path to Docker API endpoint edit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Plugins.Docker.Endpoint parameter in the agent 2 configuration file (default:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Plugins.Docker.Endpoint=unix:///var/run/docker.sock)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To test availability, run:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>zabbix_get -s docker-host -k docker.info</code></td>
</tr>
<tr>
<td>Memcached</td>
<td><code>{$MEMCACHED.CONN.URI}</code> - connection string in the URI format; port is optional;</td>
<td>Works with Memcached plugin; named sessions are supported.</td>
</tr>
<tr>
<td></td>
<td>password is not used.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If not set, the plugin’s default value is used:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tcp://localhost:11211.</td>
<td>To test availability, run:</td>
</tr>
<tr>
<td></td>
<td>Examples: tcp://127.0.0.1:11211, tcp://localhost,</td>
<td><code>zabbix_get -s memcached-host -k memcached.ping</code></td>
</tr>
<tr>
<td></td>
<td>unix:/var/run/memcached.sock.</td>
<td></td>
</tr>
<tr>
<td>Template name</td>
<td>Mandatory macros</td>
<td>Additional steps/comments</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MongoDB cluster by Zabbix agent 2</td>
<td><strong>${MONGODB.CONNSTRING}</strong> - connection string in the URI format; password is not used (default: tcp://localhost:27017). Can be a session name or a URI defined in the following format: %%% &lt;protocol(host:port)&gt;%%% For URI only TCP scheme is supported. Examples: MongoDB1, tcp://172.16.0.10 <strong>${MONGODB.USER}, ${MONGODB.PASSWORD}</strong> - MongoDB credentials (default: none). If not set and <strong>${MONGODB.CONNSTRING}</strong> is a URI, parameters from the configuration file will be used. Must be empty, if <strong>${MONGODB.CONNSTRING}</strong> is a session name.</td>
<td>Works with MongoDB plugin; named sessions are supported. For MongoDB configuration instructions, see plugins. To test availability, run: zabbix_get -s mongos.node -k 'mongodb.ping[&quot;${MONGODB.CONNSTRING}&quot;,&quot;${MONGODB.USER}&quot;,&quot;${MONGODB.PASSWORD}&quot;]'</td>
</tr>
<tr>
<td>MongoDB node by Zabbix agent 2</td>
<td><strong>${MONGODB.CONNSTRING}</strong> - connection string in the URI format; password is not used (default: tcp://localhost:27017). Can be a session name or a URI defined in the following format: %%% &lt;protocol(host:port)&gt;%%% For URI only TCP scheme is supported. Examples: MongoDB1, tcp://172.16.0.10 <strong>${MONGODB.USER}, ${MONGODB.PASSWORD}</strong> - MongoDB credentials (default: none). If not set and <strong>${MONGODB.CONNSTRING}</strong> is a URI, parameters from the configuration file will be used. Must be empty, if <strong>${MONGODB.CONNSTRING}</strong> is a session name.</td>
<td>Works with MongoDB plugin; named sessions are supported. For MongoDB configuration instructions, see plugins. To test availability, run: zabbix_get -s mongodb.node -k 'mongodb.ping[&quot;${MONGODB.CONNSTRING}&quot;,&quot;${MONGODB.USER}&quot;,&quot;${MONGODB.PASSWORD}&quot;]'</td>
</tr>
<tr>
<td>Template name</td>
<td>Mandatory macros</td>
<td>Additional steps/comments</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MySQL by Zabbix agent 2</td>
<td><strong>{MYSQL.DSN}</strong> - the system data source name of the MySQL instance (default: <code>&lt;Put your DSN&gt;</code>). Can be a session name or a URI defined in the following format: <code>%%&lt;protocol(host:port or /path/to/socket)/&gt;%%</code> For URI only TCP and Unix schemas are supported. Examples: MySQL1, tcp://localhost:3306, tcp://172.16.0.10, unix:/var/run/mysql.sock <strong>{MYSQL.USER}</strong>, <strong>{MYSQL.PASSWORD}</strong> - MySQL credentials (default: none). Required, if <code>{MYSQL.DSN}</code> is a URI. Must be empty, if <code>{MYSQL.DSN}</code> is a session name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Works with MySQL plugin; named sessions are supported.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>To grant required privileges to a MySQL user that will be used for monitoring, run: <code>GRANT USAGE,REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON *.* TO '&lt;username&gt;'@'%'</code>; See MySQL documentation for information about user privileges and Unix sockets.</td>
<td></td>
</tr>
<tr>
<td>Template name</td>
<td>Mandatory macros</td>
<td>Additional steps/comments</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Oracle by Zabbix agent 2      | **{ORACLE.CONNSTRING}** - connection string; can be a session name or a URI defined in the following format:  
<protocol(host:port or /path/to/socket)/>  
For URI only TCP schema is supported.  
Examples: Oracle1, tcp://localhost:1521  
**{ORACLE.SERVICE}** - Oracle Service name (default: ORA).  
Required, if  
{ORACLE.CONNSTRING} is a URI.  
Must be empty, if  
{ORACLE.CONNSTRING} is a session name.  
**{ORACLE.USER}**,  
**{ORACLE.PASSWORD}** - Oracle credentials (default username: zabbix, password: zabbix_password).  
Required, if  
{ORACLE.CONNSTRING} is a URI.  
Must be empty, if  
{ORACLE.CONNSTRING} is a session name. | Works with Oracle plugin; named sessions are supported.  
Install Oracle Instant Client.  
To create Oracle user with required privileges, run:  
CREATE USER zabbix_mon IDENTIFIED BY <PASSWORD>;  
-- Grant access to the zabbix_mon user.  
GRANT CONNECT, CREATE SESSION TO zabbix_mon;  
GRANT SELECT ON DBA_TABLESPACE_USAGE_METRICS TO zabbix_mon;  
GRANT SELECT ON DBA_TABLESPACES TO zabbix_mon;  
GRANT SELECT ON DBA_USERS TO zabbix_mon;  
GRANT SELECT ON SYS.DBA_DATA_FILES TO zabbix_mon;  
GRANT SELECT ON V$ACTIVE_SESSION_HISTORY TO zabbix_mon;  
GRANT SELECT ON V$ARCHIVE_DEST TO zabbix_mon;  
GRANT SELECT ON V$ASM_DISKGROUP TO zabbix_mon;  
GRANT SELECT ON V$DATABASE TO zabbix_mon;  
GRANT SELECT ON V$DATAFILE TO zabbix_mon;  
GRANT SELECT ON V$INSTANCE TO zabbix_mon;  
GRANT SELECT ON V$LOG TO zabbix_mon;  
GRANT SELECT ON V$OSSTAT TO zabbix_mon;  
GRANT SELECT ON V$PGASTAT TO zabbix_mon;  
GRANT SELECT ON V$PROCESS TO zabbix_mon;  
GRANT SELECT ON V$RECOVERY_FILE_DEST TO zabbix_mon;  
GRANT SELECT ON V$RESTORE_POINT TO zabbix_mon;  
GRANT SELECT ON V$SESSION TO zabbix_mon;  
GRANT SELECT ON V$SGASTAT TO zabbix_mon;  
GRANT SELECT ON V$SYSMETRIC TO zabbix_mon;  
GRANT SELECT ON V$SYSTEM_PARAMETER TO zabbix_mon;  
PostgreSQL Agent 2 | **{PG.URI}** - connection string; can be a session name or a URI defined in the following format:  
%%<protocol(host:port or /path/to/socket)/>%%.  
For URI only TCP and Unix schemas are supported.  
Examples: Postgres1, tcp://localhost:5432, tcp://172.16.0.10  
**{PG.USER}**,  
**{PG.PASSWORD}** - PostgreSQL credentials (default username: postgres, password: postgres).  
Required, if  
{PG.URI} is a URI.  
Must be empty, if  
{PG.URI} is a session name. | Works with PostgreSQL plugin; named sessions are supported.  
To create a user with required privileges, for PostgreSQL 10 and newer, run:  
CREATE USER 'zbx_monitor' IDENTIFIED BY '<password>';  
GRANT EXECUTE ON FUNCTION pg_catalog.pg_ls_dir(text) TO zbx_monitor;  
GRANT EXECUTE ON FUNCTION pg_catalog.pg_stat_file(text) TO zbx_monitor;  
Edit pg_hba.conf to allow connections from Zabbix agent (see PostgreSQL documentation for details). |
### Redis

- **${REDIS.CONN.URI}**
  - connection string in the URI format; port is optional; password is not used. If not set, the plugin’s default value is used: tcp://localhost:6379
  - Works with Redis plugin; named sessions are supported.
  - To test availability, run: `zabbix_get -s redis-master -k redis.ping`

### SMART by Zabbix agent 2 / SMART by Zabbix agent 2 active

- **${CERT.WEBSITE.HOSTNAME}**
  - the website’s DNS name for the connection (default: <Put DNS name>).
  - Sudo/root access rights to smartctl are required for the user executing Zabbix agent 2. The minimum required smartctl version is 7.1.
  - Disk discovery LLD rule finds all HDD, SSD, NVMe disks with S.M.A.R.T. enabled.
  - Attribute discovery LLD rule finds all Vendor Specific Attributes for each disk.
  - To skip some attributes, set regular expressions with disk names in `{${SMART.DISK.NAME.MATCHES}}` and with attribute IDs in `{${SMART.ATTRIBUTE.ID.MATCHES}}` on the host level.

### Systemd by Zabbix agent 2

- **${CERT.WEBSITE.HOSTNAME}**
  - No specific configuration is required.

### Website certificate by Zabbix agent 2

- **${CERT.WEBSITE.HOSTNAME}**
  - The website’s DNS name for the connection (default: <Put DNS name>).
  - Works with WebCertificate plugin; named sessions are not supported.
  - To test availability, run: `zabbix_get -s <zabbix_agent_addr> -k web.certificate.get[<website_DNS_name>]`
  - Create a separate host for the TLS/SSL certificate with Zabbix agent interface and link the template to this host.

---

### Zabbix agent template operation

Steps to ensure correct operation of templates that collect metrics with Zabbix agent:

1. Make sure that Zabbix agent is installed on the host. For active checks, also make sure that the host is added to the ‘ServerActive’ parameter of the agent configuration file.
2. Link the template to a target host (if the template is not available in your Zabbix installation, you may need to import the template’s .xml file first - see Templates out-of-the-box section for instructions).
3. Adjust the values of mandatory macros as needed.
4. Configure the instance being monitored to allow sharing data with Zabbix - see instructions in the Additional steps/comments column.

This page contains only a minimum set of macros and setup steps that are required for proper template operation. A detailed description of a template, including the full list of macros, items and triggers, is available in the template’s Readme.md file (accessible by clicking on a template name).

---

### Apache by Zabbix agent

- **${APACHE.STATUS.HOST}** - the hostname or IP address of Apache status page (default: 127.0.0.1)
- **${APACHE.STATUS.PATH}** - the URL path (default: server-status?auto)
- **${APACHE.STATUS.PORT}** - the port of Apache status page (default: 80)

- Apache module mod_status should be set (see Apache documentation for details).
  - To check availability, run: `http -M 2>/dev/null | grep status_module`
  - Apache configuration example:
  ```xml
  <Location "/server-status">
    SetHandler server-status
    Require host example.com
  </Location>
  ```

---

406
<table>
<thead>
<tr>
<th>Template name</th>
<th>Mandatory macros</th>
<th>Additional steps/comments</th>
</tr>
</thead>
</table>
| HAProxy by Zabbix agent                       | `{$HAPROXY.STATS.PATH}` - the path of HAProxy Stats page (default: stats)  
 `{$HAPROXY.STATS.PORT}` - the port of HAProxy Stats host or container (default: 8404)  
 `{$HAPROXY.STATS.SCHEME}` - the scheme of HAProxy Stats page. Supported: http (default), https  
 HAProxy Stats page should be set up (see HAProxy blog post for details or template’s Readme.md for configuration example). |                                                                                                                                                                         |
| IIS by Zabbix agent / IIS by Zabbix agent     | `{$IIIS.PORT}` - the port IIS Server listens on (default: 80)  
 `{$IIIS.SERVICE}` - the service for port check (default: http). See net.tcp.service section for details.                                                                 | The server should have the following roles:  
 Web Server  
 IIS Management Scripts and Tools  
 See IIS documentation for details.  
 Note, that the template doesn’t provide information about Windows services state. It is recommended to use it together with OS Windows by Zabbix agent or OS Windows by Zabbix agent active template. |
| Microsoft Exchange Server 2016 by Zabbix      | `{$NGINX.STUB_STATUS.HOST}` - the hostname or IP address of Nginx stub_status host or container (default: localhost)  
 `{$NGINX.STUB_STATUS.PATH}` - the path of Nginx stub_status page (default: basic_status)  
 `{$NGINX.STUB_STATUS.PORT}` - the port of Nginx stub_status host or container (default: 80)  
 ngx_http_stub_status_module should be set up (see Nginx documentation for details or template’s Readme.md for configuration example).  
 To check availability, run:  
 nginx -V 2>&1 \| grep -o with-http_stub_status_module   
 PHP-FPM by Zabbix agent                        | `{$PHP_FPM.HOST}` - a hostname or an IP of PHP-FPM status host or container (default: localhost)  
 `{$PHP_FPM.PING.PAGE}` - PHP-FPM ping page path (default: ping)  
 `{$PHP_FPM.PORT}` - the port of PHP-FPM status host or container (default: 80)  
 `{$PHP_FPM.PROCESS_NAME}` - PHP-FPM process name (default: php-fpm)  
 `{$PHP_FPM.STATUS.PAGE}` - PHP-FPM status page path (default: status)  
 1. Open the php-fpm configuration file and enable the status page:  
 pm.status_path = /status  
 ping.path = /ping  
 2. Validate the syntax: $ php-fpm7 -t  
 3. Reload the php-fpm service.  
 4. In the Nginx Server Block (virtual host) configuration file, add (see template’s Readme.md for an expanded example with comments):  
 location ~ ^/(status|ping) {  
 access_log off;  
 fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;  
 fastcgi_index index.php;  
 include fastcgi_params;  
 fastcgi_pass 127.0.0.1:9000;  
 }  
 5. Check the syntax: $ nginx -t  
 6. Reload Nginx  
 7. Verify: curl -L 127.0.0.1/status |
<table>
<thead>
<tr>
<th>Template name</th>
<th>Mandatory macros</th>
<th>Additional steps/comments</th>
</tr>
</thead>
</table>
| RabbitMQ cluster by Zabbix agent | {$RABBITMQ.API.CLUSTER_HOST} - the hostname or IP address of RabbitMQ cluster API endpoint (default: 127.0.0.1)  
{$RABBITMQ.API.USER}, {$RABBITMQ.API.PASSWORD} - RabbitMQ login credentials (default username: zbx_monitor, password: zabbix) | Enable RabbitMQ management plugin (see RabbitMQ documentation).  
To create a RabbitMQ user with necessary permissions for monitoring, run:  
"rabbitmqctl add_user zbx_monitor <PASSWORD> "  
rabbitmqctl set_permissions -p /  
zbx_monitor %% "*" "*" ".*"  
rabbitmqctl set_user_tags zbx_monitor monitoring  
If the cluster consists of several nodes, it is recommended to assign the cluster template to a separate balancing host. In case of a single-node installation, the cluster template can be assigned to the host with a node template. |
| MySQL by Zabbix agent | {$MYSQL.HOST} - the hostname or IP address of MySQL host or container (default: 127.0.0.1 (since 6.2.2)/localhost (before 6.2.2))  
{$MYSQL.PORT} - the database service port (default: 3306) | 1. If necessary, add the path to the mysql and mysqldadmin utilities to the global environment variable PATH.  
2. Copy the template_db_mysql.conf file from templates directory of Zabbix into folder with Zabbix agent configuration (/etc/zabbix/zabbix_agentd.d/ by default) and restart Zabbix agent.  
3. Create MySQL user zbx_monitor. To grant required privileges to the user, run:  
GRANT USAGE,REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON %% *.* TO '<username>'@'%';  
(see MYSQL documentation for details).  
4. Create .my.cnf in the home directory of Zabbix agent for Linux (/var/lib/zabbix by default ) or my.cnf in c:\ for Windows. The file must have three strings:  
[client]  
"user='zbx_monitor' "  
"password='<password>' " |
### PostgreSQL

<table>
<thead>
<tr>
<th>Template name</th>
<th>Mandatory macros</th>
<th>Additional steps/comments</th>
</tr>
</thead>
</table>
| PostgreSQL    | `{$PG.DB}` - the database name to connect to the server (default: postgres)  
  `{$PG.HOST}` - the database server host or socket directory (default: 127.0.0.1)  
  `{$PG.PORT}` - the database server port (default: 5432)  
  `{$PG.USER}` - the database username (default: zbx_monitor) | 1. Create a read-only user zbx_monitor with proper access to PostgreSQL server. For PostgreSQL 10 and newer, run:  
  `CREATE USER zbx_monitor WITH PASSWORD '<PASSWORD>' INHERIT;`  
  `GRANT pg_monitor TO zbx_monitor;`  
  For older PostgreSQL versions, run:  
  `CREATE USER zbx_monitor WITH PASSWORD '<PASSWORD>';`  
  `GRANT SELECT ON pg_stat_database TO zbx_monitor;`  
  2. Copy `postgresql` to Zabbix agent home directory (`/var/lib/zabbix`).  
  3. Copy `template_db_postgresql.conf` from templates directory of Zabbix to Zabbix agent configuration directory (`/etc/zabbix/zabbix_agentd.d/`) and restart Zabbix agent.  
  4. Edit `pg_hba.conf` to allow connections from Zabbix agent (see PostgreSQL documentation for details).  
  Row examples:  
  `host all zbx_monitor 127.0.0.1/32 trust`  
  `host all zbx_monitor 0.0.0.0/0 md5`  
  `host all zbx_monitor ::0/0 md5`  
  5. To monitor a remote server, create a `.pgpass` file in Zabbix agent home directory (`/var/lib/zabbix/`) and add rows with the instance, port, database, user and password information (see PostgreSQL documentation for details).  
  Row examples:  
  `<REMOTE_HOST1>:5432:postgres:zbx_monitor:<PASSWORD>`  
  `*:5432:postgres:zbx_monitor:<PASSWORD>` |

### 10 Notifications upon events

**Overview**

Assuming that we have configured some items and triggers and now are getting some events happening as a result of triggers changing state, it is time to consider some actions.

To begin with, we would not want to stare at the triggers or events list all the time. It would be much better to receive notification if something significant (such as a problem) has happened. Also, when problems occur, we would like to see that all the people concerned are informed.

That is why sending notifications is one of the primary actions offered by Zabbix. Who and when should be notified upon a certain event can be defined.

To be able to send and receive notifications from Zabbix you have to:

- define some media
- configure an action that sends a message to one of the defined media

Actions consist of conditions and operations. Basically, when conditions are met, operations are carried out. The two principal operations are sending a message (notification) and executing a remote command.

For discovery and autoregistration created events, some additional operations are available. Those include adding or removing a host, linking a template etc.
1 Media types

Overview

Media are the delivery channels used for sending notifications and alerts from Zabbix.

You can configure several media types:

- E-mail
- SMS
- Custom alertscripts
- Webhook

Media types are configured in Administration → Media types.

Some media types come pre-defined in the default dataset. You just need to finetune their parameters to get them working.

It is possible to test if a configured media type works, by clicking on Test in the last column (see Media type testing for more details).

To create a new media type, click on the Create media type button. A media type configuration form is opened.

Common parameters

Some parameters are common for all media types.

In the Media type tab the common general attributes are:
See the individual pages of media types for media-specific parameters.

The **Message templates** tab allows to set default notification messages for all or some of the following event types:

- Problem
- Problem recovery
- Problem update
- Service
- Service recovery
- Service update
- Discovery
- Autoregistration
- Internal problem
- Internal problem recovery

**Media types**

<table>
<thead>
<tr>
<th>Media type</th>
<th>Message templates</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem</td>
<td><code>&lt;b&gt;Problem started&lt;/b&gt; at {EVENT.TIME} on {EVENT.DAY} for {EVENT.ENTITY} in {EVENT.LOCATION}&lt;/b&gt;</code></td>
<td>Edit</td>
</tr>
<tr>
<td>Problem recovery</td>
<td><code>&lt;b&gt;Problem has been resolved&lt;/b&gt; at {EVENT.RECOV.TIME} for {EVENT.ENTITY} in {EVENT.LOCATION}</code></td>
<td>Edit</td>
</tr>
<tr>
<td>Problem update</td>
<td><code>&lt;b&gt;{USER.FULLNAME} {EVENT.UPDATE.ACTION} problem&lt;/b&gt; for {EVENT.ENTITY} in {EVENT.LOCATION}</code></td>
<td>Edit</td>
</tr>
<tr>
<td>Service</td>
<td><code>&lt;b&gt;Service problem started&lt;/b&gt; at {EVENT.TIME} on {EVENT.DATE} for {EVENT.ENTITY}</code></td>
<td>Edit</td>
</tr>
<tr>
<td>Service recovery</td>
<td><code>&lt;Service &quot;{SERVICE.NAME}&quot; has been resolved&lt;/b&gt; at {EVENT.RECOV.TIME} for {EVENT.ENTITY}</code></td>
<td>Edit</td>
</tr>
<tr>
<td>Autoregistration</td>
<td><code>&lt;b&gt;Host name&lt;/b&gt;: {HOST.HOST}&lt;br&gt;{HOST.IP}&lt;br&gt;</code></td>
<td>Edit</td>
</tr>
</tbody>
</table>

To customize message templates:

- In the Message templates tab click on **Add**: a Message template popup window will open.
- Select required Message type and edit Subject and Message texts.
- Click on Add to save the message template
### Message template parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message type</td>
<td>Type of an event for which the default message should be used. Only one default message can be defined for each event type.</td>
</tr>
<tr>
<td>Subject</td>
<td>The default message. It is limited to certain amount of characters depending on the database type (see Sending messages for more information).</td>
</tr>
<tr>
<td>Message</td>
<td>The message may contain supported macros. In problem and problem update messages, expression macros are supported (for example, <code>{?avg(/host/key,1h})</code>).</td>
</tr>
</tbody>
</table>

To make changes to an existing message template: In the Actions column click on Edit to edit the template or click on Remove to delete the message template.

It is possible to define a custom message template for a specific action (see action operations for details). Custom messages defined in the action configuration will override default media type message template.

Defining message templates is mandatory for all media types, including webhooks or custom alert scripts that do not use default messages for notifications. For example, an action “Send message to Pushover webhook” will fail to send problem notifications, if the Problem message for the Pushover webhook is not defined.

The Options tab contains alert processing settings. The same set of options is configurable for each media type.

All media types are processed in parallel. While the maximum number of concurrent sessions is configurable per media type, the total number of alerter processes on the server can only be limited by the StartAlerters parameter. Alerts generated by one trigger are processed sequentially. So multiple notifications may be processed simultaneously only if they are generated by multiple triggers.
Parameter Description

Concurrent sessions Select the number of parallel alerter sessions for the media type:
- **One** - one session
- **Unlimited** - unlimited number of sessions
- **Custom** - select a custom number of sessions

Unlimited/high values mean more parallel sessions and increased capacity for sending notifications. Unlimited/high values should be used in large environments where lots of notifications may need to be sent simultaneously.

If more notifications need to be sent than there are concurrent sessions, the remaining notifications will be queued; they will not be lost.

Attempts Number of attempts for trying to send a notification. Up to 100 attempts can be specified; the default value is ‘3’. If ‘1’ is specified Zabbix will send the notification only once and will not retry if the sending fails.

Attempt interval Frequency of trying to resend a notification in case the sending failed, in seconds (0-3600). If ‘0’ is specified, Zabbix will retry immediately.

Time suffixes are supported, e.g. 5s, 3m, 1h.

Media type testing

It is possible to test if a configured media type works.

**E-mail**

For example, to test an e-mail media type:

- Locate the relevant e-mail in the list of media types
- Click on Test in the last column of the list (a testing window will open)
- Enter a Send to recipient address and with body and optional subject
- Send a test message by clicking on Test

Test success or failure message will be displayed in the same window:

![Test media type](image)

**Webhook**

To test a webhook media type:

- Locate the relevant webhook in the list of media types
- Click on Test in the last column of the list (a testing window will open)
• Edit the webhook parameter values, if needed
• Click on Test

By default, webhook tests are performed with parameters entered during configuration. However, it is possible to change attribute values for testing. Replacing or deleting values in the testing window affects the test procedure only, the actual webhook attribute values will remain unchanged.

To view media type test log entries without leaving the test window:
• Click on Open log (a new popup window will open).

If the webhook test is successful
• “Media type test successful.” message is displayed
• Server response appears in the gray Response field
• Response type (JSON or String) is specified below the Response field

If the webhook test fails
- “Media type test failed.” message is displayed, followed by additional failure details.

**User media**

To receive notifications of a media type, a medium (e-mail address/phone number/webhook user ID etc) for this media type must be defined in the user profile. For example, an action sending messages to user “Admin” using webhook “X” will always fail to send anything if the webhook “X” medium is not defined in the user profile.

To define user media:

- Go to your user profile, or go to Administration → Users and open the user properties form
- In the Media tab, click on [Add]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>The drop-down list contains names of all configured media types.</td>
</tr>
<tr>
<td>Send to</td>
<td>Provide required contact information where to send messages.</td>
</tr>
</tbody>
</table>

For an e-mail media type it is possible to add several addresses by clicking on [Add] below the address field. In this case, the notification will be sent to all e-mail addresses provided. It is also possible to specify recipient name in the Send to field of the e-mail recipient in a format 'Recipient name <address1@company.com>'. Note, that if a recipient name is provided, an e-mail address should be wrapped in angle brackets (<>). UTF-8 characters in the name are supported, quoted pairs and comments are not. For example: John Abercroft <manager@nycdcdatacenter.com> and manager@nycdcdatacenter.com are both valid formats. Incorrect examples: John Doe zabbix@company.com, %zabbix@<H(comment)>% and zabbix@company.com %.

When active

You can limit the time when messages are sent, for example, set the working days only (1-5,09:00-18:00). Note that this limit is based on the user time zone. If the user time zone is changed and is different from the system time zone this limit may need to be adjusted accordingly so as not to miss important messages. See the Time period specification page for description of the format.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use if severity</td>
<td>Mark the checkboxes of trigger severities that you want to receive notifications for. Note that the default severity ('Not classified') must be checked if you want to receive notifications for non-trigger events. After saving, the selected trigger severities will be displayed in the corresponding severity colors, while unselected ones will be grayed out.</td>
</tr>
<tr>
<td>Status</td>
<td>Status of the user media.</td>
</tr>
<tr>
<td></td>
<td><strong>Enabled</strong> - is in use.</td>
</tr>
<tr>
<td></td>
<td><strong>Disabled</strong> - is not being used.</td>
</tr>
</tbody>
</table>

**1 E-mail**

**Overview**

To configure e-mail as the delivery channel for messages, you need to configure e-mail as the media type and assign specific addresses to users.

Multiple notifications for single event will be grouped together on the same email thread.

**Configuration**

To configure e-mail as the media type:

- Go to Administration → Media types
- Click on Create media type (or click on E-mail in the list of pre-defined media types).

The **Media type** tab contains general media type attributes:
All mandatory input fields are marked with a red asterisk.

The following parameters are specific for the e-mail media type:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTP server</td>
<td>Set an SMTP server to handle outgoing messages.</td>
</tr>
<tr>
<td>SMTP server port</td>
<td>Set the SMTP server port to handle outgoing messages. This option is supported starting with Zabbix 3.0.</td>
</tr>
<tr>
<td>SMTP helo</td>
<td>Set a correct SMTP helo value, normally a domain name.</td>
</tr>
<tr>
<td>SMTP email</td>
<td>The address entered here will be used as the From address for the messages sent. Adding a sender display name (like “Zabbix_info” in Zabbix_info <a href="mailto:zabbix@example.com">zabbix@example.com</a> in the screenshot above) with the actual e-mail address is supported since Zabbix 2.2 version. There are some restrictions on display names in Zabbix emails in comparison to what is allowed by RFC 5322, as illustrated by examples: Valid examples: <a href="mailto:zabbix@company.com">zabbix@company.com</a> (only email address, no need to use angle brackets) Zabbix_info <a href="mailto:zabbix@company.com">zabbix@company.com</a> (display name and email address in angle brackets) ΣΩ-monitoring <a href="mailto:zabbix@company.com">zabbix@company.com</a> (UTF-8 characters in display name) Invalid examples: Zabbix HQ <a href="mailto:zabbix@company.com">zabbix@company.com</a> (display name present but no angle brackets around email address) “Zabbix@&lt;H(comment)&gt;” <a href="mailto:zabbix@company.com">zabbix@company.com</a> (although valid by RFC 5322, quoted pairs and comments are not supported in Zabbix emails)</td>
</tr>
</tbody>
</table>
Parameter | Description
--- | ---
Connection security | Select the level of connection security:  
None - do not use the CURLOPT_USE_SSL option  
STARTTLS - use the CURLOPT_USE_SSL option with CURLUSESSL_ALL value  
SSL/TLS - use of CURLOPT_USE_SSL is optional  
This option is supported starting with Zabbix 3.0.
SSL verify peer | Mark the checkbox to verify the SSL certificate of the SMTP server.  
The value of "SSLCAList" server configuration directive should be put into CURLOPT_CAPATH for certificate validation.  
This sets cURL option CURLOPT_SSL_VERIFYPEER.  
This option is supported starting with Zabbix 3.0.
SSL verify host | Mark the checkbox to verify that the Common Name field or the Subject Alternate Name field of the SMTP server certificate matches.  
This sets cURL option CURLOPT_SSL VERIFYHOST.  
This option is supported starting with Zabbix 3.0.
Authentication | Select the level of authentication:  
None - no cURL options are set  
Username and password - implies "AUTH=*" leaving the choice of authentication mechanism to cURL  
Normal password - CURLOPT LOGIN_OPTIONS is set to "AUTH=PLAIN"  
This option is supported starting with Zabbix 3.0.
Username | User name to use in authentication.  
This sets the value of CURLOPT_USERNAME.  
This option is supported starting with Zabbix 3.0.
Password | Password to use in authentication.  
This sets the value of CURLOPT_PASSWORD.  
This option is supported starting with Zabbix 3.0.
Message format | Select message format:  
HTML - send as HTML  
Plain text - send as plain text

To make SMTP authentication options available, Zabbix server should be compiled with the --with-libcurl compilation option with cURL 7.20.0 or higher.

See also common media type parameters for details on how to configure default messages and alert processing options.

User media

Once the e-mail media type is configured, go to the Administration → Users section and edit user profile to assign e-mail media to the user. Steps for setting up user media, being common for all media types, are described on the Media types page.

2 SMS

Overview

Zabbix supports the sending of SMS messages using a serial GSM modem connected to Zabbix server's serial port.

Make sure that:

• The speed of the serial device (normally /dev/ttyS0 under Linux) matches that of the GSM modem. Zabbix does not set the speed of the serial link. It uses default settings.
• The ‘zabbix’ user has read/write access to the serial device. Run the command ls -l /dev/ttyS0 to see current permissions of the serial device.
• The GSM modem has PIN entered and it preserves it after power reset. Alternatively you may disable PIN on the SIM card. PIN can be entered by issuing command AT+CPIN="NNNN" (NNNN is your PIN number, the quotes must be present) in a terminal software, such as Unix minicom or Windows HyperTerminal.

Zabbix has been tested with these GSM modems:

• Siemens MC35
• Teltonika ModemCOM/G10

To configure SMS as the delivery channel for messages, you also need to configure SMS as the media type and enter the respective phone numbers for the users.

Configuration
To configure SMS as the media type:

- Go to Administration → Media types
- Click on Create media type (or click on SMS in the list of pre-defined media types).

The following parameters are specific for the SMS media type:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSM modem</td>
<td>Set the serial device name of the GSM modem.</td>
</tr>
</tbody>
</table>

See common media type parameters for details on how to configure default messages and alert processing options. Note that parallel processing of sending SMS notifications is not possible.

User media

Once the SMS media type is configured, go to the Administration → Users section and edit user profile to assign SMS media to the user. Steps for setting up user media, being common for all media types, are described on the Media types page.

### 3 Custom alerts scripts

**Overview**

If you are not satisfied with existing media types for sending alerts there is an alternative way to do that. You can create a script that will handle the notification your way.

Alert scripts are executed on Zabbix server. These scripts are located in the directory defined in the server configuration file `AlertScriptsPath` variable.

Here is an example alert script:

```bash
#!/bin/bash

to=$1
subject=$2
body=$3

cat <<EOF | mail -s "$subject" "$to"
$body
EOF
```

Starting from version 3.4 Zabbix checks for the exit code of the executed commands and scripts. Any exit code which is different from 0 is considered as a command execution error. In such case Zabbix will try to repeat failed execution.

Environment variables are not preserved or created for the script, so they should be handled explicitly.

**Configuration**

To configure custom alerts scripts as the media type:

- Go to Administration → Media types
- Click on Create media type

The Media type tab contains general media type attributes:
All mandatory input fields are marked with a red asterisk.

The following parameters are specific for the script media type:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script name</td>
<td>Enter the name of the script.</td>
</tr>
</tbody>
</table>
| Script parameters | Add command-line parameters to the script.  
                      {ALERT.SENDTO}, {ALERT.SUBJECT} and {ALERT.MESSAGE} macros are supported in script parameters.  
Customizing script parameters is supported since Zabbix 3.0. |

See common media type parameters for details on how to configure default messages and alert processing options.

Even if an alertscript doesn’t use default messages, message templates for operation types used by this media type must still be defined, otherwise a notification will not be sent.

As parallel processing of media types is implemented since Zabbix 3.4.0, it is important to note that with more than one script media type configured, these scripts may be processed in parallel by alerter processes. The total number of alerter processes is limited by the StartAlerters parameter.

User media

Once the media type is configured, go to the Administration → Users section and edit user profile to assign media of this type to the user. Steps for setting up user media, being common for all media types, are described on the Media types page.

Note, that when defining a user media, a Send to field cannot be empty. If this field will not be used in an alertscript, enter any combination of supported characters to bypass validation requirements.
4 Webhook

Overview

The webhook media type is useful for making HTTP calls using custom JavaScript code for straightforward integration with external software such as helpdesk systems, chats, or messengers. You may choose to import an integration provided by Zabbix or create a custom integration from scratch.

Integrations

The following integrations are available allowing to use predefined webhook media types for pushing Zabbix notifications to:

- brevis.one
- Discord
- Express.ms messenger
- Github issues
- GLPi
- iLert
- iTop
- Jira
- Jira Service Desk
- ManageEngine ServiceDesk
- Mattermost
- Microsoft Teams
- Opsgenie
- OTRS
- Pagerduty
- Pushover
- Redmine
- Rocket.Chat
- ServiceNow
- SIGNL4
- Slack
- SolarWinds
- SysAid
- Telegram
- TOPdesk
- VictorOps
- Zammad
- Zendesk

In addition to the services listed here, Zabbix can be integrated with Spiceworks (no webhook is required). To convert Zabbix notifications into Spiceworks tickets, create an email media type and enter Spiceworks helpdesk email address (e.g. help@zabbix.on.spiceworks.com) in the profile settings of a designated Zabbix user.

Configuration

To start using a webhook integration:

1. Locate required .xml file in the templates/media directory of the downloaded Zabbix version or download it from Zabbix git repository
2. Import the file into your Zabbix installation. The webhook will appear in the list of media types.
3. Configure the webhook according to instructions in the Readme.md file (you may click on a webhook’s name above to quickly access Readme.md).

To create a custom webhook from scratch:

- Go to Administration → Media types
- Click on Create media type

The Media type tab contains various attributes specific for this media type:
All mandatory input fields are marked with a red asterisk.

The following parameters are specific for the webhook media type:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_source</td>
<td>{EVENT.SOURCE}</td>
</tr>
<tr>
<td>event_update_status</td>
<td>{EVENT.UPDATE.STATUS}</td>
</tr>
<tr>
<td>event_value</td>
<td>{EVENT.VALUE}</td>
</tr>
<tr>
<td>express_message</td>
<td>{ALERL.MESGAGE}</td>
</tr>
<tr>
<td>express_send_to</td>
<td>{ALERL.SENDTO}</td>
</tr>
<tr>
<td>express_tags</td>
<td>{EVENT.TAGSJSON}</td>
</tr>
<tr>
<td>express_token</td>
<td>&lt;PLACE BOT TOKEN&gt;</td>
</tr>
<tr>
<td>express_url</td>
<td>&lt;PLACE INSTANCE URL&gt;</td>
</tr>
</tbody>
</table>

- **Script**: `var Express = {...}

- **Timeout**: 30s

- **Process tags**: yes

- **Include event menu entry**: no

- **Menu entry name**: 

- **Menu entry URL**: 

- **Description**: 

- **Enabled**: yes
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>Specify the webhook variables as the attribute and value pairs. For preconfigured webhooks, a list of parameters varies, depending on the service. Check the webhook’s Readme.md file for parameter description. For new webhooks, several common variables are included by default (URL:&lt;empty&gt;, HTTPProxy:&lt;empty&gt;, To:(ALERT.SENDTO), Subject:(ALERT.SUBJECT), Message:%7BALERT.MESSAGE}). feel free to keep or remove them. All macros that are supported in problem notifications are supported in the parameters. If you specify an HTTP proxy, the field supports the same functionality as in the item configuration HTTP proxy field. The proxy string may be prefixed with [scheme]:// to specify which kind of proxy is used (e.g. https, socks4, socks5; see documentation).</td>
</tr>
<tr>
<td>Script</td>
<td>Enter JavaScript code in the block that appears when clicking in the parameter field (or on the view/edit button next to it). This code will perform the webhook operation. The script is a function code that accepts parameter - value pairs. The values should be converted into JSON objects using JSON.parse() method, for example: var params = JSON.parse(value); The code has access to all parameters, it may perform HTTP GET, POST, PUT and DELETE requests and has control over HTTP headers and request body. The script must contain a return operator, otherwise it will not be valid. It may return OK status along with an optional list of tags and tag values (see Process tags option) or an error string. Note, that the script is executed only after an alert is created. If the script is configured to return and process tags, these tags will not get resolved in {EVENT.TAGS} and {EVENT.RECOVERY.TAGS} macros in the initial problem message and recovery messages because the script has not had the time to run yet.</td>
</tr>
<tr>
<td>Timeout</td>
<td>JavaScript execution timeout (1-60s, default 30s). Time suffixes are supported, e.g. 30s, 1m.</td>
</tr>
<tr>
<td>Process tags</td>
<td>Mark the checkbox to process returned JSON property values as tags. These tags are added to the already existing (if any) problem event tags in Zabbix. If a webhook uses tags (the Process tags checkbox is marked), the webhook should always return a JSON object containing at least an empty object for tags:var result = {tags: {}}; Examples of tags that can be returned: Jira ID: PROD-1234, Responsible: John Smith, Processed:&lt;no value&gt;, etc.</td>
</tr>
<tr>
<td>Include event menu entry</td>
<td>Mark the checkbox to include an entry in the event menu linking to the created external ticket. If marked, the webhook should not be used to send notifications to different users (consider creating a dedicated user instead) or in several alert actions related to a single problem event.</td>
</tr>
<tr>
<td>Menu entry name</td>
<td>Specify the menu entry name. {EVENT.TAGS:&lt;tag name&gt;} macro is supported. This field is only mandatory if Include event menu entry is selected.</td>
</tr>
<tr>
<td>Menu entry URL</td>
<td>Specify the underlying URL of the menu entry. {EVENT.TAGS:&lt;tag name&gt;} macro is supported. This field is only mandatory if Include event menu entry is selected.</td>
</tr>
</tbody>
</table>

See common media type parameters for details on how to configure default messages and alert processing options.

Even if a webhook doesn’t use default messages, message templates for operation types used by this webhook must still be defined.

User media

Once the media type is configured, go to the Administration → Users section and assign the webhook media to an existing user or create a new user to represent the webhook. Steps for setting up user media for an existing user, being common for all media types, are described on the Media types page.

If a webhook uses tags to store ticket/message ID, avoid assigning the same webhook as a media to different users as doing so may cause webhook errors (applies to the majority of webhooks that utilize Include event menu entry option). In this case, the best practice is to create a dedicated user to represent the webhook:

1. After configuring the webhook media type, go to the Administration → Users section and create a dedicated Zabbix user to
represent the webhook - for example, with a username Slack for the Slack webhook. All settings, except media, can be left at their defaults as this user will not be logging into Zabbix.
2. In the user profile, go to a tab Media and add a webhook with the required contact information. If the webhook does not use a Send to field, enter any combination of supported characters to bypass validation requirements.
3. Grant this user at least read permissions to all hosts for which it should send the alerts.

When configuring alert action, add this user in the Send to users field in Operation details - this will tell Zabbix to use the webhook for notifications from this action.

Configuring alert actions

Actions determine which notifications should be sent via the webhook. Steps for configuring actions involving webhooks are the same as for all other media types with these exceptions:

- If a webhook uses tags to store ticket/message ID and to follow up with update/resolve operations, this webhook should not be used in several alert actions for a single problem event. If `{EVENT.TAGS.<name>}` already exists, and is updated in the webhook, then its resulting value is not defined. For such a case, a new tag name should be used in the webhook to store updated values. This applies to Jira, Jira Service Desk, Mattermost, Opsgenie, OTRS, Redmine, ServiceNow, Slack, Zammad, and Zendesk webhooks provided by Zabbix and to the majority of webhooks that utilize Include event menu entry option.
Using the webhook in several operations is allowed if those operations or escalation steps belong to the same action. It is also ok to use this webhook in different actions if the actions will not be applied to the same problem event due to different filter conditions.
- When using a webhook in actions for internal events: in the action operation configuration, check the Custom message checkbox and define the custom message, otherwise, a notification will not be sent.

Webhook script examples

Overview

Though Zabbix offers a large number of webhook integrations available out-of-the-box, you may want to create your own webhooks instead. This section provides examples of custom webhook scripts (used in the Script parameter). See webhook section for description of other webhook parameters.

Jira webhook (custom)
```javascript
try {
    Zabbix.log(4, '[ Jira webhook ] Started with params: ' + value);

    var result = {
        'tags': {
            'endpoint': 'jira'
        }
    },
    params = JSON.parse(value),
    req = new HttpReques{
    fields = {},
    resp;

    if (params.HTTPProxy) {
        req.setProxy(params.HTTPProxy);
    }
```
req.addHeader('Content-Type: application/json');
req.addHeader('Authorization: Basic ' + params.authentication);

fields.summary = params.summary;
fields.description = params.description;
fields.project = {key: params.project_key};
fields.issuetype = {id: params.issue_id};

resp = req.post('https://tsupport.zabbix.lan/rest/api/2/issue/',
    JSON.stringify({"fields": fields})
);

if (req.getStatus() != 201) {
    throw 'Response code: ' + req.getStatus();
}

resp = JSON.parse(resp);
result.tags.issue_id = resp.id;
result.tags.issue_key = resp.key;

return JSON.stringify(result);
}

catch (error) {
    Zabbix.log(4, '[ Jira webhook ] Issue creation failed json : ' + JSON.stringify({"fields": fields}));
    Zabbix.log(3, '[ Jira webhook ] issue creation failed : ' + error);
    throw 'Failed with error: ' + error;
}

Slack webhook (custom)
This webhook will forward notifications from Zabbix to a Slack channel.

<table>
<thead>
<tr>
<th>Media type</th>
<th>Message templates</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Name</td>
<td>Slack chat bot</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Webhook</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>URL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HTTPProxy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>channel</td>
<td>{ALERT SENDTO}</td>
</tr>
<tr>
<td></td>
<td>text</td>
<td>{ALERT SUBJECT}</td>
</tr>
<tr>
<td></td>
<td>username</td>
<td>bot</td>
</tr>
</tbody>
</table>

* Script

```javascript
try {
    var params = JSON.parse(value),
        req = new HttpRequest(),
        response;

    if (params.HTTPProxy) {
```
req.setProxy(params.HTTPProxy);
}

req.addHeader('Content-Type: application/x-www-form-urlencoded');

Zabbix.log(4, '[ Slack webhook ] Webhook request with value=' + value);

response = req.post(params.hook_url, 'payload=' + encodeURIComponent(value));
Zabbix.log(4, '[ Slack webhook ] Responded with code: ' + req.Status() + '. Response: ' + response);

try {
    response = JSON.parse(response);
} catch (error) {
    if (req.getStatus() < 200 || req.getStatus() >= 300) {
        throw 'Request failed with status code ' + req.getStatus();
    } else {
        throw 'Request success, but response parsing failed.';
    }
}

if (req.getStatus() !== 200 || !response.ok || response.ok === 'false') {
    throw response.error;
}

return 'OK';
}
catch (error) {
    Zabbix.log(3, '[ Jira webhook ] Sending failed. Error: ' + error);
    throw 'Failed with error: ' + error;
}

2 Actions

Overview

If you want some operations taking place as a result of events (for example, notifications sent), you need to configure actions.

Actions can be defined in response to events of all supported types:

- Trigger actions - for events when trigger status changes from OK to PROBLEM and back
- Service actions - for events when service status changes from OK to PROBLEM and back
- Discovery actions - for events when network discovery takes place
- Autoregistration actions - for events when new active agents auto-register (or host metadata changes for registered ones)
- Internal actions - for events when items become unsupported or triggers go into an unknown state

Configuring an action

To configure an action, do the following:

- Go to Configuration -> Actions and select the required action type from the submenu (you can switch to another type later, using the title dropdown)
- Click on Create action
- Name the action
- Choose conditions upon which operations are carried out
- Choose the operations to carry out

Note that service actions can be configured in the service action section.

General action attributes:
All mandatory input fields are marked with a red asterisk.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Unique action name.</td>
</tr>
</tbody>
</table>
| Type of calculation| Select the evaluation option for action conditions (with more than one condition):  
                     | **And** - all conditions must be met                                        |
|                    | **Or** - enough if one condition is met                                      |
|                    | **And/Or** - combination of the two: AND with different condition types and OR with the same condition type |
|                    | **Custom expression** - a user-defined calculation formula for evaluating action conditions. |
| Conditions         | List of action conditions.                                                  |
|                    | Click on Add to add a new condition.                                        |
| Enabled            | Mark the checkbox to enable the action. Otherwise, it will be disabled.     |

### 1 Conditions

**Overview**

It is possible to define that an action is executed only if the event matches a defined set of conditions. Conditions are set when configuring the action.

Condition matching is case-sensitive.

**Trigger actions**

The following conditions can be used in trigger-based actions:

<table>
<thead>
<tr>
<th>Condition type</th>
<th>Supported operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host group</td>
<td>equals</td>
<td>Specify host groups or host groups to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td><strong>equals</strong> - event belongs to this host group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>does not equal</strong> - event does not belong to this host group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifying a parent host group implicitly selects all nested host groups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To specify the parent group only, all nested groups have to be additionally set with the <strong>does not equal</strong> operator.</td>
</tr>
<tr>
<td>Template</td>
<td>equals</td>
<td>Specify templates or templates to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td><strong>equals</strong> - event belongs to a trigger inherited from this template.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>does not equal</strong> - event does not belong to a trigger inherited from this template.</td>
</tr>
<tr>
<td>Host</td>
<td>equals</td>
<td>Specify hosts or hosts to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td><strong>equals</strong> - event belongs to this host.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>does not equal</strong> - event does not belong to this host.</td>
</tr>
<tr>
<td>Condition type</td>
<td>Supported operators</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Tag name</td>
<td>equals</td>
<td>Specify event tag or event tag to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td>- event has this tag</td>
</tr>
<tr>
<td></td>
<td>contains</td>
<td>- event does not have this tag</td>
</tr>
<tr>
<td></td>
<td>does not contain</td>
<td>- event has a tag containing this string</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- event does not have a tag containing this string</td>
</tr>
<tr>
<td>Tag value</td>
<td>equals</td>
<td>Specify event tag and value combination or tag and value combination to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td>- event has this tag and value</td>
</tr>
<tr>
<td></td>
<td>contains</td>
<td>- event does not have this tag and value</td>
</tr>
<tr>
<td></td>
<td>does not contain</td>
<td>- event has a tag and value containing these strings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- event does not have a tag and value containing these strings</td>
</tr>
<tr>
<td>Trigger</td>
<td>equals</td>
<td>Specify triggers or triggers to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td>- event is generated by this trigger.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- event is generated by any other trigger, except this one.</td>
</tr>
<tr>
<td>Trigger name</td>
<td>contains</td>
<td>Specify a string in the trigger name or a string to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not contain</td>
<td>- event is generated by a trigger, containing this string in the name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- this string cannot be found in the trigger name.</td>
</tr>
<tr>
<td>Trigger severity</td>
<td>equals</td>
<td>Specify trigger severity.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td>- equal to trigger severity</td>
</tr>
<tr>
<td></td>
<td>is greater than or</td>
<td>- not equal to trigger severity</td>
</tr>
<tr>
<td></td>
<td>equals</td>
<td>- more or equal to trigger severity</td>
</tr>
<tr>
<td></td>
<td>is less than or</td>
<td>- less or equal to trigger severity</td>
</tr>
<tr>
<td></td>
<td>equals</td>
<td></td>
</tr>
<tr>
<td>Time period</td>
<td>in</td>
<td>Specify a time period or a time period to exclude.</td>
</tr>
<tr>
<td></td>
<td>not in</td>
<td>- event time is within the time period.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- event time is not within the time period.</td>
</tr>
<tr>
<td>Problem is suppressed</td>
<td>no</td>
<td>Specify if the problem is suppressed (not shown) because of host maintenance.</td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>- problem is not suppressed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- problem is suppressed.</td>
</tr>
</tbody>
</table>

**Discovery actions**

The following conditions can be used in discovery-based events:

<table>
<thead>
<tr>
<th>Condition type</th>
<th>Supported operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host IP</td>
<td>equals</td>
<td>Specify an IP address range or a range to exclude for a discovered host.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td>- host IP is in the range.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- host IP is not in the range.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It may have the following formats:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Single IP: 192.168.1.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Range of IP addresses: 192.168.1-1.254</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IP mask: 192.168.4.0/24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>List: 192.168.1-254, 192.168.2.1-100, 192.168.2.200, 192.168.4.0/24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Support for spaces in the list format is provided since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>Condition type</td>
<td>Supported operators</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Service type</td>
<td>equals</td>
<td>Specify a service type of a discovered service or a service type to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td><strong>equals</strong> - matches the discovered service. <strong>does not equal</strong> - does not match the discovered service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Available service types: SSH, LDAP, SMTP, FTP, HTTP, HTTPS (available since Zabbix 2.2 version), POP, NNTP, IMAP, TCP, Zabbix agent, SNMPv1 agent, SNMPv2 agent, SNMPv3 agent, ICMP ping, telnet (available since Zabbix 2.2 version).</td>
</tr>
<tr>
<td>Service port</td>
<td>equals</td>
<td>Specify a TCP port range of a discovered service or a range to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td><strong>equals</strong> - service port is in the range. <strong>does not equal</strong> - service port is not in the range.</td>
</tr>
<tr>
<td>Discovery rule</td>
<td>equals</td>
<td>Specify a discovery rule or a discovery rule to exclude. <strong>equals</strong> - using this discovery rule. <strong>does not equal</strong> - using any other discovery rule, except this one.</td>
</tr>
<tr>
<td>Discovery check</td>
<td>equals</td>
<td>Specify a discovery check or a discovery check to exclude. <strong>equals</strong> - using this discovery check. <strong>does not equal</strong> - using any other discovery check, except this one.</td>
</tr>
<tr>
<td>Discovery object</td>
<td>equals</td>
<td>Specify the discovered object. <strong>equals</strong> - equal to discovered object (a device or a service).</td>
</tr>
<tr>
<td>Discovery status</td>
<td>equals</td>
<td><strong>Up</strong> - matches ‘Host Up’ and ‘Service Up’ events. <strong>Down</strong> - matches ‘Host Down’ and ‘Service Down’ events <strong>Discovered</strong> - matches ‘Host Discovered’ and ‘Service Discovered’ events <strong>Lost</strong> - matches ‘Host Lost’ and ‘Service Lost’ events</td>
</tr>
<tr>
<td>Uptime/Downtime</td>
<td>is greater than or equals</td>
<td><strong>is greater than or equals</strong> - is more or equal to. Parameter is given in seconds. <strong>is less than or equals</strong> - is less or equal to. Parameter is given in seconds.</td>
</tr>
<tr>
<td>Received value</td>
<td>equals</td>
<td>Specify the value received from an agent (Zabbix, SNMP) check in a discovery rule. String comparison. If several Zabbix agent or SNMP checks are configured for a rule, received values for each of them are checked (each check generates a new event which is matched against all conditions). <strong>equals</strong> - equal to the value. <strong>does not equal</strong> - not equal to the value. <strong>is greater than or equals</strong> - more or equal to the value. <strong>is less than or equals</strong> - less or equal to the value. contains - contains the substring. Parameter is given as a string. <strong>does not contain</strong> - does not contain the substring. Parameter is given as a string.</td>
</tr>
<tr>
<td>Proxy</td>
<td>equals</td>
<td>Specify a proxy or a proxy to exclude. <strong>equals</strong> - using this proxy. <strong>does not equal</strong> - using any other proxy except this one.</td>
</tr>
</tbody>
</table>

Service checks in a discovery rule, which result in discovery events, do not take place simultaneously. Therefore, if **multiple** values are configured for Service type, Service port or Received value conditions in the action, they will be compared to one discovery event at a time, but **not** to several events simultaneously. As a result, actions with multiple values for the same check types may not be executed correctly.

**Autoregistration actions**

The following conditions can be used in actions based on active agent autoregistration:
<table>
<thead>
<tr>
<th>Condition type</th>
<th>Supported operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host metadata</td>
<td>contains</td>
<td>Specify host metadata or host metadata to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not contain</td>
<td></td>
</tr>
<tr>
<td></td>
<td>matches</td>
<td></td>
</tr>
<tr>
<td></td>
<td>does not match</td>
<td></td>
</tr>
<tr>
<td>Host name</td>
<td>contains</td>
<td>Specify a hostname or a hostname to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not contain</td>
<td></td>
</tr>
<tr>
<td></td>
<td>matches</td>
<td></td>
</tr>
<tr>
<td></td>
<td>does not match</td>
<td></td>
</tr>
<tr>
<td>Proxy</td>
<td>equals</td>
<td>Specify a proxy or a proxy to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td></td>
</tr>
</tbody>
</table>

**Internally event actions**

The following conditions can be set for actions based on internal events:

<table>
<thead>
<tr>
<th>Condition type</th>
<th>Supported operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event type</td>
<td>equals</td>
<td><strong>Item in &quot;not supported&quot; state</strong> - matches events where an item goes from a 'normal' to 'not supported' state</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Low-level discovery rule in &quot;not supported&quot; state</strong> - matches events where a low-level discovery rule goes from a 'normal' to 'not supported' state</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Trigger in &quot;unknown&quot; state</strong> - matches events where a trigger goes from a 'normal' to 'unknown' state</td>
</tr>
<tr>
<td>Host group</td>
<td>equals</td>
<td>Specify host groups or host groups to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td></td>
</tr>
<tr>
<td>Tag name</td>
<td>equals</td>
<td>Specify event tag or event tag to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>contains</td>
<td></td>
</tr>
<tr>
<td></td>
<td>does not contain</td>
<td></td>
</tr>
<tr>
<td>Tag value</td>
<td>equals</td>
<td>Specify event tag and value combination or tag and value combination to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>contains</td>
<td></td>
</tr>
<tr>
<td></td>
<td>does not contain</td>
<td></td>
</tr>
<tr>
<td>Template</td>
<td>equals</td>
<td>Specify templates or templates to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td></td>
</tr>
<tr>
<td>Host</td>
<td>equals</td>
<td>Specify hosts or hosts to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td></td>
</tr>
</tbody>
</table>

**Type of calculation**

The following options of calculating conditions are available:

- **And** - all conditions must be met
- **Or** - enough if one condition is met
- **And/Or** - combination of the two: AND with different condition types and OR with the same condition type, for example:
Host group equals Oracle servers
Host group equals MySQL servers
Trigger name contains ‘Database is down’
Trigger name contains ‘Database is unavailable’

is evaluated as

(Host group equals Oracle servers or Host group equals MySQL servers) and (Trigger name contains ‘Database is down’ or Trigger name contains ‘Database is unavailable’)

- **Custom expression** - a user-defined calculation formula for evaluating action conditions. It must include all conditions (represented as uppercase letters A, B, C, ...) and may include spaces, tabs, brackets ( ), and (case sensitive), or (case sensitive), not (case sensitive).

While the previous example with And/Or would be represented as (A or B) and (C or D), in a custom expression you may as well have multiple other ways of calculation:

(A and B) and (C or D)
(A and B) or (C and D)
((A or B) and C) or D
(not (A or B) and C) or not D

etc.

**Actions disabled due to deleted objects**

If a certain object (host, template, trigger, etc.) used in an action condition/operation is deleted, the condition/operation is removed and the action is disabled to avoid incorrect execution of the action. The action can be re-enabled by the user.

This behavior takes place when deleting:

- host groups (“host group” condition, “remote command” operation on a specific host group);
- hosts (“host” condition, “remote command” operation on a specific host);
- templates (“template” condition, “link to template” and “unlink from template” operations);
- triggers (“trigger” condition);
- discovery rules (when using “discovery rule” and “discovery check” conditions).

Note: If a remote command has many target hosts, and we delete one of them, only this host will be removed from the target list, the operation itself will remain. But, if it’s the only host, the operation will be removed, too. The same goes for “link to template” and “unlink from template” operations.

Actions are not disabled when deleting a user or user group used in a “send message” operation.

### 2 Operations

**Overview**

You can define the following operations for all events:

- send a message
- execute a remote command

Zabbix server does not create alerts if access to the host is explicitly "denied" for the user defined as action operation recipient or if the user has no rights defined to the host at all.

For discovery and autoregistration events, there are additional operations available:

- add host
- remove host
- enable host
- disable host
- add to host group
- remove from host group
- link to template
- unlink from template
- set host inventory mode

**Configuring an operation**

To configure an operation, go to the Operations tab in **action configuration**.
**General operation attributes:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default operation step duration</td>
<td>Duration of one operation step by default (60 seconds to 1 week). For example, an hour-long step duration means that if an operation is carried out, an hour will pass before the next step. Time suffixes are supported, e.g. 60s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0.</td>
</tr>
<tr>
<td>Operations</td>
<td>Action operations (if any) are displayed, with these details: Steps - escalation step(s) to which the operation is assigned. Details - type of operation and its recipient/target. The operation list also displays the media type (e-mail, SMS or script) used as well as the name and surname (in parentheses after the username) of a notification recipient. Start in - how long after an event the operation is performed. Duration (sec) - step duration is displayed. Default is displayed if the step uses default duration, and a time is displayed if custom duration is used.</td>
</tr>
<tr>
<td>Recovery operations</td>
<td>Action operations (if any) are displayed, with these details: Details - type of operation and its recipient/target. The operation list also displays the media type (e-mail, SMS or script) used as well as the name and surname (in parentheses after the username) of a notification recipient.</td>
</tr>
<tr>
<td>Update operations</td>
<td>Action operations (if any) are displayed, with these details: Details - type of operation and its recipient/target. The operation list also displays the media type (e-mail, SMS or script) used as well as the name and surname (in parentheses after the username) of a notification recipient. Action - links for editing and removing an operation are displayed.</td>
</tr>
</tbody>
</table>
Parameter Description

Pause operations for suppressed problems
Mark this checkbox to delay the start of operations for the duration of a maintenance period. When operations are started, after the maintenance, all operations are performed including those for the events during the maintenance. Note that this setting affects only problem escalations; recovery and update operations will not be affected. If you unmark this checkbox, operations will be executed without delay even during a maintenance period. This option is not available for Service actions.

Notify about canceled escalations
Unmark this checkbox to disable notifications about canceled escalations (when host, item, trigger or action is disabled).

All mandatory input fields are marked with a red asterisk.

To configure details of a new operation, click on **Add** in the Operations block. To edit an existing operation, click on **Edit** next to the operation. A popup window will open where you can edit the operation step details.

Operation details

![Operation details](image-url)
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation</td>
<td>Select the operation:</td>
</tr>
<tr>
<td></td>
<td><strong>Send message</strong> - send message to user</td>
</tr>
<tr>
<td></td>
<td><code>&lt;remote command name&gt;</code> - execute a remote command. Commands are available for execution if previously defined in global scripts with Action operation selected as its scope. More operations are available for discovery and autoregistration based events (see above).</td>
</tr>
<tr>
<td>Steps</td>
<td></td>
</tr>
<tr>
<td>Step duration</td>
<td></td>
</tr>
<tr>
<td>Operation type:</td>
<td></td>
</tr>
<tr>
<td>Send message</td>
<td>Click on Add to select user groups to send the message to. The user group must have at least &quot;read&quot; permissions to the host in order to be notified.</td>
</tr>
<tr>
<td>Send to user groups</td>
<td>Click on Add to select user groups to send the message to. The user group must have at least &quot;read&quot; permissions to the host in order to be notified.</td>
</tr>
<tr>
<td>Send to users</td>
<td>Click on Add to select users to send the message to. The user must have at least &quot;read&quot; permissions to the host in order to be notified.</td>
</tr>
<tr>
<td>Send only to</td>
<td>Send message to all defined media types or a selected one only.</td>
</tr>
<tr>
<td>Custom message</td>
<td>If selected, the custom message can be configured.</td>
</tr>
<tr>
<td>Subject</td>
<td>For notifications about internal events via webhooks, custom message is mandatory.</td>
</tr>
<tr>
<td>Message</td>
<td>Subject of the custom message. The subject may contain macros. It is limited to 255 characters.</td>
</tr>
<tr>
<td>Operation type:</td>
<td>The custom message. The message may contain macros. It is limited to certain amount of characters depending on the type of database (see Sending message for more information).</td>
</tr>
<tr>
<td>Remote command</td>
<td>Select targets to execute the command on:</td>
</tr>
<tr>
<td>Target list</td>
<td><strong>Current host</strong> - command is executed on the host of the trigger that caused the problem event. This option will not work if there are multiple hosts in the trigger.</td>
</tr>
<tr>
<td></td>
<td><strong>Host</strong> - select host(s) to execute the command on.</td>
</tr>
<tr>
<td></td>
<td><strong>Host group</strong> - select host group(s) to execute the command on. Specifying a parent host group implicitly selects all nested host groups. Thus the remote command will also be executed on hosts from nested groups.</td>
</tr>
<tr>
<td></td>
<td>A command on a host is executed only once, even if the host matches more than once (e.g. from several host groups; individually and from a host group). The target list is meaningless if a custom script is executed on Zabbix server. Selecting more targets in this case only results in the script being executed on the server more times. Note that for global scripts, the target selection also depends on the Host group setting in global script configuration.</td>
</tr>
<tr>
<td></td>
<td>Target list option is not available for Service actions because in this case remote commands are always executed on Zabbix server.</td>
</tr>
<tr>
<td>Conditions</td>
<td>Condition for performing the operation:</td>
</tr>
<tr>
<td></td>
<td><strong>Not ack</strong> - only when the event is unacknowledged</td>
</tr>
<tr>
<td></td>
<td><strong>Ack</strong> - only when the event is acknowledged.</td>
</tr>
<tr>
<td></td>
<td>Conditions option is not available for Service actions.</td>
</tr>
</tbody>
</table>

When done, click on Add to add the operation to the list of Operations.

1 Sending message

Overview

Sending a message is one of the best ways of notifying people about a problem. That is why it is one of the primary actions offered by Zabbix.
Configuration

To be able to send and receive notifications from Zabbix you have to:

- define the media to send a message to

The default trigger severity (‘Not classified’) must be checked in user media configuration if you want to receive notifications for non-trigger events such as discovery, active agent autoregistration or internal events.

- configure an action operation that sends a message to one of the defined media

Zabbix sends notifications only to those users that have at least ‘read’ permissions to the host that generated the event. At least one host of a trigger expression must be accessible.

You can configure custom scenarios for sending messages using escalations.

To successfully receive and read e-mails from Zabbix, e-mail servers/clients must support standard ‘SMTP/MIME e-mail’ format since Zabbix sends UTF-8 data (If the subject contains ASCII characters only, it is not UTF-8 encoded.). The subject and the body of the message are base64-encoded to follow ‘SMTP/MIME e-mail’ format standard.

Message limit after all macros expansion is the same as message limit for Remote commands.

Tracking messages

You can view the status of messages sent in Monitoring → Problems.

In the Actions column you can see summarized information about actions taken. In there green numbers represent messages sent, red ones - failed messages. In progress indicates that an action is initiated. Failed informs that no action has executed successfully.

If you click on the event time to view event details, you will also see the Message actions block containing details of messages sent (or not sent) due to the event.

In Reports → Action log you will see details of all actions taken for those events that have an action configured.

2 Remote commands

Overview

With remote commands you can define that a certain pre-defined command is automatically executed on the monitored host upon some condition.

Thus remote commands are a powerful mechanism for smart pro-active monitoring.

In the most obvious uses of the feature you can try to:

- Automatically restart some application (web server, middleware, CRM) if it does not respond
- Use IPMI ‘reboot’ command to reboot some remote server if it does not answer requests
- Automatically free disk space (removing older files, cleaning /tmp) if running out of disk space
- Migrate a VM from one physical box to another depending on the CPU load
- Add new nodes to a cloud environment upon insufficient CPU (disk, memory, whatever) resources

Configuring an action for remote commands is similar to that for sending a message, the only difference being that Zabbix will execute a command instead of sending a message.

Remote commands can be executed by Zabbix server, proxy or agent. Remote commands on Zabbix agent can be executed directly by Zabbix server or through Zabbix proxy. Both on Zabbix agent and Zabbix proxy remote commands are disabled by default. They can be enabled by:

- adding an AllowKey=system.run[*] parameter in agent configuration;
- setting the EnableRemoteCommands parameter to ‘1’ in proxy configuration.

Remote commands executed by Zabbix server are run as described in Command execution including exit code checking.

Remote commands are executed even if the target host is in maintenance.

Remote command limit

Remote command limit after resolving all macros depends on the type of database and character set (non-ASCII characters require more than one byte to be stored):
The following tutorial provides step-by-step instructions on how to set up remote commands.

Configuration

Those remote commands that are executed on Zabbix agent (custom scripts) must be first enabled in the agent configuration. Make sure that the AllowKey=system.run[<command>,*] parameter is added for each allowed command in agent configuration to allow specific command with nowait mode. Restart agent daemon if changing this parameter.

Remote commands do not work with active Zabbix agents.

Then, when configuring a new action in Configuration → Actions:

- Define the appropriate conditions. In this example, set that the action is activated upon any disaster problems with one of Apache applications:

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type of calculation</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serious problem with Apache</td>
<td>And</td>
<td>A and B and C</td>
</tr>
<tr>
<td>Label</td>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>A</td>
<td>Problem is not suppressed</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Application contains Apache</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Trigger severity is greater than or equals Disaster</td>
<td></td>
</tr>
</tbody>
</table>
```

- In the Operations tab, click on Add in the Operations/Recovery operations/Update operations block
- From the Operation dropdown field select one of the predefined scripts

```
Operation details

Operation: Restart webserver
Steps:
- Send message
- Restart webserver
```

- Select the target list for the script

Predefined scripts

All scripts (webhook, script, SSH, Telnet, IPMI) that are available for action operations are defined in global scripts.

For example:

```
sudo /etc/init.d/apache restart
```

In this case, Zabbix will try to restart an Apache process. With this command, make sure that the command is executed on Zabbix agent (click the Zabbix agent button against Execute on).
Note the use of **sudo** - Zabbix user does not have permissions to restart system services by default. See below for hints on how to configure **sudo**.

Zabbix agent should run on the remote host and accept incoming connections. Zabbix agent executes commands in background. Remote commands on Zabbix agent are executed without timeout by the system.run,[nowait] key and are not checked for execution results. On Zabbix server and Zabbix proxy, remote commands are executed with timeout as set in the TrapperTimeout parameter of `zabbix_server.conf` or `zabbix_proxy.conf` file and are checked for execution results.

Access permissions

Make sure that the ‘zabbix’ user has execute permissions for configured commands. One may be interested in using **sudo** to give access to privileged commands. To configure access, execute as root:

```bash
# visudo
```

Example lines that could be used in sudoers file:

```bash
# allows 'zabbix' user to run all commands without password.
zabbix ALL=NOPASSWD: ALL

# allows 'zabbix' user to restart apache without password.
zabbix ALL=NOPASSWD: /etc/init.d/apache restart
```

On some systems sudoers file will prevent non-local users from executing commands. To change this, comment out `requiretty` option in `/etc/sudoers`.

Remote commands with multiple interfaces

If the target system has multiple interfaces of the selected type (Zabbix agent or IPMI), remote commands will be executed on the default interface.

It is possible to execute remote commands via SSH and Telnet using another interface than the Zabbix agent one. The available interface to use is selected in the following order:

- Zabbix agent default interface
- SNMP default interface
- JMX default interface
- IPMI default interface

IPMI remote commands

For IPMI remote commands the following syntax should be used:

```bash
<command> [<value>]
```

where

- `<command>` - one of IPMI commands without spaces
- `<value>` - 'on', 'off' or any unsigned integer. `<value>` is an optional parameter.

Examples

Examples of **global scripts** that may be used as remote commands in action operations.

Example 1

Restart of Windows on certain condition.

In order to automatically restart Windows upon a problem detected by Zabbix, define the following script:

<table>
<thead>
<tr>
<th>Script parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope</td>
<td>'Action operation'</td>
</tr>
<tr>
<td>Type</td>
<td>'Script'</td>
</tr>
<tr>
<td>Command</td>
<td><code>c:\windows\system32\shutdown.exe -r -f</code></td>
</tr>
</tbody>
</table>

Example 2

Restart the host by using IPMI control.

<table>
<thead>
<tr>
<th>Script parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope</td>
<td>'Action operation'</td>
</tr>
</tbody>
</table>
Example 3

Power off the host by using IPMI control.

3 Additional operations

Overview

In this section you may find some details of additional operations for discovery/autoregistration events.

Adding host

Hosts are added during the discovery process, as soon as a host is discovered, rather than at the end of the discovery process. As network discovery can take some time due to many unavailable hosts/services having patience and using reasonable IP ranges is advisable.

When adding a host, its name is decided by the standard gethostbyname function. If the host can be resolved, resolved name is used. If not, the IP address is used. Besides, if IPv6 address must be used for a host name, then all “:” (colons) are replaced by “_” (underscores), since colons are not allowed in host names.

If performing discovery by a proxy, currently hostname lookup still takes place on Zabbix server.

If a host already exists in Zabbix configuration with the same name as a newly discovered one, versions of Zabbix prior to 1.8 would add another host with the same name. Zabbix 1.8.1 and later adds _N to the hostname, where N is increasing number, starting with 2.

4 Using macros in messages

Overview

In message subjects and message text you can use macros for more efficient problem reporting.

A full list of macros supported by Zabbix is available.

Examples

Examples here illustrate how you can use macros in messages.

Example 1

Message subject:
Problem: {TRIGGER.NAME}

When you receive the message, the message subject will be replaced by something like:

Problem: Processor load is too high on Zabbix server

Example 2

Message:
Processor load is: last(/zabbix.zabbix.com/system.cpu.load[,avg1])

When you receive the message, the message will be replaced by something like:

Processor load is: 1.45
Example 3
Message:
Latest value: last(/HOST.HOST/ITEM.KEY)
MAX for 15 minutes: max(/HOST.HOST/ITEM.KEY,15m)
MIN for 15 minutes: min(/HOST.HOST/ITEM.KEY,15m)

When you receive the message, the message will be replaced by something like:

Latest value: 1.45
MAX for 15 minutes: 2.33
MIN for 15 minutes: 1.01

Example 4
Message:
http://<server_ip_or_name>/zabbix/tr_events.php?triggerid={TRIGGER.ID}&eventid={EVENT.ID}

When you receive the message, it will contain a link to the Event details page, which provides information about the event, its trigger, and a list of latest events generated by the same trigger.

Example 5
Informing about values from several hosts in a trigger expression.

Message:
Problem name: {TRIGGER.NAME}
Trigger expression: {TRIGGER.EXPRESSION}
1. Item value on {HOST.NAME1}: {ITEM.VALUE1} ({ITEM.NAME1})
2. Item value on {HOST.NAME2}: {ITEM.VALUE2} ({ITEM.NAME2})

When you receive the message, the message will be replaced by something like:

Problem name: Processor load is too high on a local host
Trigger expression: last(/Myhost/system.cpu.load[percpu,avg1])>5 or last(/Myotherhost/system.cpu.load[percpu,avg1])>5
1. Item value on Myhost: 0.83 (Processor load (1 min average per core))
2. Item value on Myotherhost: 5.125 (Processor load (1 min average per core))

Example 6
Receiving details of both the problem event and recovery event in a recovery message:

Message:
Problem:
Event ID: {EVENT.ID}
Event value: {EVENT.VALUE}
Event status: {EVENT.STATUS}
Event time: {EVENT.TIME}
Event date: {EVENT.DATE}
Event age: {EVENT.AGE}
Event acknowledgment: {EVENT.ACK.STATUS}
Event update history: {EVENT.UPDATE.HISTORY}
Recovery:
Event ID: {EVENT.RECOVERY.ID}
Event value: {EVENT.RECOVERY.VALUE}
Event status: {EVENT.RECOVERY.STATUS}
Event time: {EVENT.RECOVERY.TIME}
Event date: {EVENT.RECOVERY.DATE}
Operational data: {EVENT.OPDATA}

When you receive the message, the macros will be replaced by something like:

Problem:
Recovery:

Event ID: 21896
Event value: 0
Event status: OK
Event time: 13:10:07
Event date: 2018.01.02
Operational data: Current value is 0.83

Separate notification macros for the original problem event and recovery event are supported since Zabbix 2.2.0.

3 Recovery operations

Overview

Recovery operations allow you to be notified when problems are resolved.
Both messages and remote commands are supported in recovery operations. While several operations can be added, escalation is not supported - all operations are assigned to a single step and therefore will be performed simultaneously.

Use cases

Some use cases for recovery operations are as follows:

1. Notify all users that were notified on the problem
   * Select 'Send recovery message' as operation type
   * Add operation types for sending a message and executing a command
   * Open a ticket in external helpdesk/ticketing system and close it when the problem is resolved
   * Create an external script that communicates with the helpdesk system
   * Create an action having operation that executes this script and thus opens a ticket
   * Have a recovery operation that executes this script with other parameters and closes the ticket
   * Use the \{EVENT.ID\} macro to reference the original problem

Configuring a recovery operation

To configure a recovery operation, go to the Operations tab in action configuration.
To configure details of a new recovery operation, click on **Add** in the Recovery operations block. To edit an existing operation, click on **Edit** next to the operation. A popup window will open where you can edit the operation step details.

**Recovery operation details**

Three operation types are available for recovery events: - **Send message** - send recovery message to specified user - **Notify all**
involved - send recovery message to all users who were notified on the problem event - `<remote command name>` - execute a remote command. Commands are available for execution if previously defined in global scripts with Action operation selected as its scope.

Parameters for each operation type are described below. All mandatory input fields are marked with a red asterisk. When done, click on Add to add operation to the list of Recovery operations.

:::note classic Note that if the same recipient is defined in several operation types without specified Custom message, duplicate notifications are not sent. :::

Operation type: send message

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send to user groups</td>
<td>Click on Add to select user groups to send the recovery message to.</td>
</tr>
<tr>
<td></td>
<td>The user group must have at least &quot;read&quot; permissions to the host in order to be notified.</td>
</tr>
<tr>
<td>Send to users</td>
<td>Click on Add to select users to send the recovery message to.</td>
</tr>
<tr>
<td></td>
<td>The user must have at least &quot;read&quot; permissions to the host in order to be notified.</td>
</tr>
<tr>
<td>Send only to</td>
<td>Send default recovery message to all defined media types or a selected one only.</td>
</tr>
<tr>
<td>Custom message</td>
<td>If selected, a custom message can be defined.</td>
</tr>
<tr>
<td>Subject</td>
<td>Subject of the custom message. The subject may contain macros.</td>
</tr>
<tr>
<td>Message</td>
<td>The custom message. The message may contain macros.</td>
</tr>
</tbody>
</table>

Operation type: remote command

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target list</td>
<td>Select targets to execute the command on:</td>
</tr>
<tr>
<td>Current host</td>
<td>- command is executed on the host of the trigger that caused the problem event.</td>
</tr>
<tr>
<td></td>
<td>This option will not work if there are multiple hosts in the trigger.</td>
</tr>
<tr>
<td>Host</td>
<td>- select host(s) to execute the command on.</td>
</tr>
</tbody>
</table>
| Host group     | - select host group(s) to execute the command on. Specifying a parent host group |}
|                | implicitly selects all nested host groups. Thus the remote command will also be executed on hosts from nested groups. |
|                | A command on a host is executed only once, even if the host matches more than once (e.g. from several host groups; individually and from a host group). |
|                | The target list is meaningless if the command is executed on Zabbix server. Selecting more targets in this case only results in the command being executed on the server more times. |
|                | Note that for global scripts, the target selection also depends on the Host group setting in global script configuration. |

Operation type: notify all involved

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom message</td>
<td>If selected, a custom message can be defined.</td>
</tr>
<tr>
<td>Subject</td>
<td>Subject of the custom message. The subject may contain macros.</td>
</tr>
<tr>
<td>Message</td>
<td>The custom message. The message may contain macros.</td>
</tr>
</tbody>
</table>

4 Update operations

Overview

Update operations are available in actions with the following event sources:

- Triggers - when problems are updated by other users, i.e. commented upon, acknowledged, severity has been changed, closed (manually);
- Services - when the severity of a service has changed but the service is still not recovered.

Both messages and remote commands are supported in update operations. While several operations can be added, escalation is not supported - all operations are assigned to a single step and therefore will be performed simultaneously.

Configuring an update operation

To configure an update operation go to the Operations tab in action configuration.
To configure details of a new update operation, click on `Add` in the Update operations block. To edit an existing operation, click on `Edit` next to the operation. A popup window will open where you can edit the operation step details.

Update operations offer the same set of parameters as Recovery operations.

5 Escalations
Overview

With escalations you can create custom scenarios for sending notifications or executing remote commands.

In practical terms it means that:

- Users can be informed about new problems immediately
- Notifications can be repeated until the problem is resolved
- Sending a notification can be delayed
- Notifications can be escalated to another “higher” user group
- Remote commands can be executed immediately or when a problem is not resolved for a lengthy period

Actions are escalated based on the escalation step. Each step has a duration in time.

You can define both the default duration and a custom duration of an individual step. The minimum duration of one escalation step is 60 seconds.

You can start actions, such as sending notifications or executing commands, from any step. Step one is for immediate actions. If you want to delay an action, you can assign it to a later step. For each step, several actions can be defined.

The number of escalation steps is not limited.

Escalations are defined when configuring an operation. Escalations are supported for problem operations only, not recovery.

Miscellaneous aspects of escalation behavior

Let’s consider what happens in different circumstances if an action contains several escalation steps.

<table>
<thead>
<tr>
<th>Situation</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>The host in question goes into maintenance after the initial problem notification is sent</td>
<td>Depending on the Pause operations for suppressed problems setting in action configuration, all remaining escalation steps are executed either with a delay caused by the maintenance period or without delay. A maintenance period does not cancel operations.</td>
</tr>
<tr>
<td>The time period defined in the <strong>Time period</strong> action condition ends after the initial notification is sent</td>
<td>All remaining escalation steps are executed. The Time period condition cannot stop operations; it has effect with regard to when actions are started/not started, not operations.</td>
</tr>
<tr>
<td>A problem starts during maintenance and continues (is not resolved) after maintenance ends</td>
<td>Depending on the Pause operations for suppressed problems setting in action configuration, all escalation steps are executed either from the moment maintenance ends or immediately.</td>
</tr>
<tr>
<td>A problem starts during a no-data maintenance and continues (is not resolved) after maintenance ends</td>
<td>It must wait for the trigger to fire, before all escalation steps are executed. The execution of each new escalation supersedes the previous escalation, but for at least one escalation step that is always executed on the previous escalation. This behavior is relevant in actions upon events that are created with EVERY problem evaluation of the trigger.</td>
</tr>
<tr>
<td>Different escalations follow in close succession and overlap</td>
<td>The message in progress is sent and then one more message on the escalation is sent. The follow-up message will have the cancellation text at the beginning of the message body (NOTE: Escalation canceled) naming the reason (for example, NOTE: Escalation canceled: action <code>&lt;Action name&gt;</code> disabled). This way the recipient is informed that the escalation is canceled and no more steps will be executed. This message is sent to all who received the notifications before. The reason of cancellation is also logged to the server log file (starting from Debug Level 3=Warning).</td>
</tr>
<tr>
<td>During an escalation in progress (like a message being sent), based on any type of event:&lt;br&gt;- the action is disabled&lt;br&gt;- the host or item is disabled&lt;br&gt;- the trigger is disabled&lt;br&gt;- the trigger is based on internal event about triggers: &lt;br&gt;- the trigger is disabled&lt;br&gt;- the item is disabled&lt;br&gt;- the host is disabled</td>
<td>Note that the Escalation canceled message is also sent if operations are finished, but recovery operations are configured and are not executed yet.</td>
</tr>
<tr>
<td>Situation</td>
<td>Behavior</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>During an escalation in progress (like a message being sent) the action is deleted</td>
<td>No more messages are sent. The information is logged to the server log file (starting from Debug Level 3=Warning), for example: escalation canceled: action id:334 deleted</td>
</tr>
</tbody>
</table>

Escalation examples

Example 1

Sending a repeated notification once every 30 minutes (5 times in total) to a ‘MySQL Administrators’ group. To configure:

- In Operations tab, set the Default operation step duration to ‘30m’ (30 minutes)
- Set the escalation steps to be From ‘1’ To ‘5’
- Select the ‘MySQL Administrators’ group as recipients of the message

Notifications will be sent at 0:00, 0:30, 1:00, 1:30, 2:00 hours after the problem starts (unless, of course, the problem is resolved sooner).

If the problem is resolved and a recovery message is configured, it will be sent to those who received at least one problem message within this escalation scenario.

If the trigger that generated an active escalation is disabled, Zabbix sends an informative message about it to all those that have already received notifications.

Example 2

Sending a delayed notification about a long-standing problem. To configure:

- In Operations tab, set the Default operation step duration to ‘10h’ (10 hours)
- Set the escalation steps to be From ‘2’ To ‘2’

A notification will only be sent at Step 2 of the escalation scenario, or 10 hours after the problem starts.

You can customize the message text to something like ‘The problem is more than 10 hours old’.

Example 3

Escalating the problem to the Boss.

In the first example above we configured periodical sending of messages to MySQL administrators. In this case, the administrators will get four messages before the problem will be escalated to the Database manager. Note that the manager will get a message only in case the problem is not acknowledged yet, supposedly no one is working on it.
Details of Operation 2:

Note the use of \(\text{ESC.HISTORY}\) macro in the customized message. The macro will contain information about all previously executed steps on this escalation, such as notifications sent and commands executed.

Example 4

A more complex scenario. After multiple messages to MySQL administrators and escalation to the manager, Zabbix will try to restart the MySQL database. It will happen if the problem exists for 2:30 hours and it hasn’t been acknowledged.

If the problem still exists, after another 30 minutes Zabbix will send a message to all guest users.
If this does not help, after another hour Zabbix will reboot server with the MySQL database (second remote command) using IPMI commands.

Example 5

An escalation with several operations assigned to one step and custom intervals used. The default operation step duration is 30 minutes.

Notifications will be sent as follows:

- to MySQL administrators at 0:00, 0:30, 1:00, 1:30 after the problem starts
- to Database manager at 2:00 and 2:10 (and not at 3:00; seeing that steps 5 and 6 overlap with the next operation, the shorter custom step duration of 10 minutes in the next operation overrides the longer step duration of 1 hour tried to set here)
- to Zabbix administrators at 2:00, 2:10, 2:20 after the problem starts (the custom step duration of 10 minutes working)
- to guest users at 4:00 hours after the problem start (the default step duration of 30 minutes returning between steps 8 and 11)

3 Receiving notification on unsupported items

Overview

Receiving notifications on unsupported items is supported since Zabbix 2.2.

It is part of the concept of internal events in Zabbix, allowing users to be notified on these occasions. Internal events reflect a change of state:

- when items go from ‘normal’ to ‘unsupported’ (and back)
- when triggers go from ‘normal’ to ‘unknown’ (and back)
- when low-level discovery rules go from ‘normal’ to ‘unsupported’ (and back)

This section presents a how-to for receiving notification when an item turns unsupported.

Configuration

Overall, the process of setting up the notification should feel familiar to those who have set up alerts in Zabbix before.
Step 1

Configure some media, such as e-mail, SMS, or script to use for the notifications. Refer to the corresponding sections of the manual to perform this task.

For notifying on internal events the default severity ('Not classified') is used, so leave it checked when configuring user media if you want to receive notifications for internal events.

Step 2

Go to Configuration → Actions and select internal actions from the third level menu (or page title dropdown).

Click on Create action to the right to open an action configuration form.

Step 3

In the Action tab enter a name for the action. Then click on Add in the condition block to add a new condition.

In the new condition popup window select Event type as the condition type and then select Item in "not supported" state as the event type value.

Don't forget to click on Add to actually list the condition in the Conditions block.

Step 4

In the Operations tab, click on Add in the Operations block and select some recipients of the message (user groups/users) and the media types (or 'All') to use for delivery.

Select Custom message checkbox if you wish to enter the custom subject/content of the problem message.
<table>
<thead>
<tr>
<th>Action</th>
<th>Operations 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Default operation step duration</td>
<td>1h</td>
</tr>
</tbody>
</table>

**Operations**

<table>
<thead>
<tr>
<th>Steps</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Send message to user groups: Zabbix administrators via all media</td>
</tr>
</tbody>
</table>

**Recovery operations**

<table>
<thead>
<tr>
<th>Details</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notify all involved</td>
<td>Add</td>
</tr>
</tbody>
</table>

---

**Operation details**

<table>
<thead>
<tr>
<th>Operation type</th>
<th>Send message</th>
</tr>
</thead>
</table>

**Steps**

| 1 - | 1 (0 - infinitely) |

**Step duration**

| 0 | (0 - use action default) |

* At least one user or user group must be selected.

**Send to user groups**

<table>
<thead>
<tr>
<th>User group</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zabbix administrators</td>
<td>Remove</td>
</tr>
</tbody>
</table>

**Send to users**

<table>
<thead>
<tr>
<th>User</th>
<th>Action</th>
</tr>
</thead>
</table>

**Send only to**

| - All - |

**Custom message**

| Yes |

**Subject**

{%ITEM.STATE%}: {%HOST.NAME%}:{ITEM.NAME%}

**Message**

Host: {HOST.NAME}
Item: {ITEM.NAME}
Key: {ITEM.KEY}
State: {ITEM.STATE}

Click on Add to actually list the operation in the Operations block.
If you wish to receive more than one notification, set the operation step duration (interval between messages sent) and add another step.

Step 5

The **Recovery operations** block allows to configure a recovery notification when an item goes back to the normal state. Click on Add in the Recovery operations block, select the operation type, the recipients of the message (user groups/users) and the media types (or 'All') to use for delivery.

Select Custom message checkbox if you wish to enter the custom subject/content of the problem message.

Click on Add in the Operation details popup window to actually list the operation in the Recovery operations block.

Step 6

When finished, click on the **Add** button at the bottom of the form.

And that’s it, you’re done! Now you can look forward to receiving your first notification from Zabbix if some item turns unsupported.

### 11 Macros

**Overview**

Zabbix supports a number of built-in macros which may be used in various situations. These macros are variables, identified by a specific syntax:

```
{MACRO}
```

Macros resolve to a specific value depending on the context.

Effective use of macros allows to save time and make Zabbix configuration more transparent.

In one of typical uses, a macro may be used in a template. Thus a trigger on a template may be named "Processor load is too high on **(HOST.NAME)**". When the template is applied to the host, such as Zabbix server, the name will resolve to "Processor load is too high on Zabbix server" when the trigger is displayed in the Monitoring section.

Macros may be used in item key parameters. A macro may be used for only a part of the parameter, for example `item.key[server_{HOST.HOST}_local]`. Double-quoting the parameter is not necessary as Zabbix will take care of any ambiguous special symbols, if present in the resolved macro.

There are other types of macros in Zabbix.

Zabbix supports the following macros:

- `{MACRO}` - built-in macro (see full list)
- `{macro}.<func>({parama})` - macro functions
Macro functions

Overview

Macro functions offer the ability to customize macro values. Sometimes a macro may resolve to a value that is not necessarily easy to work with. It may be long or contain a specific substring of interest that you would like to extract. This is where macro functions can be useful.

The syntax of a macro function is:

{<macro>.<func>(<params>)}

where:

- `<macro>` - the macro to customize (for example {ITEM.VALUE} or {#LLDMACRO})
- `<func>` - the function to apply
- `<params>` - a comma-delimited list of function parameters. Parameters must be quoted if they start with " " (space), " or contain }, , .

For example:

{{TIME}.fmttime(format,time_shift)}
{{ITEM.VALUE}.regsub(pattern, output)}
{{#LLDMACRO}.regsub(pattern, output)}

Supported macro functions

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>Description</th>
<th>Parameters</th>
<th>Supported for</th>
</tr>
</thead>
<tbody>
<tr>
<td>fmtnum</td>
<td>Number formatting to control the number of digits printed after the decimal point.</td>
<td>digits - the number of digits after decimal point. No trailing zeros will be produced.</td>
<td>{ITEM.VALUE} {ITEM.LASTVALUE}</td>
</tr>
<tr>
<td>fmttime</td>
<td></td>
<td></td>
<td>Expression macros</td>
</tr>
</tbody>
</table>
**FUNCTION**

| Time formatting. | format - mandatory format string, compatible with strftime function formatting | \{TIME\} |
| time_shift - the time shift applied to the time before formatting; should start with |-<N><time_unit> or +<N><time_unit>, where N - the number of time units to add or subtract; time_unit - h (hour), d (day), w (week), M (month) or y (year). Since Zabbix 5.4, time_shift parameter supports multi-step time operations and may include /<time_unit> for shifting to the beginning of the time unit (/d - midnight, /w - 1st day of the week (Monday), /M - 1st day of the month, etc.). Examples: -1w - exactly 7 days back; -1w/w - Monday of the previous week; -1w/w+1d - Tuesday of the previous week. Note, that time operations are calculated from left to right without priorities. For example, -1M/d+1h/w will be parsed as ((-1M/d)+1h)/w. |

**iregsub**
(<pattern>,<output>)

Substring extraction by a regular expression match (case insensitive).

pattern - the regular expression to match
output - the output options. \1 - \9 placeholders are supported to capture groups. \0 returns the matched text.

**regsub**
(<pattern>,<output>)

Substring extraction by a regular expression match (case sensitive).

pattern - the regular expression to match
output - the output options. \1 - \9 placeholders are supported to capture groups. \0 returns the matched text.

If a function is used in a supported location, but applied to a macro not supporting macro functions, then the macro evaluates to 'UNKNOWN'.

If pattern is not a correct regular expression then the macro evaluates to 'UNKNOWN' (excluding low-level discovery macros where the function will be ignored in that case and macro will remain unexpanded)

If a macro function is applied to the macro in locations not supporting macro functions then the function is ignored.

Examples

The ways in which macro functions can be used to customize macro values is illustrated in the following examples on received values:
### Received value

<table>
<thead>
<tr>
<th>Value</th>
<th>Macro</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>24.3413523</td>
<td><code>{{ITEM.VALUE}.fmtnum(2)}</code></td>
<td>24.34</td>
</tr>
<tr>
<td>24.3413523</td>
<td><code>{{ITEM.VALUE}.fmtnum(0)}</code></td>
<td>24</td>
</tr>
<tr>
<td>12:36:01</td>
<td><code>{{TIME}.fmttime(%B)}</code></td>
<td>October</td>
</tr>
<tr>
<td>12:36:01</td>
<td><code>{{TIME}.fmttime(%d %B,-1M/M)}</code></td>
<td>1 September</td>
</tr>
<tr>
<td>123</td>
<td>Log line</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td><code>{{ITEM.VALUE}.regsub(&quot;([0-9]+)\&quot;,&quot;Problem&quot;)}</code></td>
<td></td>
</tr>
<tr>
<td>123</td>
<td><code>{{ITEM.VALUE}.regsub(&quot;([0-9]+)\&quot;,&quot;Problem ID: \1&quot;)}</code></td>
<td></td>
</tr>
<tr>
<td>MySQL crashed erro</td>
<td><code>{{ITEM.VALUE}.regsub(&quot;(\w+).*?\([0-9]+\)\&quot;,&quot;ProblemID:\1\&quot;)}</code></td>
<td></td>
</tr>
<tr>
<td>customername_1</td>
<td><code>{{#IFALIAS}.regsub(&quot;(.*)_(\[0-9]+)\&quot;,&quot;\1&quot;)}</code></td>
<td></td>
</tr>
<tr>
<td>customername_1</td>
<td><code>{{#IFALIAS}.regsub(&quot;(.*)_(\[0-9]+)\&quot;,&quot;\2&quot;)}</code></td>
<td></td>
</tr>
<tr>
<td>customername_1</td>
<td><code>{{#IFALIAS}.regsub(&quot;(.*)_(\[0-9]+)\&quot;,&quot;\2&quot;)}</code></td>
<td></td>
</tr>
<tr>
<td>customername_1</td>
<td><code>{{#IFALIAS}.regsub(&quot;(.*)_(\[0-9]+)\&quot;,&quot;\2&quot;)}</code></td>
<td></td>
</tr>
</tbody>
</table>

See full item values

Long values of resolved `{ITEM.VALUE}` and `{ITEM.LASTVALUE}` macros are truncated to 20 characters. To see the full values of these macros you may use macro functions, e.g.:

```
{{ITEM.VALUE}.regsub("(.*)", \1)}<br>{{ITEM.LASTVALUE}.regsub("(.*)", \1)}
```

### 2 Secret user macros

Zabbix provides two options for protecting sensitive information in user macro values:

- **Secret text**
- **Vault secret**

Note that while the value of a secret macro is hidden, the value can be revealed through the use in items. For example, in an external script an 'echo' statement referencing a secret macro may be used to reveal the macro value to the frontend because Zabbix server has access to the real macro value.

Secret macros cannot be used in trigger expressions.

**Secret text**  Values of secret text macros are masked by the asterisks.

To make macro value 'secret', click on the button at the end of the value field and select the option Secret text.
Once the configuration is saved, it will no longer be possible to view the value.

The macro value will be displayed as asterisks.

To enter a new value, hover over the value field and press Set new value button (appears on hover).

If you change macro value type or press Set new value, current value will be erased. To revert the original value, use the backwards arrow at the right end of the Value field (only available before saving new configuration). Reverting the value will not expose it.

URLs that contain a secret macro will not work as the macro in them will be resolved as "******".

**Vault secret**  With Vault secret macros, the actual macro value is stored in an external secret management software (vault).

To configure a Vault secret macro, click on the button at the end of the Value field and select the option Vault secret.

The macro value should point to a vault secret. The input format depends on the vault provider. For provider-specific configuration examples, see:

- HashiCorp
- CyberArk

Vault secret values are retrieved by Zabbix server on every refresh of configuration data and then stored in the configuration cache. To manually trigger refresh of secret values from a vault, use the ‘secrets_reload’ command-line option.

Zabbix proxy receives values of vault secret macros from Zabbix server on each configuration sync and stores them in its own configuration cache. The proxy never retrieves macro values from the vault directly. That means a Zabbix proxy cannot start data collection after a restart until it receives the configuration data update from Zabbix server for the first time.

Encryption must be enabled between Zabbix server and proxy; otherwise a server warning message is logged.

If a macro value cannot be retrieved successfully, the corresponding item using the value will turn unsupported.

### 2 User macros

**Overview**

User macros are supported in Zabbix for greater flexibility, in addition to the macros supported out-of-the-box.

User macros can be defined on global, template and host level. These macros have a special syntax:

```
{$MACRO}
```

Zabbix resolves macros according to the following precedence:

1. host level macros (checked first)
2. macros defined for first level templates of the host (i.e., templates linked directly to the host), sorted by template ID
3. macros defined for second level templates of the host, sorted by template ID
4. macros defined for third level templates of the host, sorted by template ID, etc.
5. global macros (checked last)

In other words, if a macro does not exist for a host, Zabbix will try to find it in the host templates of increasing depth. If still not found, a global macro will be used, if exists.

If a macro with the same name exists on multiple linked templates of the same level, the macro from the template with the lowest ID will be used. Thus having macros with the same name in multiple templates is a configuration risk.

If Zabbix is unable to find a macro, the macro will not be resolved.

Macros (including user macros) are left unresolved in the Configuration section (for example, in the trigger list) by design to make complex configuration more transparent.

User macros can be used in:

- item key parameter
- item update intervals and flexible intervals
- trigger name and description
- trigger expression parameters and constants (see examples)
- many other locations - see the full list

Common use cases of global and host macros

- use a global macro in several locations; then change the macro value and apply configuration changes to all locations with one click
- take advantage of templates with host-specific attributes: passwords, port numbers, file names, regular expressions, etc.

Configuration

To define user macros, go to the corresponding location in the frontend:

- for global macros, visit Administration → General → Macros
- for host and template level macros, open host or template properties and look for the Macros tab

If a user macro is used in items or triggers in a template, it is suggested to add that macro to the template even if it is defined on a global level. That way, if the macro type is text exporting the template to XML and importing it in another system will still allow it to work as expected. Values of secret macros are not exported.

A user macro has the following attributes:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro</td>
<td>Macro name. The name must be wrapped in curly brackets and start with a dollar sign. Example: ($FRONTEND_URL). The following characters are allowed in the macro names: A-Z (uppercase only), 0-9, _, .</td>
</tr>
</tbody>
</table>
Parameter Description
---
Value
- **Macro value.** Three value types are supported:
  - **Text** (default) - plain-text value
  - **Secret text** - the value is masked with asterisks
  - **Vault secret** - the value contains a path/query to a vault secret.

To change the value type click on the button at the end of the value input field.

- **Maximum length of a user macro value is 2048 characters (255 characters in versions before 5.2.0).**

Description
- **Text field used to provide more information about this macro.**

In trigger expressions user macros will resolve if referencing a parameter or constant. They will NOT resolve if referencing a host, item key, function, operator or another trigger expression. Secret macros cannot be used in trigger expressions.

**Examples**

**Example 1**
Use of host-level macro in the “Status of SSH daemon” item key:

```plaintext
net.tcp.service[ssh,,{$SSH_PORT}]
```

This item can be assigned to multiple hosts, providing that the value of `{$SSH_PORT}` is defined on those hosts.

**Example 2**
Use of host-level macro in the “CPU load is too high” trigger:

```plaintext
last(/ca_001/system.cpu.load[,avg1])>{$MAX_CPULOAD}
```

Such a trigger would be created on the template, not edited in individual hosts.

If you want to use the amount of values as the function parameter (for example, `max(/host/key,#3)`), include hash mark in the macro definition like this: `SOME_PERIOD => #3`

**Example 3**
Use of two macros in the “CPU load is too high” trigger:

```plaintext
min(/ca_001/system.cpu.load[,avg1],{$CPULOAD_PERIOD})>{$MAX_CPULOAD}
```

Note that a macro can be used as a parameter of trigger function, in this example function `min()`.

**Example 4**
Synchronize the agent unavailability condition with the item update interval:

- define `{INTERVAL}` macro and use it in the item update interval;
- use `{INTERVAL}` as parameter of the agent unavailability trigger:

```plaintext
nodata(/ca_001/agent.ping,{INTERVAL})=1
```

**Example 5**
Centralize configuration of working hours:

- create a global `{WORKING_HOURS}` macro equal to 1-5, 09:00-18:00;
- use it in the Working time field in Administration → General → GUI;
- use it in the When active field in Administration → User → Media;
- use it to set up more frequent item polling during working hours:

  ![Update interval](image)

  ![Custom intervals](image)

- use it in the Time period action condition;
- adjust the working time in Administration → General → Macros, if needed.
Use host prototype macro to configure items for discovered hosts:

- on a host prototype define user macro \(\{$SNMPVALUE\}\) with \(\{#SNMPVALUE\}\) low-level discovery macro as a value:

<table>
<thead>
<tr>
<th>Macro</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>({$SNMPVALUE})</td>
<td>({#SNMPVALUE})</td>
</tr>
</tbody>
</table>

- assign Generic SNMPv2 template to the host prototype;
- use \(\{$SNMPVALUE\}\) in the SNMP OID field of Generic SNMPv2 template items.

User macro context

See user macros with context.

3 User macros with context

Overview

An optional context can be used in user macros, allowing to override the default value with a context-specific one.

The context is appended to the macro name; the syntax depends on whether the context is a static text value:

\{\$MACRO: "static text"\}

or a regular expression:

\{\$MACRO: regex: "regular expression"\}

Note that a macro with regular expression context can only be defined in user macro configuration. If the regex: prefix is used elsewhere as user macro context, like in a trigger expression, it will be treated as static context.

Context quoting is optional (see also important notes).

Macro context examples:

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{$LOW_SPACE_LIMIT}</td>
<td>User macro without context.</td>
</tr>
<tr>
<td>{$LOW_SPACE_LIMIT:/tmp}</td>
<td>User macro with context (static string).</td>
</tr>
<tr>
<td>{$LOW_SPACE_LIMIT:regex:&quot;^/tmp$&quot;}</td>
<td>User macro with context (regular expression). Same as {$LOW_SPACE_LIMIT:/tmp}.</td>
</tr>
<tr>
<td>{$LOW_SPACE_LIMIT:regex:&quot;^/var/log/.*$}</td>
<td>User macro with context (regular expression). Matches all strings prefixed with /var/log/.</td>
</tr>
</tbody>
</table>

Use cases

User macros with context can be defined to accomplish more flexible thresholds in trigger expressions (based on the values retrieved by low-level discovery). For example, you may define the following macros:

- \{$LOW_SPACE_LIMIT\} = 10
- \{$LOW_SPACE_LIMIT:/home\} = 20
- \{$LOW_SPACE_LIMIT:regex:"^[a-z]+$\} = 30

Then a low-level discovery macro may be used as macro context in a trigger prototype for mounted file system discovery:

\texttt{last(/host/vfs.fs.size\{#FSNAME\},pfree)\{#LOW_SPACE_LIMIT:\{#FSNAME\}\}}

After the discovery different low-space thresholds will apply in triggers depending on the discovered mount points or file system types. Problem events will be generated if:
• /home folder has less than 20% of free disk space
• folders that match the regexp pattern (like /etc, /tmp or /var) have less than 30% of free disk space
• folders that don’t match the regexp pattern and are not /home have less than 10% of free disk space

Important notes
• If more than one user macro with context exists, Zabbix will try to match the simple context macros first and then context macros with regular expressions in an undefined order.

Do not create different context macros matching the same string to avoid undefined behavior.
• If a macro with its context is not found on host, linked templates or globally, then the macro without context is searched for.
• Only low-level discovery macros are supported in the context. Any other macros are ignored and treated as plain text.

Technically, macro context is specified using rules similar to item key parameters, except macro context is not parsed as several parameters if there is a \ character:
• Macro context must be quoted with " if the context contains a \ character or starts with a " character. Quotes inside quoted context must be escaped with the \ character.
• The \ character itself is not escaped, which means it’s impossible to have a quoted context ending with the \ character - the macro {$MACRO:"a:\b\c\"} is invalid.
• The leading spaces in context are ignored, the trailing spaces are not:
  – For example {$MACRO:A} is the same as {$MACRO: A}, but not {$MACRO: A }.
• All spaces before leading quotes and after trailing quotes are ignored, but all spaces inside quotes are not:
  – Macros {$MACRO:"A"}, {$MACRO: "A"}, {$MACRO:"A"} and {$MACRO: "A"} are the same, but macros {$MACRO:"A"} and {$MACRO: "A "} are not.

The following macros are all equivalent, because they have the same context: {$MACRO:A}, {$MACRO: A} and {$MACRO: "A"}. This is in contrast with item keys, where 'key[a]', 'key[ a]' and 'key["a"]' are the same semantically, but different for uniqueness purposes.

4 Low-level discovery macros

Overview
There is a type of macro used within the low-level discovery (LLD) function:

{#MACRO}

It is a macro that is used in an LLD rule and returns real values of the file system name, network interface, SNMP OID, etc.

These macros can be used for creating item, trigger and graph prototypes. Then, when discovering real file systems, network interfaces etc., these macros are substituted with real values and are the basis for creating real items, triggers and graphs.

These macros are also used in creating host and host group prototypes in virtual machine discovery.

Some low-level discovery macros come "pre-packaged" with the LLD function in Zabbix - {#FSNAME}, {#FSTYPE}, {#IFNAME}, {#SNMPINDEX}, {#SNMPVALUE}. However, adhering to these names is not compulsory when creating a custom low-level discovery rule. Then you may use any other LLD macro name and refer to that name.

Supported locations
LLD macros can be used:
• in the low-level discovery rule filter
• for item prototypes in
  – name
  – key parameters
  – unit
  – update interval
  – history storage period
  – trend storage period
  – item value preprocessing steps
  – SNMP OID
  – IPMI sensor field
  – calculated item formula
  – SSH script and Telnet script
  – database monitoring SQL query
  – JMX item endpoint field
  – description
In all those places LLD macros can be used inside static user macro context.

Using macro functions

Macro functions are supported with low-level discovery macros (except in low-level discovery rule filter), allowing to extract a certain part of the macro value using a regular expression.

For example, you may want to extract the customer name and interface number from the following LLD macro for the purposes of event tagging:

\{#IFALIAS\}=customername_1

To do so, the `regsub` macro function can be used with the macro in the event tag value field of a trigger prototype:

```
Tags
Customer  {{#IFALIAS}.regsub("\((.*)_(\[[0-9]+\]+), (\[[0-9]+\])\)\", \1\)}, Remove
Interface {{#IFALIAS}.regsub("\((.*)_(\[[0-9]+\]+), (\[[0-9]+\])\)\", \2\)}, Remove
```

Note, that commas are not allowed in unquoted item key parameters, so the parameter containing a macro function has to be quoted. The backslash (\) character should be used to escape double quotes inside the parameter. Example:

```
net.if.in\"{{#IFALIAS}.regsub\"\((.*)_(\[[0-9]+\]+), (\[[0-9]+\])\)\"\", \1\}\"\", bytes\]
```

For more information on macro function syntax, see: Macro functions

Macro functions are supported in low-level discovery macros since Zabbix 4.0.

Footnotes

1 In the fields marked with ¹ a single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported.

6 Expression macros
Overview

Expression macros are useful for formula calculations. They are calculated by expanding all macros inside and evaluating the resulting expression.

Expression macros have a special syntax:

```{?MACRO}
```

A simple expression macro that may be used, for example, in the map label or a trigger event name, may be:

```{avg({HOST.HOST}/key,1h})
```

Only `{HOST.HOST<1-9>}` macros are supported inside expression macros. See also:

- Supported macros for a list of supported locations of the expression macro
- Example of using an expression macro in the event name

12 Users and user groups

Overview

All users in Zabbix access the Zabbix application through the web-based frontend. Each user is assigned a unique login name and a password.

All user passwords are encrypted and stored in the Zabbix database. Users cannot use their user id and password to log directly into the UNIX server unless they have also been set up accordingly to UNIX. Communication between the web server and the user browser can be protected using SSL.

With a flexible user permission schema you can restrict and differentiate rights to:

- access administrative Zabbix frontend functions
- perform certain actions in the frontend
- access monitored hosts in hostgroups
- use specific API methods

1 Configuring a user

Overview

The initial Zabbix installation has two predefined users:

- Admin - a Zabbix superuser with full permissions;
- guest - a special Zabbix user. The ‘guest’ user is disabled by default. If you add it to the Guests user group, you may access monitoring pages in Zabbix without being logged in. Note that by default, ‘guest’ has no permissions on Zabbix objects.

To configure a new user:

- Go to Administration → Users
- Click on Create user (or on the user name to edit an existing user)
- Edit user attributes in the form

General attributes

The User tab contains general user attributes:
All mandatory input fields are marked with a red asterisk.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>Unique username, used as the login name.</td>
</tr>
<tr>
<td>Name</td>
<td>User first name (optional). If not empty, visible in acknowledgment information and notification recipient information.</td>
</tr>
<tr>
<td>Last name</td>
<td>User last name (optional). If not empty, visible in acknowledgment information and notification recipient information.</td>
</tr>
<tr>
<td>Groups</td>
<td>Select user groups the user belongs to. Starting with Zabbix 3.4.3 this field is auto-complete so starting to type the name of a user group will offer a dropdown of matching groups. Scroll down to select. Alternatively, click on Select to add groups. Click on 'x' to remove the selected. Adherence to user groups determines what host groups and hosts the user will have access to.</td>
</tr>
<tr>
<td>Password</td>
<td>Two fields for entering the user password. With an existing password, contains a Password button, clicking on which opens the password fields. Note that passwords longer than 72 characters will be truncated.</td>
</tr>
<tr>
<td>Language</td>
<td>Language of the Zabbix frontend. The php gettext extension is required for the translations to work.</td>
</tr>
<tr>
<td>Time zone</td>
<td>Select the time zone to override global time zone on user level or select System default to use global time zone settings.</td>
</tr>
<tr>
<td>Theme</td>
<td>Defines how the frontend looks like: System default - use default system settings Blue - standard blue theme Dark - alternative dark theme High-contrast light - light theme with high contrast High-contrast dark - dark theme with high contrast</td>
</tr>
</tbody>
</table>
### Parameter Description

**Auto-login**
Mark this checkbox to make Zabbix remember the user and log the user in automatically for 30 days. Browser cookies are used for this.

**Auto-logout**
With this checkbox marked the user will be logged out automatically, after the set amount of seconds (minimum 90 seconds, maximum 1 day).
*Time suffixes* are supported, e.g. 90s, 5m, 2h, 1d.
Note that this option will not work:
* If the "Show warning if Zabbix server is down" global configuration option is enabled and Zabbix frontend is kept open;
* When Monitoring menu pages perform background information refreshes;
* If logging in with the Remember me for 30 days option checked.

**Refresh**
Set the refresh rate used for graphs, plain text data, etc. Can be set to 0 to disable.

**Rows per page**
You can determine how many rows per page will be displayed in lists.

**URL (after login)**
You can make Zabbix transfer the user to a specific URL after successful login, for example, to Problems page.

### User media

The Media tab contains a listing of all media defined for the user. Media are used for sending notifications. Click on Add to assign media to the user.

See the [Media types](#) section for details on configuring user media.

### Permissions

The Permissions tab contains information on:

- The user role. Users cannot change their own role.
- The user type (User, Admin, Super Admin) that is defined in the role configuration.
- Host and template groups the user has access to. Users of type 'User' and 'Admin' do not have access to any groups, templates and hosts by default. To get the access they need to be included in user groups that have access to respective entities.
- Access rights to sections and elements of Zabbix frontend, modules, and API methods. Elements to which access is allowed are displayed in green color. Light gray color means that access to the element is denied.
- Rights to perform certain actions. Actions that are allowed are displayed in green color. Light gray color means that a user does not have the rights to perform this action.

See the [User permissions](#) page for details.

### 2 Permissions

#### Overview

You can differentiate user permissions in Zabbix by defining the respective user role. Then the unprivileged users need to be included in user groups that have access to host group data.

#### User role

The user role defines which parts of UI, which API methods, and which actions are available to the user. The following roles are pre-defined in Zabbix:

<table>
<thead>
<tr>
<th>User type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guest role</td>
<td>The user has access to the Monitoring, Inventory, and Reports menu sections, but without the rights to perform any actions.</td>
</tr>
<tr>
<td>User role</td>
<td>The user has access to the Monitoring, Inventory, and Reports menu sections. The user has no access to any resources by default. Any permissions to host or template groups must be explicitly assigned.</td>
</tr>
<tr>
<td>Admin role</td>
<td>The user has access to the Monitoring, Inventory, Reports and Configuration menu sections. The user has no access to any host groups by default. Any permissions to host or template groups must be explicitly given.</td>
</tr>
<tr>
<td>Super Admin role</td>
<td>The user has access to all menu sections. The user has a read-write access to all host and template groups. Permissions cannot be revoked by denying access to specific groups.</td>
</tr>
</tbody>
</table>

*User roles* are configured in the Administration→User roles section. Super Admins can modify or delete pre-defined roles and create more roles with custom sets of permissions.
To assign a role to the user, go to the Permissions tab in the user configuration form, locate the Role field and select a role. Once a role is selected a list of associated permissions will be displayed below.

<table>
<thead>
<tr>
<th>Group</th>
<th>Type</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>All groups</td>
<td>Hosts</td>
<td>None</td>
</tr>
<tr>
<td>All groups</td>
<td>Templates</td>
<td>None</td>
</tr>
</tbody>
</table>

Permissions can be assigned for user groups only.

### Access to UI elements

- Monitoring: Dashboard, Problems, Hosts, Latest data, Maps, Discovery
- Services: Services, Service actions, SLA, SLA report
- Inventory: Overview, Hosts
- Reports: Availability report, Triggers too 100, Notifications, Scheduled reports
- Configuration: Template groups, Host groups, Templates, Hosts, Maintenance, Actions, Discovery

### Access to services

- Read-write access to services: All
- Read-only access to services: All

### Access to modules

No enabled modules found.

### Access to API

Enabled

### Access to actions

- Create and edit dashboards
- Create and edit maps
- Create and edit maintenance
- Add problem comments
- Change severity
- Acknowledge problems
- Close problems
- Execute scripts
- Manage API tokens
- Manage scheduled reports
- Manage SLA
- Invoke "Execute now" on read-only hosts

### Permissions to groups

Access to any host and template data in Zabbix is granted to user groups on the host/template group level only.

That means that an individual user cannot be directly granted access to a host (or host group). It can only be granted access to a host by being part of a user group that is granted access to the host group that contains the host.

Similarly, a user can only be granted access to a template by being part of a user group that is granted access to the template group that contains the template.

### 3 User groups

Overview
User groups allow to group users both for organizational purposes and for assigning permissions to data. Permissions to viewing and configuring data of host groups and template groups are assigned to user groups, not individual users.

It may often make sense to separate what information is available for one group of users and what - for another. This can be accomplished by grouping users and then assigning varied permissions to host and template groups.

A user can belong to any number of groups.

Configuration

To configure a user group:

- Go to Administration → User groups
- Click on Create user group (or on the group name to edit an existing group)
- Edit group attributes in the form

The **User group** tab contains general group attributes:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Group name</strong></td>
<td>Unique group name.</td>
</tr>
<tr>
<td><strong>Users</strong></td>
<td>To add users to the group start typing the name of an existing user. When the dropdown with matching user names appears, scroll down to select. Alternatively you may click the Select button to select users in a popup.</td>
</tr>
</tbody>
</table>
| **Frontend access** | How the users of the group are authenticated.  
  **System default** - use default authentication method (set globally)  
  **Internal** - use Zabbix internal authentication (even if LDAP authentication is used globally). Ignored if HTTP authentication is the global default.  
  **LDAP** - use LDAP authentication (even if internal authentication is used globally). Ignored if HTTP authentication is the global default.  
  **Disabled** - access to Zabbix frontend is forbidden for this group |
| **LDAP server** | Select which LDAP server to use to authenticate the user. This field is enabled only if Frontend access is set to LDAP or System default. |
| **Enabled**    | Status of user group and group members.  
  Checked - user group and users are enabled  
  Unchecked - user group and users are disabled |
| **Debug mode** | Mark this checkbox to activate debug mode for the users. |

All mandatory input fields are marked with a red asterisk.

The **Template permissions** tab allows to specify user group access to template group (and thereby template) data:
The **Host permissions** tab allows to specify user group access to host group (and thereby host) data:

- **Template permissions** and **Host permissions** tabs support the same set of parameters.

Current permissions to groups are displayed in the Permissions block.

If current permissions of the group are inherited by all nested groups, that is indicated by the including subgroups text in the parenthesis after the group name.

You may change the level of access to a group:

- **Read-write** - read-write access to a group;
- **Read** - read-only access to a group;
- **Deny** - access to a group denied;
- **None** - no permissions are set.

Use the selection field below to select groups and the level of access to them (note that selecting None will remove a group from the list if the group is already in the list). If you wish to include nested groups, mark the Include subgroups checkbox. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. If you wish to see all groups, click on Select.

A super admin level user can enforce the same level of permissions to the nested groups as to the parent group in the host/template group configuration form.

If a user group grants **Read-write** permissions to a host, and **None** to a template, the user will not be able to edit templated items on the host, and template name will be displayed as Inaccessible template.

The **Problem tag filter** tab allows setting tag-based permissions for user groups to see problems filtered by tag name and value:
To select a host group to apply a tag filter for, click Select to get the complete list of existing host groups or start to type the name of a host group to get a dropdown of matching groups. Only host groups will be displayed, because problem tag filter cannot be applied to template groups.

To apply tag filters to nested host groups, mark the Include subgroups checkbox.

Tag filter allows to separate the access to host group from the possibility to see problems.

For example, if a database administrator needs to see only "MySQL" database problems, it is required to create a user group for database administrators first, than specify "Service" tag name and "MySQL" value.

If "Service" tag name is specified and value field is left blank, the user group will see all problems with tag name "Service" for the selected host group. If both tag name and value fields are blank, but a host group is selected, the user group will see all problems for the specified host group.

Make sure tag name and tag value are correctly specified, otherwise, the user group will not see any problems.

Let’s review an example when a user is a member of several user groups selected. Filtering in this case will use OR condition for tags.

<table>
<thead>
<tr>
<th>User group A</th>
<th>User group B</th>
<th>Visible result for a user (member) of both groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag filter</td>
<td>Tag filter</td>
<td>Service: MySQL or Oracle problems visible</td>
</tr>
<tr>
<td>Host group</td>
<td>Host group</td>
<td>Service: MySQL or Oracle problems visible</td>
</tr>
<tr>
<td>Linux servers</td>
<td>Linux servers</td>
<td>Service: MySQL or Oracle problems visible</td>
</tr>
<tr>
<td>Tag name</td>
<td>Tag name</td>
<td>Service: MySQL or Oracle problems visible</td>
</tr>
<tr>
<td>Service</td>
<td>Service</td>
<td>Service: MySQL or Oracle problems visible</td>
</tr>
<tr>
<td>Tag value</td>
<td>Tag value</td>
<td>Service: MySQL or Oracle problems visible</td>
</tr>
<tr>
<td>MySQL</td>
<td>Oracle</td>
<td>Service: MySQL or Oracle problems visible</td>
</tr>
</tbody>
</table>

Adding a filter (for example, all tags in a certain host group "Linux servers") results in not being able to see the problems of other host groups.

Access from several user groups

A user may belong to any number of user groups. These groups may have different access permissions to hosts or templates.

Therefore, it is important to know what entities an unprivileged user will be able to access as a result. For example, let us consider how access to host X (in Hostgroup 1) will be affected in various situations for a user who is in user groups A and B.

- If Group A has only Read access to Hostgroup 1, but Group B Read-write access to Hostgroup 1, the user will get Read-write access to ‘X’.

“Read-write” permissions have precedence over “Read” permissions.

- In the same scenario as above, if ‘X’ is simultaneously also in Hostgroup 2 that is denied to Group A or B, access to ‘X’ will be unavailable, despite a Read-write access to Hostgroup 1.
• If Group A has no permissions defined and Group B has a Read-write access to Host group 1, the user will get **Read-write** access to 'X'.
• If Group A has Deny access to Host group 1 and Group B has a Read-write access to Host group 1, the user will get access to 'X' **denied**.

Other details
• An Admin level user with Read-write access to a host will not be able to link/unlink templates, if he has no access to the template group they belong to. With Read access to the template group he will be able to link/unlink templates to the host, however, will not see any templates in the template list and will not be able to operate with templates in other places.
• An Admin level user with Read access to a host will not see the host in the configuration section host list; however, the host triggers will be accessible in IT service configuration.
• Any non-Super Admin user (including 'guest') can see network maps as long as the map is empty or has only images. When hosts, host groups or triggers are added to the map, permissions are respected.
• Zabbix server will not send notifications to users defined as action operation recipients if access to the concerned host is explicitly "denied".

13 Storage of secrets

Overview  Zabbix can be configured to retrieve sensitive information from a secure vault. The following secret management services are supported: HashiCorp Vault KV Secrets Engine - Version 2, CyberArk Vault CV12.

Secrets can be used for retrieving:
• user macro values
• database access credentials

Zabbix provides read-only access to the secrets in a vault, assuming that secrets are managed by someone else.

For information about specific vault provider configuration, see: - HashiCorp configuration - CyberArk configuration

Caching of secret values  Vault secret macro values are retrieved by Zabbix server on every refresh of configuration data and then stored in the configuration cache. Zabbix proxy receives values of vault secret macros from Zabbix server on each configuration sync and stores them in its own configuration cache.

Encryption must be enabled between Zabbix server and proxy; otherwise a server warning message is logged.
To manually trigger refresh of cached secret values from a vault, use the 'secrets_reload' command-line option.
For Zabbix frontend database credentials caching is disabled by default, but can be enabled by setting the option $DB['VAULT_CACHE'] = true in zabbix.conf.php. The credentials will be stored in a local cache using the filesystem temporary file directory. The web server must allow writing in a private temporary folder (for example, for Apache the configuration option PrivateTmp= True must be set). To control how often the data cache is refreshed/invalidated, use the ZBX_DATA_CACHE_TTL constant.

TLS configuration  To configure TLS for communication between Zabbix components and the vault, add a certificate signed by a certificate authority (CA) to the system-wide default CA store. To use another location, specify the directory in the SSLCAlocation Zabbix server/proxy configuration parameter, place the certificate file inside that directory, then run the CLI command:
$ c_rehash .

CyberArk configuration

This section explains how to configure Zabbix to retrieve secrets from CyberArk Vault CV12.

The vault should be installed and configured as per the official CyberArk documentation.

To learn about configuring TLS in Zabbix, see Storage of secrets section.

Database credentials
Access to a secret with database credentials is configured for each Zabbix component separately.

Server and proxies
To obtain database credentials for Zabbix server or proxy from the vault, specify the following configuration parameters in the configuration file:
• Vault - specifies which vault provider should be used.

• VaultURL - vault server HTTP[S] URL.

• VaultDBPath - query to the vault secret containing database credentials. The credentials will be retrieved by keys ‘Content’ and ‘UserName’.

• VaultTLSCertFile, VaultTLSKeyFile - SSL certificate and key file names. Setting up these options is not mandatory, but highly recommended.

Zabbix server also uses these configuration parameters (except VaultDBPath) for vault authentication when processing vault secret macros.

Zabbix server and Zabbix proxy read the vault-related configuration parameters from zabbix_server.conf and zabbix_proxy.conf upon startup.

**Example**

In zabbix_server.conf, specify:

```
Vault=CyberArk
VaultURL=https://127.0.0.1:1858
VaultDBPath=zabbix_server&Query=Safe=passwordSafe;Object=zabbix_server_database
VaultTLSCertFile=cert.pem
VaultTLSKeyFile=key.pem
```

Zabbix will send the following API request to the vault:

```
$ curl --header "Content type: application/json" --cert cert.pem --key key.pem https://127.0.0.1:1858/AIMWebService/api/Accounts?AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix_server_database
```

Vault response, from which the keys "Content" and "UserName" should be retrieved:

```
{
  "Content": <password>,
  "UserName": <username>,
  "Address": <address>,
  "Database": <Database>,
  "PasswordChangeInProcess":<PasswordChangeInProcess>
}
```

As a result, Zabbix will use the following credentials for database authentication:

• Username: <username>

• Password: <password>

**Frontend**

To obtain database credentials for Zabbix frontend from the vault, specify required settings during frontend **installation**.

At the Configure DB Connection step, set Store credentials in parameter to CyberArk Vault.
Configure DB connection

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database host</td>
<td></td>
<td>localhost</td>
<td></td>
</tr>
<tr>
<td>Database port</td>
<td></td>
<td>0</td>
<td>0 - use default port</td>
</tr>
<tr>
<td>Database name</td>
<td></td>
<td>zabbix</td>
<td></td>
</tr>
<tr>
<td>Store credentials in</td>
<td></td>
<td>Plain text HashCorp Vault CyberArk Vault</td>
<td></td>
</tr>
<tr>
<td>* Vault API endpoint</td>
<td>yes</td>
<td><a href="https://localhost:1858">https://localhost:1858</a></td>
<td>Specify the URL for connecting to the vault in the format scheme://host:port</td>
</tr>
<tr>
<td>* Vault secret query string</td>
<td>yes</td>
<td>AppID=foo&amp;Query=Safe=bar;Object=buzz;key</td>
<td>A query, which specifies from where database credentials should be retrieved. Example: AppID=foo&amp;Query=Safe=bar;Object=buzz;key</td>
</tr>
<tr>
<td>Vault certificates</td>
<td>no</td>
<td></td>
<td>After marking the checkbox, additional parameters will appear allowing to configure client authentication. While this parameter is optional, it is highly recommended to enable it for communication with the CyberArk Vault.</td>
</tr>
<tr>
<td>SSL certificate file</td>
<td>no</td>
<td>conf/certs/cyberark-cert.pem</td>
<td>Path to SSL certificate file. The file must be in PEM format. If the certificate file contains also the private key, leave the SSL key file parameter empty.</td>
</tr>
<tr>
<td>SSL key file</td>
<td>no</td>
<td>conf/certs/cyberark-key.pem</td>
<td>Name of the SSL private key file used for client authentication. The file must be in PEM format.</td>
</tr>
<tr>
<td>Database TLS encryption</td>
<td></td>
<td>Connection will not be encrypted because it uses a socket file (on Unix) or shared memory (Windows).</td>
<td></td>
</tr>
</tbody>
</table>

Then, fill in additional parameters:

- **Vault API endpoint**: yes. Specify the URL for connecting to the vault in the format scheme://host:port. A query, which specifies from where database credentials should be retrieved. Example: AppID=foo&Query=Safe=bar;Object=buzz;key. After marking the checkbox, additional parameters will appear allowing to configure client authentication. While this parameter is optional, it is highly recommended to enable it for communication with the CyberArk Vault.

- **Vault secret query string**: yes. Path to SSL certificate file. The file must be in PEM format. If the certificate file contains also the private key, leave the SSL key file parameter empty.

User macro values

To use CyberArk Vault for storing Vault secret user macro values:

- Set the Vault provider parameter in the Administration -> General -> Other web interface section to CyberArk Vault.

Storage of secrets

- Make sure that Zabbix server is configured to work with CyberArk Vault.

The macro value should contain a query (as query: key).

See Vault secret macros for detailed information about macro value processing by Zabbix.

Query syntax

The colon symbol (:) is reserved for separating the query from the key. If a query itself contains a forward slash or a colon, these symbols should be URL-encoded (/ is encoded as %2F, : is encoded as %3A).
Example

In Zabbix: add user macro \( \{ $PASSWORD \} \) with type Vault secret and value: AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix

Zabbix will send API request to the vault:

```bash
$ curl \
    --header "Content type: application/json" \
    --cert cert.pem \
    --key key.pem \
    https://127.0.0.1:1858/AIMWebService/api/Accounts?AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix
```

Vault response, from which the key "Content" should be retrieved:

```json
{
    "Content": <password>,
    "UserName": <username>,
    "Address": <address>,
    "Database" :<Database>,
    "PasswordChangeInProcess":<PasswordChangeInProcess>
}
```

Macro resolves to the value: \(<password>\)

HashiCorp configuration

This section explains how to configure Zabbix to retrieve secrets from HashiCorp Vault KV Secrets Engine - Version 2.

The vault should be deployed and configured as per the official HashiCorp documentation.

To learn about configuring TLS in Zabbix, see Storage of secrets section.

Database credentials

Access to a secret with database credentials is configured for each Zabbix component separately.

Server and proxies

To obtain database credentials for Zabbix server or proxy from the vault, specify the following configuration parameters in the configuration file:

- **Vault** - specifies which vault provider should be used.
- **VaultToken** - vault authentication token (see Zabbix server/proxy configuration file for details).
- **VaultURL** - vault server HTTP[S] URL.
- **VaultDBPath** - path to the vault secret containing database credentials. Zabbix server or proxy will retrieve the credentials by keys 'password' and 'username'.

Zabbix server also uses these configuration parameters (except VaultDBPath) for vault authentication when processing vault secret macros.

Zabbix server and Zabbix proxy read the vault-related configuration parameters from zabbix_server.conf and zabbix_proxy.conf upon startup.

Zabbix server and Zabbix proxy will additionally read "VAULT_TOKEN" environment variable once during startup and unset it so that it would not be available through forked scripts; it is an error if both VaultToken and VAULT_TOKEN contain value.

Example
In `zabbix_server.conf`, specify:

```
Vault=HashiCorp
VaultToken=hvs.CAESIIG_PILmULFYOsEyWHxz2mF2a8VPKNLE8eHdq4autYGGh4KHGh2cy5aeTY0NFNSaUp3ZnpWbDF1RUNjUkNTZExg
VaultURL=https://127.0.0.1:8200
VaultDBPath=secret/zabbix/database
```

Run the following CLI commands to create required secret in the vault:

```
# Enable "secret/" mount point if not already enabled, note that "kv-v2" must be used
$ vault secrets enable -path=secret/ kv-v2

# Put new secrets with keys username and password under mount point "secret/" and path "secret/zabbix/database"
$ vault kv put secret/zabbix/database username=zabbix password=<password>

# Test that secret is successfully added
$ vault kv get secret/zabbix/database

# Finally test with Curl, note that "data" need to be manually added after mount point and "/v1" before the URL
```

As a result of this configuration, Zabbix server will retrieve the following credentials for database authentication:

- Username: zabbix
- Password: <password>

**Frontend**

To obtain database credentials for Zabbix frontend from the vault, specify required settings during frontend installation.

At the Configure DB Connection step, set Store credentials in parameter to HashiCorp Vault.

Then, fill in additional parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vault API endpoint</td>
<td>yes</td>
<td><a href="https://localhost:8200">https://localhost:8200</a></td>
<td>Specify the URL for connecting to the vault in the format scheme://host:port</td>
</tr>
<tr>
<td>Vault secret path</td>
<td>no</td>
<td>secret/zabbix/database</td>
<td>A path to the secret from where credentials for the database shall be retrieved by the keys 'password' and 'username'</td>
</tr>
</tbody>
</table>

**Example:** secret/zabbix/database_frontend
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vault authentication token</td>
<td>no</td>
<td>Provide an authentication token for read-only access to the secret path.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See HashiCorp documentation for information about creating tokens and vault policies.</td>
</tr>
</tbody>
</table>

**User macro values**

To use HashiCorp Vault for storing Vault secret user macro values, make sure that:

- The Vault provider parameter in the Administration -> General -> Other web interface section is set to HashiCorp Vault (default).

**Storage of secrets**

- Zabbix server is configured to work with HashiCorp Vault.

The macro value should contain a reference path (as path: key, for example, secret/zabbix:password). The authentication token specified during Zabbix server configuration (by 'VaultToken' parameter) must provide read-only access to this path.

See Vault secret macros for detailed information about macro value processing by Zabbix.

**Path syntax**

The symbols forward slash and colon are reserved. A forward slash can only be used to separate a mount point from a path (e.g. secret/zabbix where the mount point is "secret" and "zabbix" is the path) and, in case of Vault macros, a colon can only be used to separate a path/query from a key. It is possible to URL-encode "/" and ":" if there is a need to create a mount point with the name that is separated by a forward slash (e.g. foo/bar/zabbix, where the mount point is "foo/bar" and the path is "zabbix", as "foo%2Fbar/zabbix") and if a mount point name or path need to contain a colon.

**Example**

In Zabbix: add user macro \{$PASSWORD\} with type Vault secret and value secret/zabbix:password

Run the following CLI commands to create required secret in the vault:

```bash
# Enable "secret/" mount point if not already enabled, note that "kv-v2" must be used
$ vault secrets enable -path=secret/ kv-v2

# Put new secret with key password under mount point "secret/" and path "secret/zabbix"
$ vault kv put secret/zabbix password=<password>

# Test that secret is successfully added
$ vault kv get secret/zabbix

# Finally test with Curl, note that "data" need to be manually added after mount point and "/v1" before the URL
$ curl --header "X-Vault-Token: <VaultToken>" https://127.0.0.1:8200/v1/secret/data/zabbix
```

Now the macro \{$PASSWORD\} will resolve to the value: <password>

**14 Scheduled reports**
Overview

This section provides information about configuring scheduled reports.

Pre-requisites:

- Zabbix web service must be installed and configured correctly to enable scheduled report generation - see Setting up scheduled reports for instructions.

- A user must have a user role of type Admin or Super admin with the following permissions:
  - *Scheduled reports* in the *Access to UI elements* block (to view reports);
  - *Manage scheduled reports* in the *Access to actions* block (to create/edit reports).

Currently the support of scheduled reports is experimental.

To create a scheduled report in Zabbix frontend, do the following:

- Go to: Reports → Scheduled reports
- Click on Create report in the upper right corner of the screen
- Enter parameters of the report in the form

You can also create a report by opening an existing one, pressing the Clone button, and then saving under a different name.

Configuration

The scheduled reports tab contains general report attributes.
All mandatory input fields are marked with a red asterisk.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td>User that creates a report. Super admin level users are allowed to change the owner. For Admin level users, this field is read-only.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the report; must be unique.</td>
</tr>
<tr>
<td>Dashboard</td>
<td>Dashboard on which the report is based; only one dashboard can be selected at a time. To select a dashboard, start typing the name - a list of matching dashboards will appear; scroll down to select. Alternatively, you may click on Select next to the field and select a dashboard from the list in a popup window. If a dashboard contains multiple pages, only the first page will be sent as a report.</td>
</tr>
<tr>
<td>Period</td>
<td>Period for which the report will be prepared. Select one of the available options: Previous day, Previous week, Previous month, Previous year.</td>
</tr>
<tr>
<td>Cycle</td>
<td>Report generation frequency. The reports can be sent daily, weekly, monthly, or yearly. Weekly mode allows to select days of the week when the report will be sent.</td>
</tr>
<tr>
<td>StartTime</td>
<td>Time of the day in the format hh:mm when the report will be prepared.</td>
</tr>
<tr>
<td>Repeat On</td>
<td>Days of the week when the report will be sent. This field is available only if Cycle is set to weekly.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Start date</td>
<td>The date when regular report generation should be started</td>
</tr>
<tr>
<td>End date</td>
<td>The date when regular report generation should be stopped.</td>
</tr>
<tr>
<td>Subject</td>
<td>Subject of the report email. Supports {TIME} macro.</td>
</tr>
<tr>
<td>Message</td>
<td>Body of the report email. Supports {TIME} macro.</td>
</tr>
<tr>
<td>Subscriptions</td>
<td>List of report recipients. By default, includes only the report owner. Any Zabbix user with configured email media may be specified as a report recipient. Press Add user or Add user group to add more recipients. Press on the username to edit settings: Generate report by - whether the report should be generated on behalf of the report owner or the recipient. Status - select Include to send the report to user or Exclude to prevent sending the report to this user. At least one user must have Include status. Exclude status can be used to exclude specific users from a user group that is included. Note that users with insufficient permissions will see Inaccessible user or Inaccessible user group instead of the actual names in the fields Recipient and Generate report by; the fields Status and Action will be displayed as read-only.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Report status. Clearing this checkbox will disable the report.</td>
</tr>
<tr>
<td>Description</td>
<td>An optional description of the report. This description is for internal use and will not be sent to report recipients.</td>
</tr>
</tbody>
</table>

*Users with insufficient permissions are users who have a role based on the Admin user type and are not members of the user group the recipient or the report owner is a member of.*

Form buttons

Buttons at the bottom of the form allow to perform several operations.

**Add**

Add a report. This button is only available for new reports.

**Update**

Update the properties of a report.

**Clone**

Create another report based on the properties of the current report.

**Test**

Test if report configuration is correct by sending a report to the current user.

**Delete**

Delete the report.

**Cancel**

Cancel the editing of report properties.

Testing

To test a report, click on the Test button at the bottom of the report configuration form.

Test button is not available, if a report configuration form has been opened from the dashboard action menu.

If the configuration is correct, the test report is sent immediately to the current user. For test reports, subscribers and ‘generated by’ user settings are ignored.

If the configuration is incorrect, an error message is displayed describing the possible cause.
Updating a report

To update an existing report, press on the report name, then make required configuration changes and press Update button.

If an existing report is updated by another user and this user changes the Dashboard, upon pressing the Update button a warning message “Report generated by other users will be changed to the current user” will be displayed.

Pressing OK at this step will lead to the following changes:

- Generated by settings will be updated to display the user who edited the report last (unless Generated by is set to the Recipient).
- Users that have been displayed as Inaccessible user or Inaccessible user group will be deleted from the list of report subscribers.

Pressing Cancel will close the popup window and cancel the report update.

Cloning a report

To quickly clone an existing report, press Clone button at the bottom of an existing report configuration form. When cloning a report, created by another user, the current user becomes the owner of the new report.

Report settings will be copied to the new report configuration form with respect to user permissions:

- If the user that clones a report has no permissions to a dashboard, the Dashboard field will be cleared.
- If the user that clones a report has no permissions to some users or user groups in the Subscriptions list, inaccessible recipients will not be cloned.
- Generated by settings will be updated to display the current user (unless Generated by is set to the Recipient).

Change required settings and the report name, then press Add.

8. Service monitoring

Overview  Service monitoring is a business-level monitoring that can be used to get an overview of the entire IT infrastructure service tree, identify weak places of the infrastructure, calculate SLA of various IT services, and check out other information at a higher level. Service monitoring focuses on the overall availability of a service instead of low-level details, such as the lack of disk space, high processor load, etc. Since Zabbix 6.0, service monitoring also provides functionality to find the root cause of a problem if a service is not performing as expected.

Service monitoring allows to create a hierarchy representation of monitored data.

A very simple service structure may look like:
Each node of the structure has attribute status. The status is calculated and propagated to upper levels according to the selected algorithm. The status of individual nodes is affected by the status of the mapped problems. Problem mapping is accomplished with tagging.

Zabbix can send notifications or automatically execute a script on the Zabbix server in case service status change is detected. It is possible to define flexible rules whether a parent service should go into a ‘Problem state’ based on the statuses of child services. Services problem data can then be used to calculate SLA and send SLA reports based on the flexible set of conditions.

Service monitoring is configured in the Services menu, which consists of the following sections:

- **Services**
  Services section allows to build a hierarchy of your monitored infrastructure by adding parent services, and then - child services to the parent services.

  In addition to configuring service tree, this section provides an overview of the whole infrastructure and allows to quickly identify the problems that led to a service status change.

- **Service actions**
  In this section you can configure service actions. Service actions are optional and allow to: - send a notification that a service is down; - execute a remote command on a Zabbix server upon a service status change; - send a recovery notification when a service is up again.

  - **SLA**
    In this section you can define service level agreements and set service level objectives for specific services.

- **SLA report**
  In this section you can view SLA reports.

See also:
- SLA monitoring configuration example
- Notes about upgrading services from Zabbix versions below 6.0

1 Service tree

Service tree is configured in the Services->Services menu section. In the upper right corner, switch from View to the Edit mode.

To configure a new service, click on the Create service button in the top right-hand corner.

To quickly add a child service, you can alternatively press a plus icon next to the parent service. This will open the same service configuration form, but the Parent services parameter will be pre-filled.

**Service configuration** In the Service tab, specify required service parameters:
All mandatory input fields are marked with a red asterisk.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Service name. Parent services the service belongs to.</td>
</tr>
<tr>
<td>Parent services</td>
<td>Leave this field empty if you are adding the service of highest level.</td>
</tr>
<tr>
<td></td>
<td>One service may have multiple parent services. In this case, it will be</td>
</tr>
<tr>
<td></td>
<td>displayed in the service tree under each of the parent services.</td>
</tr>
<tr>
<td>Problem tags</td>
<td>Specify tags to map problem data to the service:</td>
</tr>
<tr>
<td></td>
<td><strong>Equals</strong> - include the specified tag names and values (case-sensitive)</td>
</tr>
<tr>
<td></td>
<td><strong>Contains</strong> - include the specified tag names where the tag values contain</td>
</tr>
<tr>
<td></td>
<td>the entered string (substring match, case-insensitive)</td>
</tr>
<tr>
<td></td>
<td>Tag name matching is always case-sensitive.</td>
</tr>
<tr>
<td>Sort order</td>
<td>Sort order for display, lowest comes first.</td>
</tr>
<tr>
<td>Status calculation</td>
<td>Rule for calculating service status:</td>
</tr>
<tr>
<td>rule</td>
<td><strong>Most critical if all children have problems</strong> - the most critical problem</td>
</tr>
<tr>
<td></td>
<td>in the child nodes is used to color the service status, if all children have</td>
</tr>
<tr>
<td></td>
<td>problems</td>
</tr>
<tr>
<td></td>
<td><strong>Most critical of child nodes</strong> - the most critical problem in the child</td>
</tr>
<tr>
<td></td>
<td>nodes is used to color the service status</td>
</tr>
<tr>
<td></td>
<td><strong>Set status to OK</strong> - do not calculate service status</td>
</tr>
<tr>
<td>Description</td>
<td>Mark the Advanced configuration checkbox below to configure additional status</td>
</tr>
<tr>
<td></td>
<td>calculation rules.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Mark the checkbox to access advanced configuration options.</td>
</tr>
<tr>
<td>configuration</td>
<td></td>
</tr>
</tbody>
</table>

Advanced configuration
Additional status calculation rules can only be used to increase severity level over the level calculated according to the main Status calculation rule parameter. If according to additional rules the status should be Warning, but according to the Status calculation rule the status is Disaster - the service will have status Disaster.

The **Tags** tab contains **service-level tags**. Service-level tags are used to identify a service. Tags of this type are not used to map problems to the service (for that, use **Problem tags** from the first tab).

The **Child services** tab allows to specify dependant services. Click on Add to add a service from the list of existing services. If you want to add a new child service, save this service first, then click on a plus icon next to the service that you have just created.

**Tags**
- There are two different types of tags in services:
  - Service tags
  - Problem tags

Service tags

Service tags are used to match services with service actions and SLAs. These tags are specified at the Tabs service configuration tab. For mapping SLAs OR logic is used: a service will be mapped to an SLA if it has at least one matching tag. In service actions, mapping rules are configurable and can use either AND, OR, or AND/OR logic.
Problem tags

Problem tags are used to match problems and services. These tags are specified at the primary service configuration tab. Only child services of the lowest hierarchy level may have problem tags defined and be directly correlated to problems. If problem tags match, the service status will change to the same status as the problem has. In case of several problems, a service will have the status of the most severe one. Status of a parent service is then calculated based on child services statuses according to Status calculation rules.

If several tags are specified, AND logic is used: a problem must have all tags specified in the service configuration to be mapped to the service.

A problem in Zabbix inherits tags from the whole chain of templates, hosts, items, web scenarios, and triggers. Any of these tags can be used for matching problems to services.

Example:

Problem Web camera 3 is down has tags type: video surveillance, floor: 1st and name: webcam 3 and status Warning

The service Web camera 3 has the only problem tag specified: name: webcam 3

Service status will change from OK to Warning when this problem is detected.

If the service Web camera 3 had problem tags name: webcam 3 and floor: 2nd, its status would not be changed, when the problem is detected, because the conditions are only partially met.

Modifying existing services

The buttons described below are visible only when Services section is in the Edit mode.

To edit an existing service, press the pencil icon next to the service.

To clone an existing service, press the pencil icon to open its configuration and then press Clone button. When a service is cloned, its parent links are preserved, while the child links are not.

To delete a service, press on the x icon next to it. When you delete a parent service, its child services will not be deleted and will move one level higher in the service tree (1st level children will get the same level as the deleted parent service).

Two buttons below the list of services offer some mass-editing options:

- Mass update - mass update service properties
- Delete - delete the services

To use these options, mark the checkboxes before the respective services, then click on the required button.
2 Service actions

Overview  In this section you can view and configure service actions.

Service actions are useful if you want some operations taking place as a result of service status change (OK ⇿ PROBLEM), for example:

- send message
- restart webserver

Service actions are functionally similar to other action types in Zabbix (for example, trigger actions).

Configuration  To create a new service action, go to the Service actions subsection of the Services menu, then click on Create action in the upper right corner.

Service actions are configured in the same way as other types of actions in Zabbix. For more details, see configuring actions.

The key differences are:

- User access to service actions depends on access rights to services granted by user’s role.
- Service actions support different set of conditions.

Conditions  The following conditions can be used in service actions:

<table>
<thead>
<tr>
<th>Condition type</th>
<th>Supported operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service</td>
<td>equals</td>
<td>Specify a service or a service to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>equals - event belongs to this service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>does not equal - event does not belong to this service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifying a parent service implicitly selects all child services. To</td>
</tr>
<tr>
<td></td>
<td></td>
<td>specify the parent service only, all nested services have to be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>additionally set with the does not equal operator.</td>
</tr>
<tr>
<td>Service name</td>
<td>contains</td>
<td>Specify a string in the service name or a string to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not contain</td>
<td>contains - event is generated by a service, containing this string in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>does not contain - this string cannot be found in the service name.</td>
</tr>
<tr>
<td>Service tag name</td>
<td>equals</td>
<td>Specify an event tag or an event tag to exclude.</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td>Service event tags can be defined in the service configuration section Tags.</td>
</tr>
<tr>
<td></td>
<td>contains</td>
<td>does not contain - event does not have this tag</td>
</tr>
<tr>
<td></td>
<td>does not contain</td>
<td>contains - event has a tag containing this string</td>
</tr>
<tr>
<td>Service tag value</td>
<td>equals</td>
<td>Specify an event tag and value combination to exclude. Service event tags</td>
</tr>
<tr>
<td></td>
<td>does not equal</td>
<td>can be defined in the service configuration section Tags.</td>
</tr>
<tr>
<td></td>
<td>contains</td>
<td>equals - event has this tag and value</td>
</tr>
<tr>
<td></td>
<td>does not contain</td>
<td>does not equal - event does not have this tag and value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>contains - event has a tag and value containing these strings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>does not contain - event does not have a tag and value containing these</td>
</tr>
</tbody>
</table>

Make sure to define message templates for Service actions in the Administration->Media types menu. Otherwise, the notifications will not be sent.

3 SLA

Overview  Once the services are created, you can start monitoring whether their performance is on track with service level agreement (SLA).

Services->SLA menu section allows to configure SLAs for various services. An SLA in Zabbix defines service level objective (SLO), expected uptime schedule and planned downtimes.
SLAs and services are matched by service tags. The same SLA may be applied to multiple services - performance will be measured for each matching service separately. A single service may have multiple SLAs assigned - data for each of the SLAs will be displayed separately.

In SLA reports Zabbix provides Service level indicator (SLI) data, which measures real service availability. Whether a service meets the SLA targets is determined by comparing SLO (expected availability in %) with SLI (real-life availability in %).

**Configuration** To create a new SLA, click on the Create SLA button.

The **SLA** tab allows to specify general SLA parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter the SLA name.</td>
</tr>
<tr>
<td>SLO</td>
<td>Enter the service level objective (SLO) as percentage.</td>
</tr>
<tr>
<td>Reporting period</td>
<td>Selecting the period will affect what periods are used in the SLA report - daily, weekly, monthly, quarterly, or annually.</td>
</tr>
<tr>
<td>Time zone</td>
<td>Select the SLA time zone.</td>
</tr>
<tr>
<td>Schedule</td>
<td>Select the SLA schedule - 24x7 or custom.</td>
</tr>
<tr>
<td>Effective date</td>
<td>Select the date of starting SLA calculation.</td>
</tr>
<tr>
<td>Service tags</td>
<td>Add service tags to identify the services towards which this SLA should be applied. Name - service tag name, must be exact match, case-sensitive. Operation - select Equals if the tag value must match exactly (case-sensitive) or Contains if part of the tag value must match (case-insensitive). Value - service tag value to search for according to selected operation.</td>
</tr>
<tr>
<td>Description</td>
<td>Add a description for the SLA.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Mark the checkbox to enable the SLA calculation.</td>
</tr>
</tbody>
</table>

The **Excluded downtimes** tab allows to specify downtimes that are excluded from the SLA calculation.
Click on Add to configure excluded downtimes, then enter the period name, start date and duration.

**SLA reports**  How a service performs compared to an SLA is visible in the SLA report. SLA reports can be viewed:

- from the SLA section by clicking on the SLA report hyperlink;
- from the Services section by clicking on the SLA name in the info tab;
- in the Dashboard widget SLA report.

Once an SLA is configured, the Info tab in the services section will also display some information about service performance.

### 4 Setup example

**Overview**  This section describes a simple setup for monitoring Zabbix high availability cluster as a service.

**Pre-requisites**  Prior to configuring service monitoring, you need to have the hosts configured:

- HA node 1 with at least one trigger and a tag (preferably set on a trigger level) component:HA node 1
- HA node 2 with at least one trigger and a tag (preferably set on a trigger level) component:HA node 2

**Service tree**  The next step is to build the service tree. In this example, the infrastructure is very basic and consists of three services: Zabbix cluster (parent) and two child services Zabbix server node 1 and Zabbix server node 2.

```
Zabbix cluster
|-- Zabbix server node 1
|-- Zabbix server node 2
```

At the Services page, turn on Edit mode and press Create service:

In the service configuration window, enter name Zabbix cluster and mark the checkbox Advanced configuration.
Configure additional rule:

Zabbix cluster will have two child services - one for each of the HA nodes. If both HA nodes have problems of at least Warning status, parent service status should be set to Disaster. To achieve this, additional rule should be configured as:

- Set status to: Disaster
- Condition: If at least \( N \) child services have Status status or above
- \( N: 2 \)
- Status: Warning

Switch to the Tags tab and add a tag Zabbix: server. This tag will be used later for service actions and SLA reports.
Save the new service.

To add a child service, press on the plus icon next to the Zabbix cluster service (the icon is visible only in Edit mode).

In the service configuration window, enter name Zabbix server node 1. Note that the Parent services parameter is already pre-filled with Zabbix cluster.

Availability of this service is affected by problems on the host HA node 1, marked with component:HA node 1 problem tag. In the Problem tags parameter, enter:

- Name: component
- Operation: Equals
- Value: HA node 1

Switch to the Tags tab and add a service tag: Zabbix server:node 1. This tag will be used later for service actions and SLA reports.
Save the new service.

Create another child service of Zabbix cluster with name "Zabbix server node 2".

Set the Problem tags as:

- Name: component
- Operation: Equals
- Value: HA node 2

Switch to the Tags tab and add a service tag: Zabbix server: node 2.

Save the new service.

**SLA** In this example, expected Zabbix cluster performance is 100% excluding semi-annual one hour maintenance period.

First, you need to add a new service level agreement.

Go to the Services->SLA menu section and press Create SLA. Enter name Zabbix cluster performance and set the SLO to 100%.

The service Zabbix cluster has a service tag Zabbix: server. To use this SLA for measuring performance of Zabbix cluster, in the Service tags parameter, specify:

- Name: Zabbix
- Operation: Equals
- Value: server
In a real-life setup, you can also update desired reporting period, time zone and start date or change the schedule from 24/7 to custom. For this example, the default settings are sufficient.

Switch to the Excluded downtimes tab and add downtimes for scheduled maintenance periods to exclude these periods from SLA calculation. In the Excluded downtimes section press the Add link, enter downtime name, planned start time and duration.

Press Add to save the new SLA.

Switch to the SLA reports section to view the SLA report for Zabbix cluster.

The SLA info can also be checked in the Services section.
9. Web monitoring

Overview  With Zabbix you can check several availability aspects of web sites.

To perform web monitoring Zabbix server must be initially configured with cURL (libcurl) support.

To activate web monitoring you need to define web scenarios. A web scenario consists of one or several HTTP requests or “steps”. The steps are periodically executed by Zabbix server in a pre-defined order. If a host is monitored by proxy, the steps are executed by the proxy.

Web scenarios are attached to hosts/templates in the same way as items, triggers, etc. That means that web scenarios can also be created on a template level and then applied to multiple hosts in one move.

The following information is collected in any web scenario:

- average download speed per second for all steps of whole scenario
- number of the step that failed
- last error message

The following information is collected in any web scenario step:

- download speed per second
- response time
- response code

For more details, see web monitoring items.

Data collected from executing web scenarios is kept in the database. The data is automatically used for graphs, triggers and notifications.

Zabbix can also check if a retrieved HTML page contains a pre-defined string. It can execute a simulated login and follow a path of simulated mouse clicks on the page.

Zabbix web monitoring supports both HTTP and HTTPS. When running a web scenario, Zabbix will optionally follow redirects (see option Follow redirects below). Maximum number of redirects is hard-coded to 10 (using cURL option CURLOPT_MAXREDIRS). All cookies are preserved during the execution of a single scenario.

See also known issues for web monitoring using HTTPS protocol.

Configuring a web scenario  To configure a web scenario:

- Go to: Configuration → Hosts (or Templates)
- Click on Web in the row of the host/template
- Click on Create scenario to the right (or on the scenario name to edit an existing scenario)
- Enter parameters of the scenario in the form
The **Scenario** tab allows you to configure the general parameters of a web scenario.

### Scenario parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Name of the host/template that the scenario belongs to.</td>
</tr>
<tr>
<td>Name</td>
<td>Unique scenario name.</td>
</tr>
<tr>
<td>Update interval</td>
<td>How often the scenario will be executed. <strong>Time suffixes</strong> are supported, e.g. 30s, 1m, 2h, 1d. <strong>User macros</strong> are supported. Note that if a user macro is used and its value is changed (e.g. 5m → 30s), the next check will be executed according to the previous value (farther in the future with the example values). New web scenarios will be checked within 60 seconds of their creation.</td>
</tr>
<tr>
<td>Attempts</td>
<td>The number of attempts for executing web scenario steps. In case of network problems (timeout, no connectivity, etc) Zabbix can repeat executing a step several times. The figure set will equally affect each step of the scenario. Up to 10 attempts can be specified, default value is 1. Note: Zabbix will not repeat a step because of a wrong response code or the mismatch of a required string.</td>
</tr>
<tr>
<td>Agent</td>
<td>Select a client agent. Zabbix will pretend to be the selected browser. This is useful when a website returns different content for different browsers. <strong>User macros</strong> can be used in this field.</td>
</tr>
</tbody>
</table>

All mandatory input fields are marked with a red asterisk.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP proxy</td>
<td>You can specify an HTTP proxy to use, using the format [protocol://][username[:password]@]proxy.example.com[:port]. This sets the CURLOPT_PROXY cURL option. The optional protocol:// prefix may be used to specify alternative proxy protocols (the protocol prefix support was added in cURL 7.21.7). With no protocol specified, the proxy will be treated as an HTTP proxy. By default, 1080 port will be used. If specified, the proxy will overwrite proxy related environment variables like http_proxy, HTTPS_PROXY. If not specified, the proxy will not overwrite proxy-related environment variables. The entered value is passed on &quot;as is&quot;, no sanity checking takes place. You may also enter a SOCKS proxy address. If you specify the wrong protocol, the connection will fail and the item will become unsupported. Note that only simple authentication is supported with HTTP proxy. User macros can be used in this field.</td>
</tr>
</tbody>
</table>
| Variables    | Variables that may be used in scenario steps (URL, post variables). They have the following format:  
{macro1}=value1  
{macro2}=value2  
{macro3}=regex:<regular expression>  
For example:  
{username}=Alexei  
{password}=kj3h5kJ34bd  
{hostid}=regex:hostidis([0-9]+)  
The macros can then be referenced in the steps as {username}, {password} and {hostid}. Zabbix will automatically replace them with actual values. Note that variables with regex: need one step to get the value of the regular expression so the extracted value can only be applied to the step after. If the value part starts with regex: then the part after it is treated as a regular expression that searches the web page and, if found, stores the match in the variable. At least one subgroup must be present so that the matched value can be extracted. User macros and {HOST.*} macros are supported. Variables are automatically URL-encoded when used in query fields or form data for post variables, but must be URL-encoded manually when used in raw post or directly in URL. |
| Headers      | Custom HTTP headers that will be sent when performing a request. Headers should be listed using the same syntax as they would appear in the HTTP protocol, optionally using some additional features supported by the CURLOPT_HTTPHEADER cURL option. For example:  
Accept-Charset=utf-8  
Accept-Language=en-US  
Content-Type=application/xml; charset=utf-8  
User macros and {HOST.*} macros are supported. |
| Enabled      | The scenario is active if this box is checked, otherwise - disabled. |

Note that when editing an existing scenario, two extra buttons are available in the form:

- Clone: Create another scenario based on the properties of the existing one.
- Clear history and trends: Delete history and trend data for the scenario. This will make the server perform the scenario immediately after deleting the data.

If HTTP proxy field is left empty, another way for using an HTTP proxy is to set proxy related environment variables.

For HTTP checks - set the http_proxy environment variable for the Zabbix server user. For example, http_proxy=http://proxy_ip:proxy_port.

For HTTPS checks - set the HTTPS_PROXY environment variable. For example, HTTPS_PROXY=http://proxy_ip:proxy_port.

More details are available by running a shell command: # man curl.

The Steps tab allows you to configure the web scenario steps. To add a web scenario step, click on Add in the Steps block.
Secret user macros must not be used in URLs as they will resolve to "*****".

### Step of web scenario

<table>
<thead>
<tr>
<th>Name</th>
<th>URL</th>
<th>Required</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site availability</td>
<td><a href="http://www.example.com">http://www.example.com</a></td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>About</td>
<td><a href="http://www.example.com/about">http://www.example.com/about</a></td>
<td>200</td>
<td></td>
</tr>
</tbody>
</table>

---

### Configuring steps

**Name**: Site availability

**URL**: http://www.example.com

**Query fields**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

**Post type**: Form data

**Post fields**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

**Variables**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

**Headers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

**Follow redirects**: Yes

**Retrieve mode**: Body

**Timeout**: 15s

**Required string**: pattern

**Required status codes**: 200
Step parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Unique step name.</td>
</tr>
</tbody>
</table>
| URL       | URL to connect to and retrieve data. For example:  
https://www.example.com  
http://www.example.com/download  
Domain names can be specified in Unicode characters. They are automatically punycode-converted to ASCII when executing the web scenario step.  
The Parse button can be used to separate optional query fields (like ?name=Admin&password=mypassword) from the URL, moving the attributes and values into Query fields for automatic URL-encoding.  
Variables can be used in the URL, using the {macro} syntax. Variables can be URL-encoded manually using a {{macro}.urlencode()} syntax.  
User macros and {HOST.*} macros are supported. Limited to 2048 characters. |
| Query fields | HTTP GET variables for the URL.  
Specified as attribute and value pairs.  
Values are URL-encoded automatically. Values from scenario variables, user macros or {HOST.*} macros are resolved and then URL-encoded automatically. Using a {{macro}.urlencode()} syntax will double URL-encode them.  
User macros and {HOST.*} macros are supported. |
| Post | HTTP POST variables.  
In Form data mode, specified as attribute and value pairs.  
Values are URL-encoded automatically. Values from scenario variables, user macros or {HOST.*} macros are resolved and then URL-encoded automatically.  
In Raw data mode, attributes/values are displayed on a single line and concatenated with a & symbol.  
Raw values can be URL-encoded/decoded manually using a {{macro}.urlencode()} or {{macro}.urldecode()} syntax.  
For example: id=2345&userid={user}  
If {user} is defined as a variable of the web scenario, it will be replaced by its value when the step is executed. If you wish to URL-encode the variable, substitute {user} with {{user}.urlencode()}.  
User macros and {HOST.*} macros are supported. |
| Variables | Step-level variables that may be used for GET and POST functions.  
Specified as attribute and value pairs.  
Step-level variables override scenario-level variables or variables from the previous step.  
However, the value of a step-level variable only affects the step after (and not the current step).  
They have the following format:  
{macro}=value  
{macro}=regex:<regular expression>  
For more information see variable description on the scenario level.  
Variables are automatically URL-encoded when used in query fields or form data for post variables, but must be URL-encoded manually when used in raw post or directly in URL. |
| Headers | Custom HTTP headers that will be sent when performing a request.  
Specified as attribute and value pairs.  
Headers on the step level will overwrite the headers specified for the scenario.  
For example, setting a 'User-Agent' attribute with no value will remove the User-Agent value set on scenario level.  
User macros and {HOST.*} macros are supported.  
This sets the CURLOPT_HTTPHEADER cURL option. |
| Follow redirects | Mark the checkbox to follow HTTP redirects.  
This sets the CURLOPT_FOLLOWLOCATION cURL option. |
| Retrieve mode | Select the retrieve mode:  
**Body** - retrieve only body from the HTTP response  
**Headers** - retrieve only headers from the HTTP response  
**Body and headers** - retrieve body and headers from the HTTP response |
| Timeout | Zabbix will not spend more than the set amount of time on processing the URL (from one second to maximum of 1 hour). Actually this parameter defines the maximum time for making connection to the URL and maximum time for performing an HTTP request. Therefore, Zabbix will not spend more than 2 x Timeout seconds on the step.  
Time suffixes are supported, e.g. 30s, 1m, 1h. User macros are supported. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required string</td>
<td>Required regular expression pattern. Unless retrieved content (HTML) matches the required pattern the step will fail. If empty, no check on required string is performed. For example: Homepage of Zabbix Welcome.<em>admin Note: Referencing regular expressions created in the Zabbix frontend is not supported in this field. User macros and {HOST.</em>} macros are supported.</td>
</tr>
<tr>
<td>Required status codes</td>
<td>List of expected HTTP status codes. If Zabbix gets a code which is not in the list, the step will fail. If empty, no check on status codes is performed. For example: 200,201,210-299 User macros are supported.</td>
</tr>
</tbody>
</table>

Any changes in web scenario steps will only be saved when the whole scenario is saved.

See also a real-life example of how web monitoring steps can be configured.

**Configuring tags**  The Tags tab allows to define scenario-level tags.

![Tags tab](image)

Tagging allows to filter web scenarios and web monitoring items.

**Configuring authentication**  The Authentication tab allows you to configure scenario authentication options. A green dot next to the tab name indicates that some type of HTTP authentication is enabled.

![Authentication tab](image)

Authentication parameters:
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| Authentication  | Authentication options.  
None - no authentication used.  
Basic - basic authentication is used.  
NTLM - NTLM (Windows NT LAN Manager) authentication is used.  
Kerberos - Kerberos authentication is used. See also: Configuring Kerberos with Zabbix.  
Digest - Digest authentication is used.  
Selecting an authentication method will provide two additional fields for entering a user name and password.  
User macros can be used in user and password fields. |
| SSL verify peer | Mark the checkbox to verify the SSL certificate of the web server.  
The server certificate will be automatically taken from system-wide certificate authority (CA) location. You can override the location of CA files using Zabbix server or proxy configuration parameter SSLCA LOCATION.  
This sets the CURLOPT_SSL_VERIFYPEER cURL option. |
| SSL verify host | Mark the checkbox to verify that the Common Name field or the Subject Alternate Name field of the web server certificate matches.  
This sets the CURLOPT_SSL_VERIFYHOST cURL option. |
| SSL certificate file | Name of the SSL certificate file used for client authentication. The certificate file must be in PEM format. If the certificate file contains also the private key, leave the SSL key file field empty. If the key is encrypted, specify the password in SSL key password field. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSL CERT LOCATION.  
HOST.* macros and user macros can be used in this field.  
This sets the CURLOPT_SSLCERT cURL option. |
| SSL key file | Name of the SSL private key file used for client authentication. The private key file must be in PEM format. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSL KEY LOCATION.  
HOST.* macros and user macros can be used in this field.  
This sets the CURLOPT_SSLKEY cURL option. |
| SSL key password | SSL private key file password.  
User macros can be used in this field.  
This sets the CURLOPT_SSL Key Password cURL option. |

[1] Zabbix supports certificate and private key files in PEM format only. In case you have your certificate and private key data in PKCS #12 format file (usually with extension *.p12 or *.pfx) you may generate the PEM file from it using the following commands:  
openssl pkcs12 -in ssl-cert.p12 -clcerts -nokeys -out ssl-cert.pem  
openssl pkcs12 -in ssl-cert.p12 -nocerts -nodes -out ssl-cert.key  
Zabbix server picks up changes in certificates without a restart.  
If you have client certificate and private key in a single file just specify it in a "SSL certificate file" field and leave "SSL key file" field empty. The certificate and key must still be in PEM format. Combining certificate and key is easy:  
cat client.crt client.key > client.pem  
**Display**: To view web scenarios configured for a host, go to Monitoring → Hosts, locate the host in the list and click on the Web hyperlink in the last column. Click on the scenario name to get detailed information.
An overview of web scenarios can also be displayed in Monitoring → Dashboard by a Web monitoring widget.

Recent results of the web scenario execution are available in the Monitoring → Latest data section.

Extended monitoring Sometimes it is necessary to log received HTML page content. This is especially useful if some web scenario step fails. Debug level 5 (trace) serves that purpose. This level can be set in server and proxy configuration files or using a runtime control option (-R log_level_increase="http poller,N", where N is the process number). The following examples demonstrate how extended monitoring can be started provided debug level 4 is already set:

Increase log level of all http pollers:
shell> zabbix_server -R log_level_increase="http poller"

Increase log level of second http poller:
shell> zabbix_server -R log_level_increase="http poller,2"

If extended web monitoring is not required it can be stopped using the -R log_level_decrease option.

1 Web monitoring items

Overview
Some new items are automatically added for monitoring when web scenarios are created.

All items inherit tags from the web scenario.

Scenario items

As soon as a scenario is created, Zabbix automatically adds the following items for monitoring.

<table>
<thead>
<tr>
<th>Item Description</th>
<th>Item key: web.test.in[Scenario,bps]</th>
<th>Type: Numeric(float)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download speed for scenario</td>
<td>This item will collect information about the download speed (bytes per second) of the whole scenario, i.e. average for all steps.</td>
<td></td>
</tr>
<tr>
<td>Failed step of scenario</td>
<td>This item will display the number of the step that failed on the scenario. If all steps are executed successfully, 0 is returned.</td>
<td></td>
</tr>
<tr>
<td>Last error message of scenario</td>
<td>This item returns the last error message text of the scenario. A new value is stored only if the scenario has a failed step. If all steps are ok, no new value is collected.</td>
<td></td>
</tr>
</tbody>
</table>

The actual scenario name will be used instead of “Scenario”.

Web monitoring items are added with a 30 day history and a 90 day trend retention period.

If scenario name starts with a doublequote or contains comma or square bracket, it will be properly quoted in item keys. In other cases no additional quoting will be performed.

These items can be used to create triggers and define notification conditions.

Example 1

To create a “Web scenario failed” trigger, you can define a trigger expression:

```
last(/host/web.test.fail[Scenario])<>0
```

Make sure to replace ‘Scenario’ with the real name of your scenario.

Example 2

To create a “Web scenario failed” trigger with a useful problem description in the trigger name, you can define a trigger with name:

```
Web scenario "Scenario" failed: {ITEM.VALUE}
```

and trigger expression:

```
length(last(/host/web.test.error[Scenario]))>0 and last(/host/web.test.fail[Scenario])>0
```

Make sure to replace ‘Scenario’ with the real name of your scenario.

Example 3

To create a “Web application is slow” trigger, you can define a trigger expression:

```
last(/host/web.test.in[Scenario,,bps])<10000
```

Make sure to replace ‘Scenario’ with the real name of your scenario.

Scenario step items

As soon as a step is created, Zabbix automatically adds the following items for monitoring.

<table>
<thead>
<tr>
<th>Item Description</th>
<th>Item key: web.test.in[Scenario,Step,bps]</th>
<th>Type: Numeric(float)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Download speed for step</td>
<td>This item will collect information about the download speed (bytes per second) of the step.</td>
<td></td>
</tr>
</tbody>
</table>

497
### Item Description

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
</table>
| Response time for step `<Step>` of scenario `<Scenario>` | This item will collect information about the response time of the step in seconds. Response time is counted from the beginning of the request until all information has been transferred. Item key: `web.test.time[Scenario,Step,resp]`  
Type: Numeric(float) |
| Response code for step `<Step>` of scenario `<Scenario>` | This item will collect response codes of the step. Item key: `web.test.rspcode[Scenario,Step]`  
Type: Numeric(unsigned) |

Actual scenario and step names will be used instead of “Scenario” and “Step” respectively.

Web monitoring items are added with a 30 day history and a 90 day trend retention period.

If scenario name starts with a doublequote or contains comma or square bracket, it will be properly quoted in item keys. In other cases no additional quoting will be performed.

These items can be used to create triggers and define notification conditions. For example, to create a “Zabbix GUI login is too slow” trigger, you can define a trigger expression:

```plaintext
last(/zabbix/web.test.time[ZABBIX GUI,Login,resp])>3
```

### 2 Real life scenario

Overview

This section presents a step-by-step real-life example of how web monitoring can be used.

Let’s use Zabbix web monitoring to monitor the web interface of Zabbix. We want to know if it is available, provides the right content and how quickly it works. To do that we also must log in with our user name and password.

**Scenario**

**Step 1**

Add a new web scenario.

We will add a scenario to monitor the web interface of Zabbix. The scenario will execute a number of steps.

Go to Configuration → Hosts, pick a host and click on Web in the row of that host. Then click on Create web scenario.
All mandatory input fields are marked with a red asterisk.

In the new scenario form we will name the scenario as Zabbix frontend. We will also create two variables: `{user}` and `{password}`. You may also want to add a new Application:Zabbix frontend tag in the Tags tab.

**Step 2**

Define steps for the scenario.

Click on Add button in the Steps tab to add individual steps.

**Web scenario step 1**

We start by checking that the first page responds correctly, returns with HTTP response code 200 and contains text "Zabbix SIA".
**Step of web scenario**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>First page</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://localhost/zabbix/index.php">http://localhost/zabbix/index.php</a></td>
</tr>
</tbody>
</table>

**Query fields**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

**Post type**

- Form data
- Raw data

**Post fields**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

**Variables**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

**Headers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

**Follow redirects**

- On

**Retriev mode**

- Body
- Headers
- Body and headers

**Timeout**

- 15s

**Required string**

- Zabbix SIA

**Required status codes**

- 200

When done configuring the step, click on Add.

Web scenario step 2

We continue by logging into the Zabbix frontend, and we do so by reusing the macros (variables) we defined on the scenario level - {user} and {password}.
Note that Zabbix frontend uses JavaScript redirect when logging in, thus first we must login, and only in further steps we may check for logged-in features. Additionally, the login step must use full URL to `index.php` file.

Take note also of how we are getting the content of the `{sid}` variable (session ID) using a variable syntax with regular expression: `regex:name="csrf-token" content="([0-9a-z]{16})"`. This variable will be required in step 4.

Web scenario step 3

Being logged in, we should now verify the fact. To do so, we check for a string that is only visible when logged in - for example, `Administration`. 
Now that we have verified that frontend is accessible and we can log in and retrieve logged-in content, we should also log out - otherwise Zabbix database will become polluted with lots and lots of open session records.
Step of web scenario

* Name: Log out
* URL: http://localhost/zabbix/index.php

Query fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
<td>{sid}</td>
</tr>
<tr>
<td>reconnect</td>
<td>1</td>
</tr>
</tbody>
</table>

Post type: Form data

Post fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

Headers

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

Follow redirects: ✔️

Retrieve mode: Body

* Timeout: 15s

Required string

Required status codes: 200

Web scenario step 5

We can also check that we have logged out by looking for the Username string.
A complete configuration of web scenario steps should look like this:

**Name**: Logout check

**URL**: http://localhost/zabbix/index.php

**Query fields**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

**Post fields**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

**Variables**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

**Headers**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

Follow redirects

Retrieve mode

- Body
- Headers
- Body and headers

**Timeout**: 15s

**Required string**: Username

**Required status codes**: 200
Step 3

Save the finished web monitoring scenario.

The scenario will be added to a host. To view web scenario information go to Monitoring → Hosts, locate the host in the list and click on the Web hyperlink in the last column.

Click on the scenario name to see more detailed statistics:
10. Virtual machine monitoring

Overview  Support of monitoring VMware environments is available in Zabbix starting with version 2.2.0. Zabbix can use low-level discovery rules to automatically discover VMware hypervisors and virtual machines and create hosts to monitor them, based on predefined host prototypes.

The default dataset in Zabbix offers several ready-to-use templates for monitoring VMware vCenter or ESX hypervisor.

The minimum required VMware vCenter or vSphere version is 5.1.

Details  The virtual machine monitoring is done in two steps. First, virtual machine data is gathered by VMware collector Zabbix processes. Those processes obtain necessary information from VMware web services over the SOAP protocol, pre-process it and store into Zabbix server shared memory. Then, this data is retrieved by pollers using Zabbix simple check VMware keys.

Starting with Zabbix version 2.4.4 the collected data is divided into 2 types: VMware configuration data and VMware performance counter data. Both types are collected independently by VMware collectors. Because of this it is recommended to enable more collectors than the monitored VMware services. Otherwise retrieval of VMware performance counter statistics might be delayed by the retrieval of VMware configuration data (which takes a while for large installations).
Currently only datastore, network interface and disk device statistics and custom performance counter items are based on the VMware performance counter information.

**Configuration** For virtual machine monitoring to work, Zabbix should be **compiled** with the `--with-libxml2` and `--with-libcurl` compilation options.

The following configuration file options can be used to tune the Virtual machine monitoring:

- **StartVMwareCollectors** - the number of pre-forked vmware collector instances.
  
  This value depends on the number of VMware services you are going to monitor. For the most cases this should be:
  
  \[
  \text{servicenum} < \text{StartVMwareCollectors} < (\text{servicenum} \times 2)
  \]
  
  where `servicenum` is the number of VMware services. E.g. if you have 1 VMware service to monitor set `StartVMwareCollectors` to 2, if you have 3 VMware services, set it to 5. Note that in most cases this value should not be less than 2 and should not be 2 times greater than the number of VMware services that you monitor. Also keep in mind that this value also depends on your VMware environment size and VMwareFrequency and VMwarePerfFrequency configuration parameters (see below).

- **VMwareCacheSize**
- **VMwareFrequency**
- **VMwarePerfFrequency**
- **VMwareTimeout**

For more details, see the configuration file pages for Zabbix **server** and **proxy**.

To support datastore capacity metrics Zabbix requires VMware configuration `vpxd.stats.maxQueryMetrics` parameter to be at least 64. See also the VMware knowledge base **article**.

**Discovery** Zabbix can use a low-level discovery rule to automatically discover VMware hypervisors and virtual machines.
**Host prototypes**  Host prototypes can be created with the low-level discovery rule. When virtual machines are discovered, these prototypes become real hosts. For information about creating host prototypes, see [low-level discovery](#).

**Ready-to-use templates**  The default dataset in Zabbix offers several ready-to-use templates for monitoring VMware vCenter or directly ESX hypervisor. These templates contain pre-configured LLD rules as well as a number of built-in checks for monitoring virtual installations.

- **Templates for VMware vCenter and ESX hypervisor monitoring:**
  - VMware - uses UUID data for corresponding macros;
  - VMware FQDN - uses FQDN data for corresponding macros.

In order for the VMware FQDN template to work correctly each monitored VM should have a unique OS name compliant with FQDN rules and VMware Tools must be installed on every machine. If these conditions are met, it is recommended to use VMware FQDN template. The creation of VMware FQDN template became possible after introducing the ability to create hosts with custom interfaces in Zabbix 5.2. A classic VMware template is still available and can be used if FQDN requirements cannot be met. Please keep in mind, that the VMware template has a known issue. Hosts for discovered virtual machines will be created with the names saved in the vCenter (for example, VM1, VM2, etc.). If Zabbix agent active is installed on these hosts later with autoregistration enabled, the autoregistration process will read host names as they have been registered upon launch (for example, vm1.example.com, vm2.example.com, etc.) and create new hosts since no name matches have been found. As a result there will be two duplicate hosts for each machine with different names.

---

**Discovery rule**

<table>
<thead>
<tr>
<th><strong>Discovery rule</strong></th>
<th><strong>Preprocessing</strong></th>
<th><strong>LLD macros</strong></th>
<th><strong>Filters</strong></th>
<th><strong>Overrides</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Discover VMware hypervisors</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Simple check</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td>vmware hv discovery[{$VMWARE.URL}]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Host interface</strong></td>
<td>192.0.2.255:10050</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>User name</strong></td>
<td>{$VMWARE.USERNAME}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>{$VMWARE.PASSWORD}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Update interval</strong></td>
<td>1h</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Custom intervals**

<table>
<thead>
<tr>
<th><strong>Type</strong></th>
<th><strong>Interval</strong></th>
<th><strong>Period</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexible</td>
<td>Scheduling</td>
<td>50s</td>
</tr>
</tbody>
</table>

**Enabled**

Add

Test

Cancel

All mandatory input fields are marked with a red asterisk.

The discovery rule key in the above screenshot is `vmware hv discovery[{$VMWARE.URL}]`.

In order for the VMware FQDN template to work correctly each monitored VM should have a unique OS name compliant with FQDN rules and VMware Tools must be installed on every machine. If these conditions are met, it is recommended to use VMware FQDN template. The creation of VMware FQDN template became possible after introducing the ability to create hosts with custom interfaces in Zabbix 5.2. A classic VMware template is still available and can be used if FQDN requirements cannot be met. Please keep in mind, that the VMware template has a known issue. Hosts for discovered virtual machines will be created with the names saved in the vCenter (for example, VM1, VM2, etc.). If Zabbix agent active is installed on these hosts later with autoregistration enabled, the autoregistration process will read host names as they have been registered upon launch (for example, vm1.example.com, vm2.example.com, etc.) and create new hosts since no name matches have been found. As a result there will be two duplicate hosts for each machine with different names.
Templates used by discovery (normally, these templates should not be manually linked to a host):

- VMware Hypervisor;
- VMware Guest.

### Host configuration

To use VMware simple checks the host must have the following user macros defined:

- `{VMMWARE.URL}` - VMware service (vCenter or ESX hypervisor) SDK URL (https://servername/sdk)
- `{VMMWARE.USERNAME}` - VMware service user name
- `{VMMWARE.PASSWORD}` - VMware service `{VMMWARE.USERNAME}` user password

### Example

The following example demonstrates how to quickly setup VMware monitoring on Zabbix:

- compile zabbix server with required options (--with-libxml2 and --with-libcurl)
- set the StartVMwareCollectors option in Zabbix server configuration file to 1 or more
- create a new host
- set the host macros required for VMware authentication:

  ```
  {{..:..:assets:en:manual:vm_monitoring:vm_host_macros.png|}}
  ```

  * Link the host to the VMware service template:

  ```
  {{..:..:assets:en:manual:vm_monitoring:vm_host_templates.png|}}
  ```

  * Click on the //Add// button to save the host

### Extended logging

The data gathered by VMware collector can be logged for detailed debugging using debug level 5. This level can be set in server and proxy configuration files or using a runtime control option (`-R log_level_increase="vmware collector,N"`, where N is a process number). The following examples demonstrate how extended logging can be started provided debug level 4 is already set:

- Increase log level of all vmware collectors:
  ```
  shell> zabbix_server -R log_level_increase="vmware collector"
  ```

- Increase log level of second vmware collector:
  ```
  shell> zabbix_server -R log_level_increase="vmware collector,2"
  ```

If extended logging of VMware collector data is not required it can be stopped using the `-R log_level_decrease` option.

### Troubleshooting

- In case of unavailable metrics, please make sure if they are not made unavailable or turned off by default in recent VMware vSphere versions or if some limits are not placed on performance-metric database queries. See ZBX-12094 for additional details.

- In case of `config.vpxd.stats.maxQueryMetrics` is invalid or exceeds the maximum number of characters permitted** error, add a `config.vpxd.stats.maxQueryMetrics` parameter to the vCenter Server settings. The value of this parameter should be the same as the value of `maxQuerysize` in VMware’s web.xml. See this VMware knowledge base article for details.
VMware monitoring item keys

This page provides details on the simple checks that can be used to monitor VMware environments. The metrics are grouped by the monitoring target.

**General service metrics**

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vmware.eventlog</code></td>
<td>VMware event log</td>
<td>Log</td>
<td><code>url</code> - VMware service URL, <code>mode</code> - all (default), skip - skip processing of older data</td>
<td>There must be only one <code>vmware.eventlog[]</code> item key per URL. See also: example of filtering VMware event log records.</td>
</tr>
<tr>
<td><code>vmware.fullname</code></td>
<td>VMware service full name</td>
<td>String</td>
<td><code>url</code> - VMware service URL</td>
<td></td>
</tr>
<tr>
<td><code>vmware.version</code></td>
<td>VMware service version</td>
<td>String</td>
<td><code>url</code> - VMware service URL</td>
<td></td>
</tr>
</tbody>
</table>

**Cluster**

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vmware.cl.perfcounter</code></td>
<td>VMware cluster performance counter metrics.</td>
<td>Integer</td>
<td><code>url</code> - VMware service URL, <code>id</code> - VMware cluster ID, <code>path</code> - performance counter path, <code>instance</code> - performance counter instance</td>
<td>id can be received from <code>vmware.cluster.discovery[]</code> as <code>{#CLUSTER.ID}</code></td>
</tr>
<tr>
<td><code>vmware.cluster.alarms.get</code></td>
<td>VMware cluster alarms data</td>
<td>JSON object</td>
<td><code>url</code> - VMware service URL, <code>id</code> - VMware cluster ID</td>
<td></td>
</tr>
<tr>
<td><code>vmware.cluster.discovery</code></td>
<td>Discovery of VMware clusters.</td>
<td>JSON object</td>
<td><code>url</code> - VMware service URL</td>
<td></td>
</tr>
<tr>
<td><code>vmware.cluster.property</code></td>
<td>VMware cluster property</td>
<td>String</td>
<td><code>url</code> - VMware service URL, <code>id</code> - VMware cluster ID, <code>prop</code> - property path</td>
<td></td>
</tr>
<tr>
<td><code>vmware.cluster.status</code></td>
<td>VMware cluster status</td>
<td>Integer</td>
<td><code>url</code> - VMware service URL, <code>name</code> - VMware cluster name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.cluster.tags.get</code></td>
<td>VMware cluster tags array</td>
<td>JSON object</td>
<td><code>url</code> - VMware service URL, <code>id</code> - VMware cluster ID</td>
<td></td>
</tr>
</tbody>
</table>
## Datastore

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vmware.datastore.alarms.get</code></td>
<td>JSON object</td>
<td>url - VMware service URL</td>
<td>uuid - VMware datastore name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.datastore.property</code></td>
<td>String</td>
<td>url - VMware service URL</td>
<td>uuid - VMware datastore name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.datastore.tags.get</code></td>
<td>JSON object</td>
<td>url - VMware service URL</td>
<td>uuid - VMware datastore name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.datastore.discovery</code></td>
<td>JSON object</td>
<td>url - VMware service URL</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>vmware.datastore.hv.list</code></td>
<td>String</td>
<td>url - VMware service URL</td>
<td>datastore - datastore name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.datastore.read</code></td>
<td>Integer</td>
<td>url - VMware service URL</td>
<td>datastore - datastore name</td>
<td>mode - latency (average value, default), maxlatency (maximum value)</td>
</tr>
<tr>
<td><code>vmware.datastore.size</code></td>
<td>Integer - for bytes, Float - for percentage</td>
<td>url - VMware service URL</td>
<td>datastore - datastore name</td>
<td>mode - possible values: total (default), free, pfree (free, percentage), uncommitted</td>
</tr>
<tr>
<td><code>vmware.datastore.write</code></td>
<td>Integer</td>
<td>url - VMware service URL</td>
<td>datastore - datastore name</td>
<td>mode - latency (average value, default), maxlatency (maximum value)</td>
</tr>
</tbody>
</table>

## Datacenter

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vmware.dc.alarms.get</code></td>
<td>JSON object</td>
<td>url - VMware service URL</td>
<td>id - VMware datacenter ID</td>
<td></td>
</tr>
<tr>
<td><code>vmware.dc.discovery</code></td>
<td>JSON object</td>
<td>url - VMware service URL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Key

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vmware.dc.tags.get[&lt;url&gt;,&lt;id&gt;]</code></td>
<td>VMware datacenter tags array.</td>
<td>JSON object</td>
<td>url - VMware service URL id - VMware datacenter ID</td>
<td></td>
</tr>
</tbody>
</table>

### vSphere Distributed Switch

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vmware.dvswitch.discovery[&lt;url&gt;]</code></td>
<td>Discovery of VMware vSphere Distributed Switches.</td>
<td>JSON object</td>
<td>url - VMware service URL</td>
<td></td>
</tr>
<tr>
<td><code>vmware.dvswitch.fetchports.get[&lt;url&gt;,&lt;uuid&gt;,&lt;filter&gt;,&lt;mode&gt;]</code></td>
<td>VMware vSphere Distributed Switch ports data.</td>
<td>JSON object</td>
<td>uuid - ID of the switch filter - single string with comma-separated criteria for selecting ports mode - state (all XML without &quot;config&quot; XML nodes, default), full</td>
<td>Parameter filter supports the criteria available in the VMware data object DistributedVirtualSwitchPortCriteria Example: <code>vmware.dvswitch.fetchports.get[{$VMWARE.URL},{$VMWARE.DVS.UUID},&quot;active:true,connected:false,host:host-18,inside:true,nsxPort:true,uplinkPort:false&quot;,state]</code></td>
</tr>
</tbody>
</table>

### Hypervisor

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vmware.hv.alarms.get[&lt;url&gt;,&lt;uuid&gt;]</code></td>
<td>VMware hypervisor alarms data.</td>
<td>JSON object</td>
<td>url - VMware service URL uuid - VMware hypervisor host name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.hv.cluster.name[&lt;url&gt;,&lt;uuid&gt;]</code></td>
<td>VMware hypervisor cluster name.</td>
<td>String</td>
<td>url - VMware service URL uuid - VMware hypervisor host name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.hv.connectionstate[&lt;url&gt;,&lt;uuid&gt;]</code></td>
<td>VMware hypervisor connection state.</td>
<td>String: connected notResponding</td>
<td>url - VMware service URL uuid - VMware hypervisor host name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.hv.cpu.usage[&lt;url&gt;,&lt;uuid&gt;]</code></td>
<td>VMware hypervisor processor usage (Hz).</td>
<td>Integer</td>
<td>url - VMware service URL uuid - VMware hypervisor host name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.hv.cpu.usage.perf[&lt;url&gt;,&lt;uuid&gt;]</code></td>
<td>VMware hypervisor processor usage as a percentage during the interval.</td>
<td>Float</td>
<td>url - VMware service URL uuid - VMware hypervisor host name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.hv.cpu.utilization[&lt;url&gt;,&lt;uuid&gt;]</code></td>
<td>VMware hypervisor processor usage as a percentage during the interval.</td>
<td>Float</td>
<td>url - VMware service URL uuid - VMware hypervisor host name</td>
<td></td>
</tr>
</tbody>
</table>
### Key

| VMware hypervisor processor usage as a percentage during the interval, depends on power management or HT. |

| VMware hypervisor datacenter name. |

| Discovery of VMware hypervisor datastores. |

| List of VMware hypervisor datastores. |

| Number of available datastore paths. |

| Average amount of time for a read operation from the datastore (milliseconds). |

| VMware hypervisor datastore space in bytes or in percentage from total. |

| Average amount of time for a write operation to the datastore (milliseconds). |

| Discovery of VMware hypervisors. |

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>url</code></td>
<td>VMware service URL</td>
</tr>
<tr>
<td><code>uuid</code></td>
<td>VMware hypervisor host name</td>
</tr>
</tbody>
</table>

#### `vmware.hv.datacenter.name`[
url>,<uuid>]

VMware hypervisor datacenter name.

#### `vmware.hv.datastore.discovery`[
url>,<uuid>]

Discovery of VMware hypervisor datastores.

#### `vmware.hv.datastore.list`[
url>,<uuid>]

List of VMware hypervisor datastores.

#### `vmware.hv.datastore.multipath`[
url>,<uuid>,<datastore>,<partitionid>]

Number of available datastore paths.

#### `vmware.hv.datastore.read`[
url>,<uuid>,<datastore>,<mode>]

Average amount of time for a read operation from the datastore (milliseconds).

#### `vmware.hv.datastore.size`[
url>,<uuid>,<datastore>,<mode>]

VMware datastore space in bytes or in percentage from total.

#### `vmware.hv.datastore.write`[
url>,<uuid>,<datastore>,<mode>]

Average amount of time for a write operation to the datastore (milliseconds).

#### `vmware.hv.discovery`[
url]

Discovery of VMware hypervisors.
<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmware.hv.fullname</td>
<td>String</td>
<td>VMware hypervisor full name.</td>
</tr>
<tr>
<td>vmware.hv.hw.cpu.freq</td>
<td>Integer</td>
<td>Processor frequency (Hz).</td>
</tr>
<tr>
<td>vmware.hv.hw.cpu.model</td>
<td>String</td>
<td>Processor model.</td>
</tr>
<tr>
<td>vmware.hv.hw.cpu.num</td>
<td>Integer</td>
<td>Number of processor cores on VMware hypervisor.</td>
</tr>
<tr>
<td>vmware.hv.hw.cpu.threads</td>
<td>Integer</td>
<td>Number of processor threads on VMware hypervisor.</td>
</tr>
<tr>
<td>vmware.hv.hw.memory</td>
<td>Integer</td>
<td>Total memory size (bytes).</td>
</tr>
<tr>
<td>vmware.hv.hw.model</td>
<td>String</td>
<td>VMware hypervisor model.</td>
</tr>
<tr>
<td>vmware.hv.hw.serialnumber</td>
<td>String</td>
<td>Serial number. This item works with vSphere API 6.7 and newer.</td>
</tr>
<tr>
<td>vmware.hv.hw.uuid</td>
<td>String</td>
<td>BIOS UUID.</td>
</tr>
<tr>
<td>vmware.hv.hw.vendor</td>
<td>String</td>
<td>Vendor name.</td>
</tr>
<tr>
<td>vmware.hv.maintenance</td>
<td>Integer</td>
<td>Maintenance status. Returns '0' - not in maintenance or '1' - in maintenance</td>
</tr>
<tr>
<td>vmware.hv.memory.size.ballooned</td>
<td>Integer</td>
<td>Memory size ballooned.</td>
</tr>
<tr>
<td>Key</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>vmware.hv.memory.used</strong></td>
<td>Integer</td>
<td>VMware hypervisor used memory size (bytes).</td>
</tr>
<tr>
<td><strong>vmware.hv.net.if.discovery</strong></td>
<td>JSON object</td>
<td>Discovery of VMware hypervisor network interfaces.</td>
</tr>
<tr>
<td><strong>vmware.hv.network.in</strong></td>
<td>Integer²</td>
<td>VMware hypervisor network input statistics (bytes per second).</td>
</tr>
<tr>
<td><strong>vmware.hv.network.out</strong></td>
<td>Integer²</td>
<td>VMware hypervisor network output statistics (bytes per second).</td>
</tr>
<tr>
<td><strong>vmware.hv.perfcounter</strong></td>
<td>Integer²</td>
<td>VMware hypervisor performance counter value.</td>
</tr>
<tr>
<td><strong>vmware.hv.property</strong></td>
<td>String</td>
<td>VMware hypervisor property.</td>
</tr>
<tr>
<td><strong>vmware.hv.power</strong></td>
<td>Integer</td>
<td>VMware hypervisor power usage (W).</td>
</tr>
<tr>
<td><strong>vmware.hv.sensor.health.state</strong></td>
<td>Integer</td>
<td>VMware hypervisor health state rollup sensor.</td>
</tr>
</tbody>
</table>

³ Returns 0, if network interface is down, otherwise speed value of the interface.
Key

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMware</td>
<td>JSON object</td>
</tr>
<tr>
<td>hypervisor</td>
<td>url - VMware service URL</td>
</tr>
<tr>
<td>HW vendor</td>
<td>uuid - VMware hypervisor host name</td>
</tr>
<tr>
<td>state sensors.</td>
<td></td>
</tr>
<tr>
<td>vmware.hv.status</td>
<td>VMware Integer: url - VMware service URL</td>
</tr>
<tr>
<td>hypervisor</td>
<td>uuid - VMware hypervisor host name</td>
</tr>
<tr>
<td>status.</td>
<td>0 - gray; 1 - green; 2 - yellow; 3 - red</td>
</tr>
<tr>
<td></td>
<td>Uses the host system overall status property.</td>
</tr>
<tr>
<td>vmware.hv.tags.get</td>
<td>VMware JSON object url - VMware service URL</td>
</tr>
<tr>
<td>hypervisor</td>
<td>uuid - VMware hypervisor host name</td>
</tr>
<tr>
<td>tags array.</td>
<td></td>
</tr>
<tr>
<td>vmware.hv.uptime</td>
<td>VMware Integer: url - VMware service URL</td>
</tr>
<tr>
<td>hypervisor</td>
<td>uuid - VMware hypervisor host name</td>
</tr>
<tr>
<td>uptime.</td>
<td></td>
</tr>
<tr>
<td>vmware.hv.version</td>
<td>VMware String: url - VMware service URL</td>
</tr>
<tr>
<td>hypervisor</td>
<td>uuid - VMware hypervisor host name</td>
</tr>
<tr>
<td>version.</td>
<td></td>
</tr>
<tr>
<td>vmware.hv.vm.num</td>
<td>VMware Integer: url - VMware service URL</td>
</tr>
<tr>
<td>hypervisor</td>
<td>uuid - VMware hypervisor host name</td>
</tr>
<tr>
<td>virtual machines on VMware hypervisor.</td>
<td></td>
</tr>
</tbody>
</table>

Resource pool

Key

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>vmware.rp.cpu.usage</td>
<td>CPU usage in hertz during the interval on VMware Resource Pool.</td>
<td>Integer</td>
<td>url - VMware service URL</td>
</tr>
<tr>
<td></td>
<td>VMware</td>
<td>Memory metrics of VMware resource pool.</td>
<td>Integer</td>
<td>url - VMware service URL</td>
</tr>
</tbody>
</table>
### Virtual machine

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>vmware.alarms.get[&lt;url&gt;]</code></td>
<td>VMware virtual center alarms data.</td>
<td>JSON object</td>
<td><code>url</code> - VMware service URL</td>
<td></td>
</tr>
<tr>
<td><code>vmware.vm.alarms.get[&lt;url&gt;,&lt;uuid&gt;]</code></td>
<td>VMware virtual machine alarms data.</td>
<td>JSON object</td>
<td><code>url</code> - VMware service URL, <code>uuid</code> - VMware virtual machine name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.vm.attribute[&lt;url&gt;,&lt;uuid&gt;,&lt;name&gt;]</code></td>
<td>VMware virtual machine custom attribute value.</td>
<td>String</td>
<td><code>url</code> - VMware service URL, <code>uuid</code> - VMware virtual machine host name, <code>name</code> - custom attribute name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.vm.cluster.name[&lt;url&gt;,&lt;uuid&gt;]</code></td>
<td>VMware virtual machine name.</td>
<td>String</td>
<td><code>url</code> - VMware service URL, <code>uuid</code> - VMware virtual machine host name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.vm.consolidationneeded[&lt;url&gt;,&lt;uuid&gt;]</code></td>
<td>VMware virtual machine disk requires consolidation.</td>
<td>String: true - consolidation is needed; false - consolidation is not needed.</td>
<td><code>url</code> - VMware service URL, <code>uuid</code> - VMware virtual machine host name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.vm.cpu.latency[&lt;url&gt;,&lt;uuid&gt;]</code></td>
<td>Percentage of time the virtual machine is unable to run because it is contending for access to the physical CPU(s).</td>
<td>Float</td>
<td><code>url</code> - VMware service URL, <code>uuid</code> - VMware virtual machine host name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.vm.cpu.num[&lt;url&gt;,&lt;uuid&gt;]</code></td>
<td>Number of processors on VMware virtual machine.</td>
<td>Integer</td>
<td><code>url</code> - VMware service URL, <code>uuid</code> - VMware virtual machine host name</td>
<td></td>
</tr>
<tr>
<td><code>vmware.vm.cpu.readiness[&lt;url&gt;,&lt;uuid&gt;,&lt;instance&gt;]</code></td>
<td>Percentage of time that the virtual machine was ready, but could not get scheduled to run on the physical CPU.</td>
<td>Float</td>
<td><code>url</code> - VMware service URL, <code>uuid</code> - VMware virtual machine host name, <code>instance</code> - CPU instance</td>
<td></td>
</tr>
</tbody>
</table>
vmware.vm.cpu.ready\[<url>,<uuid>\]
Time (in milliseconds) Integer² url - VMware service URL
that the virtual machine was ready, but could not get scheduled to run on the physical CPU. CPU ready time is dependent on the number of virtual machines on the host and their CPU loads (%).

vmware.vm.cpu.swapwait\[<url>,<uuid>,<instance>\]
Percentage of CPU time spent waiting for swap-in.

vmware.vm.cpu.usage\[<url>,<uuid>\]
VMware virtual machine processor usage (Hz).

vmware.vm.cpu.usage.perf\[<url>,<uuid>\]
VMware virtual machine processor usage as a percentage during the interval.

vmware.vm.datacenter.name\[<url>,<uuid>\]
VMware virtual machine datacenter name.

vmware.vm.discovery\[<url>\]
Discovery of VMware virtual machines.

vmware.vm.guest.memory.size.swapped\[<url>,<uuid>\]
Amount of guest physical memory that is swapped out to the swap space (KB).

vmware.vm.guest.osuptime\[<url>,<uuid>\]
<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total time elapsed since the last operating system boot-up (in seconds).</td>
<td>Integer</td>
<td>- VMware service URL</td>
</tr>
<tr>
<td>vmware.vm.hv.name[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>String</td>
<td>VMware virtual machine hypervisor name.</td>
</tr>
<tr>
<td>vmware.vm.memory.size[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>Integer</td>
<td>VMware virtual machine total memory size (bytes).</td>
</tr>
<tr>
<td>vmware.vm.memory.size.ballooned[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>Integer</td>
<td>VMware virtual machine ballooned memory size (bytes).</td>
</tr>
<tr>
<td>vmware.vm.memory.size.compressed[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>Integer</td>
<td>VMware virtual machine compressed memory size (bytes).</td>
</tr>
<tr>
<td>vmware.vm.memory.size.consumed[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>Integer</td>
<td>Amount of host physical memory consumed for backing up guest physical memory pages (KB).</td>
</tr>
<tr>
<td>vmware.vm.memory.size.private[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>Integer</td>
<td>VMware virtual machine private memory size (bytes).</td>
</tr>
<tr>
<td>vmware.vm.memory.size.shared[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>Integer</td>
<td>VMware virtual machine shared memory size (bytes).</td>
</tr>
<tr>
<td>vmware.vm.memory.size.swapped[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>Integer</td>
<td>- VMware service URL</td>
</tr>
<tr>
<td>Key</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VMware virtual machine swapped memory size (bytes).</td>
<td><strong>url</strong> - VMware service URL, <strong>uuid</strong> - VMware virtual machine host name</td>
<td></td>
</tr>
<tr>
<td><strong>vmware.vm.memory.size.usage.guest</strong>[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>Integer</td>
<td>VMware virtual machine guest memory usage (bytes).</td>
</tr>
<tr>
<td><strong>vmware.vm.memory.size.usage.host</strong>[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>Integer</td>
<td>VMware virtual machine host memory usage (bytes).</td>
</tr>
<tr>
<td><strong>vmware.vm.memory.usage</strong>[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>Percentage of host physical memory that has been consumed.</td>
<td></td>
</tr>
<tr>
<td><strong>vmware.vm.net.if.discovery</strong>[&lt;url&gt;,&lt;uuid&gt;]</td>
<td><strong>url</strong> - VMware service URL, <strong>uuid</strong> - VMware virtual machine host name</td>
<td></td>
</tr>
<tr>
<td><strong>vmware.vm.net.if.in</strong>[&lt;url&gt;,&lt;uuid&gt;,&lt;instance&gt;,&lt;mode&gt;]</td>
<td>Integer $^2$</td>
<td>VMware virtual machine network interface input statistics (bytes/packets per second).</td>
</tr>
<tr>
<td><strong>vmware.vm.net.if.out</strong>[&lt;url&gt;,&lt;uuid&gt;,&lt;instance&gt;,&lt;mode&gt;]</td>
<td>Integer $^2$</td>
<td>VMware virtual machine network interface output statistics (bytes/packets per second).</td>
</tr>
<tr>
<td><strong>vmware.vm.net.if.usage</strong>[&lt;url&gt;,&lt;uuid&gt;,&lt;instance&gt;]</td>
<td></td>
<td>VMware virtual machine network interface statistics (bytes/packets per second).</td>
</tr>
</tbody>
</table>
Key

| VMware | Integer | url - VMware service URL |
| virtual | | uuid - VMware virtual machine host name |
| machine | | instance - network interface instance |

VMware virtual machine network utilization (combined transmit-rates and receive-rates) during the interval (KBps).

vmware.vm.perfcounter[<url>,<uuid>,<path>,<instance>]

VMware virtual machine performance counter value.

vmware.vm.powerstate[<url>,<uuid>]

VMware virtual machine power state.

Integer:

0 - poweredOff;
1 - poweredOn;
2 - suspended

vmware.vm.property[<url>,<uuid>,<prop>]

VMware virtual machine property.

String

vmware.vm.snapshot.get[<url>,<uuid>]

VMware virtual machine snapshot state.

JSON object

vmware.vm.state[<url>,<uuid>]

VMware virtual machine state.

String:

notRunning
resetting
running
shuttingDown
standby
unknown

vmware.vm.storage.committed[<url>,<uuid>]

VMware virtual machine committed storage space (bytes).

Integer

vmware.vm.storage.readio[<url>,<uuid>,<instance>]

Average number of outstanding read requests to the virtual disk during the collection interval.

Integer

(1) path - performance counter path
(2) instance - performance counter instance.
<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmware.vm.storage.totalreadlatency[&lt;url&gt;,&lt;uuid&gt;,&lt;instance&gt;]</td>
<td>integer</td>
<td>The average time a read from the virtual disk takes (milliseconds).</td>
</tr>
<tr>
<td></td>
<td>url</td>
<td>- VMware service URL</td>
</tr>
<tr>
<td></td>
<td>uuid</td>
<td>- VMware virtual machine host name</td>
</tr>
<tr>
<td></td>
<td>instance</td>
<td>- disk device instance (mandatory)</td>
</tr>
<tr>
<td>vmware.vm.storage.totalwritelatency[&lt;url&gt;,&lt;uuid&gt;,&lt;instance&gt;]</td>
<td>integer</td>
<td>The average time a write to the virtual disk takes (milliseconds).</td>
</tr>
<tr>
<td></td>
<td>url</td>
<td>- VMware service URL</td>
</tr>
<tr>
<td></td>
<td>uuid</td>
<td>- VMware virtual machine host name</td>
</tr>
<tr>
<td></td>
<td>instance</td>
<td>- disk device instance (mandatory)</td>
</tr>
<tr>
<td>vmware.vm.storage.uncommitted[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>integer</td>
<td>VMware virtual machine uncommitted storage space (bytes).</td>
</tr>
<tr>
<td></td>
<td>url</td>
<td>- VMware service URL</td>
</tr>
<tr>
<td></td>
<td>uuid</td>
<td>- VMware virtual machine host name</td>
</tr>
<tr>
<td>vmware.vm.storage.unshared[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>integer</td>
<td>VMware virtual machine unshared storage space (bytes).</td>
</tr>
<tr>
<td></td>
<td>url</td>
<td>- VMware service URL</td>
</tr>
<tr>
<td></td>
<td>uuid</td>
<td>- VMware virtual machine host name</td>
</tr>
<tr>
<td>vmware.vm.storage.writeoio[&lt;url&gt;,&lt;uuid&gt;,&lt;instance&gt;]</td>
<td>integer</td>
<td>Average number of outstanding write requests to the virtual disk during the collection interval.</td>
</tr>
<tr>
<td></td>
<td>url</td>
<td>- VMware service URL</td>
</tr>
<tr>
<td></td>
<td>uuid</td>
<td>- VMware virtual machine host name</td>
</tr>
<tr>
<td></td>
<td>instance</td>
<td>- disk device instance (mandatory)</td>
</tr>
<tr>
<td>vmware.vm.tags.get[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>json object</td>
<td>VMware virtual machine tags array.</td>
</tr>
<tr>
<td></td>
<td>url</td>
<td>- VMware service URL</td>
</tr>
<tr>
<td></td>
<td>uuid</td>
<td>- VMware virtual machine host name</td>
</tr>
<tr>
<td>vmware.vm.tools[&lt;url&gt;,&lt;uuid&gt;,&lt;mode&gt;]</td>
<td>string</td>
<td>VMware virtual machine guest tools state.</td>
</tr>
<tr>
<td></td>
<td>url</td>
<td>- VMware service URL</td>
</tr>
<tr>
<td></td>
<td>uuid</td>
<td>- VMware virtual machine host name</td>
</tr>
<tr>
<td></td>
<td>mode</td>
<td>- version, status</td>
</tr>
<tr>
<td>vmware.vm.uptime[&lt;url&gt;,&lt;uuid&gt;]</td>
<td>integer</td>
<td>VMware virtual machine uptime (seconds).</td>
</tr>
<tr>
<td></td>
<td>url</td>
<td>- VMware service URL</td>
</tr>
<tr>
<td></td>
<td>uuid</td>
<td>- VMware virtual machine host name</td>
</tr>
</tbody>
</table>
### Key

| **vmware.vm.vfs.dev.discovery**<url>,<uuid> | Discovery of VMware virtual machine disk devices. |
| **vmware.vm.vfs.dev.read**<url>,<uuid>,<instance>,<mode> | VMware virtual machine disk device read statistics (bytes/operations per second). |
| **vmware.vm.vfs.dev.write**<url>,<uuid>,<instance>,<mode> | VMware virtual machine disk device write statistics (bytes/operations per second). |
| **vmware.vm.vfs.fs.discovery**<url>,<uuid> | Discovery of VMware virtual machine file systems. |
| **vmware.vm.vfs.fs.size**<url>,<uuid>,<fsname>,<mode> | VMware virtual machine file system statistics (bytes/percentages). |

#### JSON object

- **url**: VMware service URL
- **uuid**: VMware virtual machine host name
- **instance**: disk device instance
- **mode**: bps (default)/ops-

### Footnotes

1. The VMware performance counter path has the `group/counter[rollup]` format where:
   - **group**: the performance counter group, for example `cpu`
   - **counter**: the performance counter name, for example `usagemhz`
   - **rollup**: the performance counter rollup type, for example `average`

   So the above example would give the following counter path: `cpu/usagemhz[average]`

   The performance counter group descriptions, counter names and rollup types can be found in VMware documentation.

2. The value of these items is obtained from VMware performance counters and the `VMwarePerfFrequency` parameter is used to refresh their data in Zabbix VMware cache:
   - `vmware.hv.datastore.read`
   - `vmware.hv.datastore.write`
   - `vmware.hv.network.in`
   - `vmware.hv.network.out`
   - `vmware.hv.perfcounter`
   - `vmware.vm.cpu.ready`
   - `vmware.vm.net.if.in`
   - `vmware.vm.net.if.out`
   - `vmware.vm.perfcounter`
   - `vmware.vm.vfs.dev.read`
   - `vmware.vm.vfs.dev.write`

More info
See Virtual machine monitoring for detailed information how to configure Zabbix to monitor VMware environments.

### Virtual machine discovery key fields

The following table lists fields returned by virtual machine related discovery keys.

<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
<th>Field</th>
<th>Retrieved content</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmware.cluster.discovery</td>
<td>Performs cluster discovery.</td>
<td>{#CLUSTER.ID}</td>
<td>Cluster identifier.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{#CLUSTER.NAME}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;resource_pool&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;tags&quot;</td>
<td></td>
</tr>
<tr>
<td>vmware.datastore.discovery</td>
<td>Performs datastore discovery.</td>
<td>{#DATASTORE}</td>
<td>Datastore name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{#DATASTORE.EXTENT}</td>
<td>JSON object with an array of {nodeName:partitionId}.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{#DATASTORE.UUID}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;tags&quot;</td>
<td></td>
</tr>
<tr>
<td>vmware.dc.discovery</td>
<td>Performs datacenter discovery.</td>
<td>{#DATACENTER}</td>
<td>Datacenter name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{#DATACENTERID}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;tags&quot;</td>
<td></td>
</tr>
<tr>
<td>vmware.dvswitch.discovery</td>
<td>Performs vSphere distributed switches discovery.</td>
<td>{#DVS.NAME}</td>
<td>Switch name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{#DVS.UUID}</td>
<td></td>
</tr>
<tr>
<td>vmware.hv.discovery</td>
<td>Performs hypervisor discovery.</td>
<td>{#HV.UUID}</td>
<td>Unique hypervisor identifier.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{#HV.ID}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{#HV.NAME}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{#HV.NETNAME}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>{#HV.IP}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

524
### Item key

| #CLUSTER.NAME | Cluster name, might be empty. |
| #DATACENTER.NAME | Datacenter name. |
| #PARENT.NAME | Name of container that stores the hypervisor. Supported since Zabbix 4.0.3. |
| #PARENT.TYPE | Type of container in which the hypervisor is stored. The values could be Datacenter, Folder, ClusterComputeResource, VMware, where 'VMware' stands for unknown container type. Supported since Zabbix 4.0.3. |

"resource_pool" An array containing resource pool data, including resource group ID, tags array, resource pool path, number of virtual machines:

```
[{
  "rpid": "resource group id",
  "tags": [
    {}
  ],
  "rpath": "resource group path",
  "vm_count": 0
}
```

For tags array structure, see "tags" field.

"tags" An array containing tags with tag name, description and category:

```
[{
  "tag": "tag name",
  "tag_description": "tag description",
  "category": "tag category"
}
```

### vmware.hv.datastore.discovery

Performs hypervisor datastore discovery. Note that multiple hypervisors can use the same datastore.

| #DATASTORE | Datastore name. |
| #DATASTORE.UUID | Datastore identifier. |
| #MULTIPATH.COUNT | Registered number of datastore paths. |
| #MULTIPATH.PARTITION.COUNT | Number of available disk partitions. |

"tags" An array containing tags with tag name, description and category:

```
[{
  "tag": "tag name",
  "tag_description": "tag description",
  "category": "tag category"
}
```

### vmware.hv.net.if.discovery

Performs hypervisor network interfaces discovery.

| #IFNAME | Interface name. |
| #IFDRIVER | Interface driver. |
| #IFDUPLEX | Interface duplex settings. |
| #IFSPEED | Interface speed. |
| #IFMAC | Interface mac address. |

### vmware.vm.discovery

Performs virtual machine discovery.

| #VM.UUID | Unique virtual machine identifier. |
| #VM.ID | Virtual machine identifier (VirtualMachine managed object name). |
| #VM.NAME | Virtual machine name. |
| #HV.NAME | Hypervisor name. |
| #HV.UUID | Unique hypervisor identifier. |
| #HV.ID | Hypervisor identifier (HostSystem managed object name). |
| #CLUSTER.NAME | Cluster name, might be empty. |
| #DATACENTER.NAME | Datacenter name. |
| #DATASTORE.NAME | Datastore name. |
| #DATASTORE.UUID | Datastore identifier. |
| #VM.IP | Virtual machine IP address, might be empty. |
| #VM.DNS | Virtual machine DNS name, might be empty. |
| #VM.GUESTFAMILY | Guest virtual machine OS family, might be empty. |
| #VM.GUESTFULLNAME | Full guest virtual machine OS name, might be empty. |
| #VM.FOLDER | The chain of virtual machine parent folders, can be used as value for nested groups; folder names are combined with "/". Might be empty. |
### Item key

<table>
<thead>
<tr>
<th>Item key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{#VM.TOOLS.STATUS}</code></td>
<td>VMware virtual machine tools state.</td>
</tr>
<tr>
<td><code>{#VM.POWERSTATE}</code></td>
<td>VMware virtual machine power state (poweredOFF, poweredOn, or suspended).</td>
</tr>
<tr>
<td><code>{#VM.RPOOL.ID}</code></td>
<td>Resource pool identifier.</td>
</tr>
</tbody>
</table>
| `{#VM.RPOOL.PATH}`             | Full resource pool path excluding the "root" name "Resources". Folder names are combined with "/".
| `{#VM.SNAPSHOT.COUNT}`         | Number of VM snapshots.                                                     |
| "tags"                         | An array containing tags with tag name, description and category:           |
| {"tag":"tagname","tag_description":"tag description", "category":"tag category"} | An array of virtual machine custom attributes (if defined):                 |
| vm.customattribute = [[]]      |                                                                           |

#### vmware.vm.net.if.discovery

Performs virtual machine network interface discovery.

- `{#IFNAME}`: Network interface name.
- `{#IFMAC}`: Interface mac address.
- `{#IFCONNECTED}`: Interface connection status (0 - disconnected, 1 - connected).
- `{#IFTYPE}`: Interface type.
- `{#IFBACKINGDEVICENAME}`: Name of the backing device.
- `{#IFDVSWITCH.UUID}`: Unique vSphere Distributed Switch identifier.
- `{#IFDVSWITCH.PORTGROUP}`: Distributed port group.
- `{#IFDVSWITCH.PORT}`: vSphere Distributed Switch port.

#### vmware.vm.vfs.dev.discovery

Performs virtual machine disk device discovery.

#### vmware.vm.vfs.fs.discovery

Performs virtual machine file system discovery.

#### JSON examples for VMware items

**Overview**  This section provides additional information about JSON objects returned by various VMware items.

**vmware.*.alarms.get**  The items `vmware.alarms.get()`, `vmware.cluster.alarms.get()`, `vmware.datastore.alarms.get()`, `vmware.dc.alarms.get()`, `vmware.hv.alarms.get()`, `vmware.vm.alarms.get()` return JSON objects with the following structure (values are provided as an example):

```json
{
  "alarms": [
    {
      "name": "Host connection and power state",
      "system_name": "alarm.HostConnectionStateAlarm",
      "description": "Default alarm to monitor host connection and power state",
      "enabled": true,
      "key": "alarm-1.host-2013",
      "time": "2022-06-27T05:27:38.759976Z",
      "overall_status": "red",
      "acknowledged": false
    },
    {
      "name": "Host memory usage",
      "system_name": "alarm.HostMemoryUsageAlarm",
      "description": "Default alarm to monitor host memory usage",
```

526
"enabled": true,
"key": "alarm-4.host-1004",
"time": "2022-05-16T13:32:42.47863Z",
"overall_status": "yellow",
"acknowledged": false
},
{
// other alarms
}
]

vmware.*.tags.get  The items vmware.cluster.tags.get[], vmware.datastore.tags.get[], vmware.dc.tags.get[], vmware.hv.tags.get[], vmware.vm.tags.get[] return JSON objects with the following structure (values are provided as an example):

{
  "tags": [
    {
      "name": "Windows",
      "description": "tag for cat OS type",
      "category": "OS type"
    },
    {
      "name": "SQL Server",
      "description": "tag for cat application name",
      "category": "application name"
    },
    {
      // other tags
    }
  ]
}

vmware.dvswitch.fetchports.get  The item vmware.dvswitch.fetchports.get[] returns JSON objects with the following structure (values are provided as an example):

{  "FetchDVPortsResponse":
  "returnval": [
    {  "key": "0",
      "dvsUuid": "50 36 6a 24 25 c0 10 9e-05 4a f6 ea 4e 3d 09 88",
      "portgroupKey": "dvportgroup-2023",
      "proxyHost":
        {
          "@type": "HostSystem",
          "#text": "host-2021"
        },
      "connectee":
        {
          "connectedEntity":
          {
            "@type": "HostSystem",
            "#text": "host-2021"
          },
        "nicKey": "vmnic0",
        "type": "pnic"
      },
    "conflict": "false",
    "state":
    {  
    }
"runtimeInfo": {
  "linkUp": "true",
  "blocked": "false",
  "vlanIds": {
    "start": "0",
    "end": "4094"
  },
  "trunkingMode": "true",
  "linkPeer": "vmnic0",
  "macAddress": "00:00:00:00:00:00",
  "statusDetail": null,
  "vmDirectPathGen2Active": "false",
  "vmDirectPathGen2InactiveReasonOther": "portNptIncompatibleConnectee"
},
"stats": {
  "packetsInMulticast": "2385470",
  "packetsOutMulticast": "45",
  "bytesInMulticast": "309250248",
  "bytesOutMulticast": "5890",
  "packetsInUnicast": "155601537",
  "packetsOutUnicast": "113008658",
  "bytesInUnicast": "121609489384",
  "bytesOutUnicast": "47240279759",
  "packetsInBroadcast": "1040420",
  "packetsOutBroadcast": "7051",
  "bytesInBroadcast": "77339771",
  "bytesOutBroadcast": "430392",
  "packetsInDropped": "0",
  "packetsOutDropped": "0",
  "packetsInException": "0",
  "packetsOutException": "0"
},
"connectionCookie": "1702765133",
"lastStatusChange": "2022-03-25T14:01:11Z",
"hostLocalPort": "false"
},
//other keys
}
}

**vmware.hv.hw.sensors.get**  The item `vmware.hv.hw.sensors.get[]` returns JSON objects with the following structure (values are provided as an example):

```json
{
  "val": {
    "@type": "HostHardwareStatusInfo",
    "storageStatusInfo": [
      {
        "name": "Intel Corporation HD Graphics 630 #2",
        "status": {
          "label": "Unknown",
          "summary": "Cannot report on the current status of the physical element",
          "key": "Unknown"
        }
      }
    ]
  }
}
```
The item `vmware.hv.sensors.get[]` returns JSON objects with the following structure (values are provided as an example):

```
{
    "val": [
        {
            "@type": "ArrayOfHostNumericSensorInfo", "HostNumericSensorInfo": [
                {
                    "@type": "HostNumericSensorInfo",
                    "name": "System Board 1 PwrMeter Output --- Normal",
                    "healthState": {
                        "label": "Green",
                        "summary": "Sensor is operating under normal conditions",
                        "key": "green"
                    },
                    "currentReading": "10500",
                    "unitModifier": "-2",
                    "baseUnits": "Watts",
                    "sensorType": "other"
                },
                {
                    "@type": "HostNumericSensorInfo",
                    "name": "Power Supply 1 PS 1 Output --- Normal",
                    "healthState": {
                        "label": "Green",
                        "summary": "Sensor is operating under normal conditions",
                        "key": "green"
                    },
                    "currentReading": "10000",
                    "unitModifier": "-2",
                    "baseUnits": "Watts",
                    "sensorType": "power"
                }
            ]
        }
    ]
}
```

`vmware.vm.snapshot.get` If any snapshots exist, the item `vmware.snapshot.get[]` returns a JSON object with the following structure (values are provided as an example):

```
{
    "val": [
        {
            "@type": "ArrayOfVMSnapshotEntryInfo", "VMSnapshotEntryInfo": [
                {
                    "@type": "VMSnapshotEntryInfo",
                    "name": "Snapshot 1",
                    "description": "Description of snapshot 1",
                    "creationTime": "2023-01-15T12:00:00Z",
                    "status": {
                        "label": "Ready",
                        "summary": "Ready to use"
                    }
                }
            ]
        }
    ]
}
```
If no snapshot exists, the item `vmware.snapshot.get[]` returns a JSON object with empty values:

```json
{
  "snapshot": [],
  "count": 0,
  "latestdate": "null",
  "size": 0,
  "uniquesize": 0
}
```

### 11. Maintenance

**Overview**  You can define maintenance periods for host groups, hosts and specific triggers/services in Zabbix.

There are two maintenance types - with data collection and with no data collection.

During a maintenance "with data collection" triggers are processed as usual and events are created when required. However, problem escalations are paused for hosts/triggers in maintenance, if the Pause operations for suppressed problems option is checked in action configuration. In this case, escalation steps that may include sending notifications or remote commands will be ignored for as long as the maintenance period lasts. Note that problem recovery and update operations are not suppressed during maintenance, only escalations.

For example, if escalation steps are scheduled at 0, 30 and 60 minutes after a problem start, and there is a half-hour-long maintenance lasting from 10 minutes to 40 minutes after a real problem arises, steps two and three will be executed a half-hour later, or at 60 minutes and 90 minutes (providing the problem still exists). Similarly, if a problem arises during the maintenance, the escalation will start after the maintenance.

To receive problem notifications during the maintenance normally (without delay), you have to uncheck the Pause operations for suppressed problems option in action configuration.

If at least one host (used in the trigger expression) is not in maintenance mode, Zabbix will send a problem notification.
Zabbix server must be running during maintenance. Timer processes are responsible for switching host status to/from maintenance at 0 seconds of every minute. Note that when a host enters maintenance, Zabbix server timer processes will read all open problems to check if it is required to suppress those. This may have a performance impact if there are many open problems. Zabbix server will also read all open problems upon startup, even if there are no maintenances configured at the time.

A proxy will always collect data regardless of the maintenance type (including "no data" maintenance). The data is later ignored by the server if 'no data collection' is set.

When "no data" maintenance ends, triggers using nodata() function will not fire before the next check during the period they are checking.

If a log item is added while a host is in maintenance and the maintenance ends, only new logfile entries since the end of the maintenance will be gathered.

If a timestamped value is sent for a host that is in a "no data" maintenance type (e.g. using Zabbix sender) then this value will be dropped however it is possible to send a timestamped value in for an expired maintenance period and it will be accepted.

To ensure predictable behavior of recurring maintenance periods (daily, weekly, monthly), it is required to use a common time zone for all parts of Zabbix.

If maintenance period, hosts, groups or tags are changed by the user, the changes will only take effect after configuration cache synchronization.

**Configuration**

To configure a maintenance period:

- Go to: Configuration → Maintenance
- Click on Create maintenance period (or on the name of an existing maintenance period)
- Enter maintenance parameters in the form
All mandatory input fields are marked with a red asterisk.

### Parameter Description

**Name**
Name of the maintenance period.

**Maintenance type**
Two types of maintenance can be set:
- **With data collection** - data will be collected by the server during maintenance, triggers will be processed
- **No data collection** - data will not be collected by the server during maintenance

**Active since**
The date and time when executing maintenance periods becomes active.
Note: Setting this time alone does not activate a maintenance period; for that go to the Periods tab.

**Active till**
The date and time when executing maintenance periods stops being active.

**Periods**
This block allows you to define the exact days and hours when the maintenance takes place.
Clicking on **Add** opens a popup window with a flexible Maintenance period form where you can define maintenance schedule. See [Maintenance periods](#) for a detailed description.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host groups</td>
<td>Select host groups that the maintenance will be activated for. The maintenance will be activated for all hosts from the specified host group(s). This field is auto-complete, so starting to type in it will display a dropdown of all available host groups. Specifying a parent host group implicitly selects all nested host groups. Thus the maintenance will also be activated on hosts from nested groups.</td>
</tr>
<tr>
<td>Hosts</td>
<td>Select hosts that the maintenance will be activated for. This field is auto-complete, so starting to type in it will display a dropdown of all available hosts.</td>
</tr>
<tr>
<td>Tags</td>
<td>If maintenance tags are specified, maintenance for the selected hosts will be activated, but only problems with matching tags will be suppressed (i.e. no actions will be taken). In case of multiple tags, they are calculated as follows: And/Or - all tags must correspond; however tags with the same tag name are calculated by the Or condition. Or - enough if one tag corresponds. There are two ways of matching the tag value: Contains - case-sensitive substring match (tag value contains the entered string) Equals - case-sensitive string match (tag value equals the entered string) Tags can be specified only if With data collection mode is selected.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of maintenance period.</td>
</tr>
</tbody>
</table>

**Maintenance periods**

The maintenance period window is for scheduling time for a recurring or a one-time maintenance. The form is dynamic with available fields changing based on the Period type selected.

![Maintenance period](image)

<table>
<thead>
<tr>
<th>Period type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>One time only</td>
<td>Define the date and time, and the length of the maintenance period. Every day(s) - maintenance frequency: 1 (default) - every day, 2 - every two days, etc. At (hour:minute) - time of the day when maintenance starts Maintenance period length - for how long the maintenance will be active.</td>
</tr>
<tr>
<td>Daily</td>
<td>Every week(s) - maintenance frequency: 1 (default) - every day, 2 - every two days, etc. Day of week - on which day the maintenance should take place. At (hour:minute) - time of the day when maintenance starts Maintenance period length - for how long the maintenance will be active.</td>
</tr>
<tr>
<td>Weekly</td>
<td>Every week(s) - maintenance frequency: 1 (default) - every day, 2 - every two days, etc. Day of week - on which day the maintenance should take place. At (hour:minute) - time of the day when maintenance starts Maintenance period length - for how long the maintenance will be active.</td>
</tr>
<tr>
<td>Monthly</td>
<td>Month - select all months during which the regular maintenance is carried out. Date: Day of month - Select this option if the maintenance takes place on the same date each month (for example, every 1st day of the month). Then, select the required day in the new field that appears. Date: Day of week - Select this option if the maintenance takes place only on certain days (for example, every first Monday of the month). Then, in the drop-down select the required week of the month (first, second, third, fourth, or last) and mark the checkboxes for maintenance day(s). At (hour:minute) - time of the day when maintenance starts Maintenance period length - for how long the maintenance will be active.</td>
</tr>
</tbody>
</table>
When done, press Add to add the maintenance period to the Periods block.

Notes:

- **When Every day/Every week parameter is greater than 1, the starting day or week is the day/week that the Active since time falls on.** For example:
  - with Active since set to January 1st at 12:00 and a one-hour maintenance set for every two days at 11pm will result in the first maintenance period starting on January 1st at 11pm, while the second maintenance period will start on January 3rd at 11pm;
  - with the same Active since time and a one-hour maintenance set for every two days at 1am, the first maintenance period will start on January 3rd at 1am, while the second maintenance period will start on January 5th at 1am.

- **Daylight Saving Time (DST) changes do not affect how long the maintenance will be.** Let’s say we have a two-hour maintenance that usually starts at 1am and finishes at 3am:
  - If after one hour of maintenance (at 2am) a DST change happens and current time changes from 2:00 to 3:00, the maintenance will continue for one more hour till 4:00;
  - If after two hours of maintenance (at 3am) a DST change happens and current time changes from 3:00 to 2:00, the maintenance will stop because two hours have passed.
  - If a maintenance period is set to 1 day it usually starts at 12am and finishes at 12am the next day. Since Zabbix calculates days in hours, the actual period of the maintenance is 24 hours. If current time changes forward one hour, the maintenance will stop at 1am the next day. If current time changes back one hour, the maintenance will stop at 11pm that day. If a maintenance period starts during the hour, skipped by DST change: The maintenance will not start.

**Display** Displaying hosts in maintenance

An orange wrench icon next to the host name indicates that this host is in maintenance in:

- Monitoring → Dashboard
- Monitoring → Problems
- Inventory → Hosts → Host inventory details
- Configuration → Hosts (See ‘Status’ column)

Maintenance details are displayed when the mouse pointer is positioned over the icon.

Additionally, hosts in maintenance get an orange background in Monitoring → Maps.

**Displaying suppressed problems**

Normally problems for hosts in maintenance are suppressed, i.e. not displayed in the frontend. However, it is also possible to configure that suppressed problems are shown, by selecting the Show suppressed problems option in these locations:

- Monitoring → Dashboard (in Problem hosts, Problems, Problems by severity, Trigger overview widget configuration)
- Monitoring → Problems (in the filter)
- Monitoring → Maps (in map configuration)
- Global notifications (in user profile configuration)

When suppressed problems are displayed, the following icon is displayed: Suppressed until: 13:05. Rolling a mouse over the icon displays more details:
12. Regular expressions

Overview  Perl Compatible Regular Expressions (PCRE, PCRE2) are supported in Zabbix.

There are two ways of using regular expressions in Zabbix:

- manually entering a regular expression
- using a global regular expression created in Zabbix

Regular expressions  You may manually enter a regular expression in supported places. Note that the expression may not start with @ because that symbol is used in Zabbix for referencing global regular expressions.

It’s possible to run out of stack when using regular expressions. See the pcrestack man page for more information.

Note that in multi-line matching, the ^ and $ anchors match at the beginning/end of each line respectively, instead of the beginning/end of the entire string.

Global regular expressions  There is an advanced editor for creating and testing complex regular expressions in Zabbix frontend.

Once a regular expression has been created this way, it can be used in several places in the frontend by referring to its name, prefixed with @, for example, @mycustomregexp.

To create a global regular expression:

- Go to: Administration → General
- Select Regular expressions from the dropdown
- Click on New regular expression

The Expressions tab allows to set the regular expression name and add subexpressions.

All mandatory input fields are marked with a red asterisk.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Set the regular expression name. Any Unicode characters are allowed.</td>
</tr>
<tr>
<td>Expressions</td>
<td>Click on Add in the Expressions block to add a new subexpression.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Expression type</td>
<td>Select expression type:</td>
</tr>
<tr>
<td></td>
<td><strong>Character string included</strong> - match the substring</td>
</tr>
<tr>
<td></td>
<td><strong>Any character string included</strong> - match any substring from a delimited list. The delimited list includes a comma (,), a dot (.) or a forward slash (/).</td>
</tr>
<tr>
<td></td>
<td><strong>Character string not included</strong> - match any string except the substring</td>
</tr>
<tr>
<td></td>
<td><strong>Result is TRUE</strong> - match the regular expression</td>
</tr>
<tr>
<td></td>
<td><strong>Result is FALSE</strong> - do not match the regular expression</td>
</tr>
<tr>
<td>Expression Delimiter</td>
<td>Enter substring/regular expression.</td>
</tr>
<tr>
<td></td>
<td>A comma (,), a dot (.) or a forward slash (/) to separate text strings in a regular expression.</td>
</tr>
<tr>
<td>Case sensitive</td>
<td>This parameter is active only when “Any character string included” expression type is selected.</td>
</tr>
<tr>
<td></td>
<td>A checkbox to specify whether a regular expression is sensitive to capitalization of letters.</td>
</tr>
</tbody>
</table>

A forward slash (/) in the expression is treated literally, rather than a delimiter. This way it is possible to save expressions containing a slash, without errors.

A custom regular expression name in Zabbix may contain commas, spaces, etc. In those cases where that may lead to misinterpretation when referencing (for example, a comma in the parameter of an item key) the whole reference may be put in quotes like this: “@My custom regexp for purpose1, purpose2”. Regular expression names must not be quoted in other locations (for example, in LLD rule properties).

In the **Test** tab the regular expression and its subexpressions can be tested by providing a test string.

Results show the status of each subexpression and total custom expression status.

Total custom expression status is defined as Combined result. If several sub expressions are defined Zabbix uses AND logical operator to calculate Combined result. It means that if at least one Result is False Combined result has also False status.

**Default global regular expressions**  Zabbix comes with several global regular expression in its default dataset.
<table>
<thead>
<tr>
<th>Name</th>
<th>Expression Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>File systems for discovery</td>
<td>`- (btrfs</td>
</tr>
</tbody>
</table>
| Network interfaces for discovery | `- Software Loopback Interface` Strings starting with “Software Loopback Interface”.  

- `^-lo$` "lo"  
- `^((In)?[Ll]oop[Bb]ack[0-9._]*$` Strings that optionally start with "In", then have "L" or "l", then "oop", then "B" or "b", then "ack", which can be optionally followed by any number of digits, dots or underscores.  
- `^-NULL[0-9_.]*$` Strings starting with "NULL" optionally followed by any number of digits or dots.  
- `^-[Ll]oop[0-9_.]*$` Strings starting with "Lo" or "lo" and optionally followed by any number of digits or dots.  
- `^-[Ss]ystem$` Strings starting with "System" or "system" optionally followed by any number of digits or dots.  
- `^-Nu[0-9_.]*$` Strings starting with "Nu" optionally followed by any number of digits or dots.  

- `^(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space)$` "Physical memory" or "Virtual memory" or "Memory buffers" or "Cached memory" or "Swap space"  
- `^- (MMCSS|gupdate|SysmonLog|clr_optimization_v2.0.50727_32|clr_optimization_v4.0.30319_32)$` "MMCSS" or "gupdate" or "SysmonLog" or strings like "clr_optimization_v2.0.50727_32" and "clr_optimization_v4.0.30319_32" where instead of dots you can put any character except newline.  "automatic" or "automatic delayed"  

- `^- (automatic|automatic delayed)$` "automatic" or "automatic delayed"  

**Examples**  
**Example 1**  
Use of the following expression in low-level discovery to discover databases except a database with a specific name:  
`^TESTDATABASE$`  

<table>
<thead>
<tr>
<th>Test string</th>
<th>TESTDATABASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>Expression type</td>
</tr>
</tbody>
</table>
| Chosen Expression type: | "Result is FALSE". Doesn’t match name, containing string "TESTDATABASE".  

**Example with an inline regex modifier**  
Use of the following regular expression including an inline modifier (?i) to match the characters "error":  
`(?i)error`
Chosen Expression type: "Result is TRUE". Characters "error" are matched.

Another example with an inline regex modifier

Use of the following regular expression including multiple inline modifiers to match the characters after a specific line:

```
(?<=match (?i)everything(?-i) after this line\n)(?sx).*# we add s modifier to allow . match newline characters
```

Chosen Expression type: "Result is TRUE". Characters after a specific line are matched.

**g** modifier can’t be specified in line. The list of available modifiers can be found in *pcresyntax man page*. For more information about PCRE syntax please refer to *PCRE HTML documentation*.

### Regular expression support by location

<table>
<thead>
<tr>
<th>Location</th>
<th>Regular expression</th>
<th>Global regular expression</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Agent items</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eventlog[]</td>
<td>Yes</td>
<td>Yes</td>
<td>regexp, severity, source, eventid parameters</td>
</tr>
<tr>
<td>log[]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>log.count[]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>logrt[]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>logrt.count[]</td>
<td>Yes/No</td>
<td></td>
<td>regexp parameter supports both, <em>file_regexp</em> parameter</td>
</tr>
<tr>
<td>proc.cpu.util[]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>proc.mem[]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>proc.num[]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sensor[]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>system.hw.macaddr[]</td>
<td></td>
<td></td>
<td>device and sensor parameters on Linux 2.4</td>
</tr>
<tr>
<td>system.sw.packages[]</td>
<td></td>
<td></td>
<td>interface parameter</td>
</tr>
<tr>
<td>vfs.dir.count[]</td>
<td></td>
<td></td>
<td>package parameter</td>
</tr>
</tbody>
</table>

<p>|          |                    |                           | <strong>regex_incl, regex_excl, regex_excl_dir</strong> parameters                     |</p>
<table>
<thead>
<tr>
<th>Location</th>
<th>Regular expression</th>
<th>Global regular expression</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>vfs.dir.size[]</td>
<td></td>
<td>regex_incl, regex_excl, regex_excl_dir parameters</td>
<td></td>
</tr>
<tr>
<td>vfs.file.regex[]</td>
<td></td>
<td>regexp parameter</td>
<td></td>
</tr>
<tr>
<td>vfs.file.regmatch[]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>web.page.regex[]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SNMP**

<table>
<thead>
<tr>
<th>Item</th>
<th>Yes</th>
<th>No</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>snmptrap[]</td>
<td>Yes</td>
<td>Yes</td>
<td>regexp parameter</td>
</tr>
</tbody>
</table>

**Functions for triggers/calculated items**

<table>
<thead>
<tr>
<th>Function</th>
<th>Yes</th>
<th>No</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>count()</td>
<td>Yes</td>
<td>Yes</td>
<td>pattern parameter if operator parameter is regexp or iregexp</td>
</tr>
<tr>
<td>countunique()</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>find()</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>logeventid()</td>
<td>Yes</td>
<td>Yes</td>
<td>pattern parameter</td>
</tr>
<tr>
<td>logsource()</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Low-level discovery**

<table>
<thead>
<tr>
<th>Filters</th>
<th>Yes</th>
<th>No</th>
<th>Regular expression field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overrides</td>
<td>Yes</td>
<td>No</td>
<td>In matches, does not match options for Operation conditions</td>
</tr>
</tbody>
</table>

**Action conditions**

| Web monitoring | Yes | No | Variables with a regex: prefix |
| User macro context | Yes | No | Required string field |

**Macro functions**

| regsub() | Yes | No | pattern parameter |
| iregsub() | | | |

**Icon mapping**

| Value mapping | Yes | No | Expression field |
| Value field if mapping type is regexp | | | |
13. Problem acknowledgment

Overview Problem events in Zabbix can be acknowledged by users.

If a user gets notified about a problem event, they can go to Zabbix frontend, open the problem update popup window of that problem using one of the ways listed below and acknowledge the problem. When acknowledging, they can enter their comment for it, saying that they are working on it or whatever else they may feel like saying about it.

This way, if another system user spots the same problem, they immediately see if it has been acknowledged and the comments so far.

This way the workflow of resolving problems with more than one system user can take place in a coordinated way.

Acknowledgment status is also used when defining action operations. You can define, for example, that a notification is sent to a higher level manager only if an event is not acknowledged for some time.

To acknowledge events and comment on them, a user must have at least read permissions to the corresponding triggers. To change problem severity or close problem, a user must have read-write permissions to the corresponding triggers.

There are several ways to access the problem update popup window, which allows acknowledging a problem.

- You may select problems in Monitoring → Problems and then click on Mass update below the list
- You can click in the Ack column showing the acknowledgment status of problems in:
  - Monitoring → Dashboard (Problems and Problems by severity widgets)
  - Monitoring → Problems
  - Monitoring → Problems → Event details

The Ack column contains either a ‘Yes’ or a ‘No’ link, indicating an acknowledged or an unacknowledged problem respectively. Clicking on the links will take you to the problem update popup window.

- You can click on an unresolved problem cell in:
  - Monitoring → Dashboard (Trigger overview widget)

The popup menu contains an Acknowledge option that will take you to the problem update window.

Updating problems The problem update popup allows to:

- comment on the problem
- view comments and actions so far
- change problem severity
- suppress/unsuppress problem
- acknowledge/unacknowledge problem
- manually close problem
Update problem

Problem: Disk space is critically low (used > 90%)

Message

History

Scope

Change severity

Suppress

 Unsuspend

 Acknowledge

 Close problem

All mandatory input fields are marked with a red asterisk.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem</td>
<td>If only one problem is selected, the problem name is displayed. If several problems are selected, N problems selected is displayed.</td>
</tr>
<tr>
<td>Message</td>
<td>Enter text to comment on the problem (maximum 2048 characters).</td>
</tr>
<tr>
<td>History</td>
<td>Previous activities and comments on the problem are listed, along with the time and user details. For the meaning of icons used to denote user actions see the event detail page. Note that history is displayed if only one problem is selected for the update.</td>
</tr>
</tbody>
</table>
| Scope     | Define the scope of such actions as changing severity, acknowledging or manually closing problems:  
**Only selected problem** - will affect this event only  
**Selected and all other problems of related triggers** - in case of acknowledgment/closing problem, will affect this event and all other problems that are not acknowledged/closed so far. If the scope contains problems already acknowledged or closed, these problems will not be acknowledged/closed repeatedly. On the other hand, the number of message and severity change operations are not limited. |
| Change severity | Mark the checkbox and click on the severity button to update problem severity. The checkbox for changing severity is available if read-write permissions exist for at least one of the selected problems. Only those problems that are read-writable will be updated when clicking on Update. If read-write permissions exist for none of the selected triggers, the checkbox is disabled. |
| Suppress | Mark the checkbox to suppress the problem:  
**Indefinitely** - suppress indefinitely  
**Until** - suppress until a given time. Both absolute and relative time formats are supported, for example:  
now+1d - for one day from now (default)  
now/ today - until the end of the current week  
2022-05-28 12:00:00 - until absolute date/time  
Note that a simple period (e.g., 1d, 1w) is not supported. Availability of this option depends on the "Suppress problems" user role settings. See also: Problem suppression |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uns suppress</td>
<td>Mark the checkbox to unsuppress the problem. This checkbox is active only if at least one of the selected problems is suppressed. Availability of this option depends on the “Suppress problems” user role settings.</td>
</tr>
<tr>
<td>Acknowledge</td>
<td>Mark the checkbox to acknowledge the problem. This checkbox is available if there is at least one unacknowledged problem among the selected. It is not possible to add another acknowledgment for an already acknowledged problem (it is possible to add another comment though).</td>
</tr>
<tr>
<td>Unacknowledge</td>
<td>Mark the checkbox to unacknowledge the problem. This checkbox is available if there is at least one acknowledged problem among the selected.</td>
</tr>
<tr>
<td>Close problem</td>
<td>Mark the checkbox to manually close the selected problem(s). The checkbox for closing a problem is available if the Allow manual close option is checked in trigger configuration for at least one of the selected problems. Only those problems will be closed that are allowed to be closed when clicking on Update. If no problem is manually closeable, the checkbox is disabled. Already closed problems will not be closed repeatedly.</td>
</tr>
</tbody>
</table>

**Display** Based on acknowledgment information it is possible to configure how the problem count is displayed in the dashboard or maps. To do that, you have to make selections in the Problem display option, available in both map configuration and the Problems by severity dashboard widget. It is possible to display all problem count, unacknowledged problem count as separated from the total or unacknowledged problem count only.

Based on problem update information (acknowledgment, etc.), it is possible to configure update operations - send a message or execute remote commands.

1. **Problem suppression**

**Overview**

Problem suppression offers a way of temporarily hiding a problem that can be dealt with later. This is useful for cleaning up the problem list in order to give the highest priority to the most urgent issues. For example, sometimes an issue may arise on the weekend that is not urgent enough to be dealt with immediately, so it can be “snoozed” until Monday morning.

Problem suppression allows to hide a single problem, in contrast to problem suppression through host maintenance when all problems of the maintenance host are hidden.

Operations for trigger actions will be paused for suppressed problems the same way as it is done with host maintenance.

**Configuration**

A problem can be suppressed through the problem update window, where suppression is one of the problem update options along with commenting, changing severity, acknowledging, etc.

A problem may also be unsuppressed through the same problem update window.

**Display**

Once suppressed the problem is marked by a blinking suppression icon in the Info column, before being hidden.

The suppression icon is blinking while the suppression task is in the waiting list. Once the task manager has suppressed the problem, the icon will stop blinking. If the suppression icon keeps blinking for a long time, this may indicate a server problem, for example, if the server is down and the task manager cannot complete the task. The same logic applies to unsuppression. In the short period after the task is submitted and the server has not completed it, the unsuppression icon is blinking.

A suppressed problem may be both hidden or shown, depending on the problem filter/widget settings.

When shown in the problem list, a suppressed problem is marked by the suppression icon and suppression details are shown on mouseover:
14. Configuration export/import

Overview  Zabbix export/import functionality makes it possible to exchange various configuration entities between one Zabbix system and another.

Typical use cases for this functionality:

- share templates or network maps - Zabbix users may share their configuration parameters
- share web scenarios on share.zabbix.com - export a template with the web scenarios and upload to share.zabbix.com. Then others can download the template and import the file into Zabbix.
- integrate with third-party tools - universal YAML, XML and JSON formats make integration and data import/export possible with third party tools and applications

What can be exported/imported

Objects that can be exported/imported are:

- host groups (through Zabbix API only)
- template groups (through Zabbix API only)
- templates
- hosts
- network maps
- media types
- images

Export format

Data can be exported using the Zabbix web frontend or Zabbix API. Supported export formats are YAML, XML and JSON.

Details about export

- All supported elements are exported in one file.
- Host and template entities (items, triggers, graphs, discovery rules) that are inherited from linked templates are not exported.
- Any changes made to those entities on a host level (such as changed item interval, modified regular expression or added prototypes to the low-level discovery rule) will be lost when exporting; when importing, all entities from linked templates are re-created as on the original linked template.
- Entities created by low-level discovery and any entities depending on them are not exported. For example, a trigger created for an LLD-rule generated item will not be exported.

Details about import

- Import stops at the first error.
- When updating existing images during image import, “imagetype” field is ignored, i.e. it is impossible to change image type via import.
- When importing hosts/templates using the “Delete missing” option, host/template macros not present in the import file will be deleted too.
- Empty tags for items, triggers, graphs, host/template applications, discoveryRules, itemPrototypes, triggerPrototypes, graphPrototypes are meaningless i.e. it’s the same as if it was missing. Other tags, for example, item applications, are meaningful i.e. empty tag means no applications for item, missing tag means don’t update applications.
• Import supports YAML, XML and JSON, the import file must have a correct file extension: .yaml and .yml for YAML, .xml for XML and .json for JSON.
• See compatibility information about supported XML versions.

```yaml
zabbix_export:
  version: '6.2'
  date: '2020-04-22T06:20:11Z'
```

**YAML base format**

*zabbix_export*:  
Root node for Zabbix YAML export.  
version: '6.2'  
Export version.  
date: '2020-04-22T06:20:11Z'  
Date when export was created in ISO 8601 long format.  
Other nodes are dependent on exported objects.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
  <version>6.0</version>
  <date>2020-04-22T06:20:11Z</date>
</zabbix_export>
```

**XML format**

<?xml version="1.0" encoding="UTF-8"?>

Default header for XML documents.  
<zabbix_export>  
Root element for Zabbix XML export.  
<version>6.0</version>  
Export version.  
<date>2020-04-22T06:20:11Z</date>  
Date when export was created in ISO 8601 long format.  
Other tags are dependent on exported objects.

```
{
  "zabbix_export": {
    "version": "6.0",
    "date": "2020-04-22T06:20:11Z"
  }
}
```

**JSON format**

"zabbix_export":  
Root node for Zabbix JSON export.  
"version": "6.0"  
Export version.  
"date": "2020-04-22T06:20:11Z"  
Date when export was created in ISO 8601 long format.  
Other nodes are dependent on exported objects.
1 Template groups

In the frontend template groups can be exported only with template export. When a template is exported all groups it belongs to are exported with it automatically.

API allows to export template groups independently from templates.

Export format

```json
template_groups:
  -
    uuid: 36bff6c29af64692839d077febf7079
    name: 'Network devices'
```

Element tags

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>string</td>
<td>Unique identifier for this template group.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Group name.</td>
</tr>
</tbody>
</table>

2 Host groups

In the frontend host groups can be exported only with host export. When a host is exported all groups it belongs to are exported with it automatically.

API allows to export host groups independently from hosts.

Export format

```json
host_groups:
  -
    uuid: 6f6799aa69e844b4b3918f779f2abf08
    name: 'Zabbix servers'
```

Element tags

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>string</td>
<td>Unique identifier for this host group.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Group name.</td>
</tr>
</tbody>
</table>

3 Templates

Overview

Templates are exported with many related objects and object relations.

Template export contains:

- linked template groups
- linked host groups (if used in host prototype configuration)
- template data
- linkage to other templates
- linkage to template groups
- directly linked items
- directly linked triggers
- directly linked graphs
- directly linked dashboards
- directly linked discovery rules with all prototypes
- directly linked web scenarios
- value maps
Exporting

To export templates, do the following:

- Go to: Configuration → Templates
- Mark the checkboxes of the templates to export
- Click on Export below the list

Depending on the selected format, templates are exported to a local file with a default name:

- `zabbix_export_templates.yaml` - in YAML export (default option for export)
- `zabbix_export_templates.xml` - in XML export
- `zabbix_export_templates.json` - in JSON export

Importing

To import templates, do the following:

- Go to: Configuration → Templates
- Click on Import to the right
- Select the import file
- Mark the required options in import rules
- Click on Import
All mandatory input fields are marked with a red asterisk.

Import rules:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update existing</td>
<td>Existing elements will be updated with data taken from the import file. Otherwise, they will not be updated.</td>
</tr>
<tr>
<td>Create new</td>
<td>The import will add new elements using data from the import file. Otherwise, it will not add them.</td>
</tr>
<tr>
<td>Delete missing</td>
<td>The import will remove existing elements not present in the import file. Otherwise, it will not remove them. If Delete missing is marked for template linkage, existing template linkage not present in the import file will be removed from the template along with all entities inherited from the potentially unlinked templates (items, triggers, etc).</td>
</tr>
</tbody>
</table>

On the next screen, you will be able to view the content of a template being imported. If this is a new template all elements will be listed in green. If updating an existing template, new template elements are highlighted in green; removed template elements are highlighted in red; elements that have not changed are listed on a gray background.
The menu on the left can be used to navigate through the list of changes. Section Updated highlights all changes made to existing template elements. Section Added lists new template elements. The elements in each section are grouped by element type; press on the gray arrow down to expand or collapse the group of elements.
Review template changes, then press Import to perform template import. A success or failure message of the import will be displayed in the frontend.

Export format

Export format in YAML:

```yaml
zabbix_export:
  version: '6.2'
  date: '2021-08-31T12:40:55Z'
  template_groups:
  -
    uuid: a571c0d144b14fd4a87a9d9b2aa9fcd6
    name: Templates/Applications
  templates:
  -
    uuid: 56079badd056419383cc26e6a4fcc7e0
    template: VMware
    name: VMware
    description: |
      You can discuss this template or leave feedback on our forum https://www.zabbix.com/forum/zabbix-solution-center

Template tooling version used: 0.38
```
templates:
- name: 'VMware macros'
groups:
- name: Templates/Applications
items:
- uuid: 5ce209f4d94f460488a74a92a52d92b1
  name: 'VMware: Event log'
  type: SIMPLE
  key: 'vmware.eventlog[{{VMWARE.URL}},skip]'
  history: 7d
  trends: '0'
  value_type: LOG
  username: '{${VMWARE.USERNAME}}'
  password: '{${VMWARE.PASSWORD}}'
  description: 'Collect VMware event log. See also: https://www.zabbix.com/documentation/6.2/manual/config/items/preprocessing/examples#filtering_vmware_event_log_records'
  tags:
    - tag: Application
      value: VMware

- uuid: ee2edadb8ce943ef81d25dbbba8667a4
  name: 'VMware: Full name'
  type: SIMPLE
  key: 'vmware.fullname[{{VMWARE.URL}}]' delay: 1h
  history: 7d
  trends: '0'
  value_type: CHAR
  username: '{${VMWARE.USERNAME}}'
  password: '{${VMWARE.PASSWORD}}'
  description: 'VMware service full name.'
  preprocessing:
    - type: DISCARD_UNCHANGED_HEARTBEAT
      parameters:
      - 1d
  tags:
    - tag: Application
      value: VMware

- uuid: a0ec9145f2234fbea79a28c57ebdb44d
  name: 'VMware: Version'
  type: SIMPLE
  key: 'vmware.version[{{VMWARE.URL}}]' delay: 1h
  history: 7d
  trends: '0'
  value_type: CHAR
  username: '{${VMWARE.USERNAME}}'
  password: '{${VMWARE.PASSWORD}}'
  description: 'VMware service version.'
  preprocessing:
    - type: DISCARD_UNCHANGED_HEARTBEAT
      parameters:
      - 1d
  tags:
tag: Application
value: VMware
discovery_rules:
  - uuid: 16ffc933ce74cf28a66e9036a99782
    name: 'Discover VMware clusters'
    type: SIMPLE
    key: 'vmware.cluster.discovery[{$VMWARE.URL}]'
    delay: 1h
    username: '{$VMWARE.USERNAME}'
    password: '{$VMWARE.PASSWORD}'
    description: 'Discovery of clusters'
  item_prototypes:
    - uuid: 4611ff91d564a459dbc1d396e2e6c76
      name: 'VMware: Status of "#{CLUSTER.NAME}" cluster'
      type: SIMPLE
      key: 'vmware.cluster.status[{$VMWARE.URL},#{CLUSTER.NAME}]'
      history: 7d
      username: '{$VMWARE.USERNAME}'
      password: '{$VMWARE.PASSWORD}'
      description: 'VMware cluster status.'
      valuemap:
        name: 'VMware status'
        tags:
          - tag: Application
            value: VMware
    - uuid: 8fb6a45be074b0cb6df53758e2c6623
      name: 'Discover VMware datastores'
      type: SIMPLE
      key: 'vmware.datastore.discovery[{$VMWARE.URL}]'
      delay: 1h
      username: '{$VMWARE.USERNAME}'
      password: '{$VMWARE.PASSWORD}'
      item_prototypes:
        - uuid: 4b61838ba4c34e709b25081ae5b059b5
          name: 'VMware: Average read latency of the datastore {#DATASTORE}'
          type: SIMPLE
          key: 'vmware.datastore.read[{$VMWARE.URL},#{DATASTORE},latency]'
          history: 7d
          username: '{$VMWARE.USERNAME}'
          password: '{$VMWARE.PASSWORD}'
          description: 'Amount of time for a read operation from the datastore (milliseconds).'
          tags:
            - tag: Application
              value: VMware
        - uuid: 5355c401dc244bc588cc1d8767577c93
          name: 'VMware: Free space on datastore {#DATASTORE} (percentage)'
          type: SIMPLE
          key: 'vmware.datastore.size[{$VMWARE.URL},#{DATASTORE},pfree]'
          delay: 5m
          history: 7d
          value_type: FLOAT
          units: '%'
          username: '{$VMWARE.USERNAME}'
          password: '{$VMWARE.PASSWORD}'
          description: 'VMware datastore space in percentage from total.'
tags:
- tag: Application
  value: VMware

  uuid: 84f13c4fde2d4a17baaf0c8c1eb4f2c0
  name: 'VMware: Total size of datastore {#DATASTORE}
  type: SIMPLE
  key: 'vmware.datastore.size[{{VMWARE.URL}},{#DATASTORE}]'
  delay: 5m
  history: 7d
  units: B
  username: '{${VMWARE.USERNAME}'}
  password: '{${VMWARE.PASSWORD}'}
  description: 'VMware datastore space in bytes.'
  tags:
    - tag: Application
      value: VMware

  uuid: 540cd0fbc56c4b8ea19f2ff5839ce00d
  name: 'VMware: Average write latency of the datastore {#DATASTORE}
  type: SIMPLE
  key: 'vmware.datastore.write[{{VMWARE.URL}},{#DATASTORE},latency]'
  history: 7d
  username: '{${VMWARE.USERNAME}'}
  password: '{${VMWARE.PASSWORD}'}
  description: 'Amount of time for a write operation to the datastore (milliseconds).'
  tags:
    - tag: Application
      value: VMware

  uuid: a5bc075e89f248e7b41d8f5897a08
  name: 'Discover VMware hypervisors'
  type: SIMPLE
  key: 'vmware.hv.discovery[{{VMWARE.URL}]}'
  delay: 1h
  username: '{${VMWARE.USERNAME}'}
  password: '{${VMWARE.PASSWORD}'}
  description: 'Discovery of hypervisors.'
  host_prototypes:
    - uuid: 051a1469d4d045cbbf818f7d0452e
      host: '#{HV.UUID}'}
      name: '#{HV.NAME}'}
      group_links:
        - group:
            name: Templates/Applications
          group_prototypes:
            - name: '#{CLUSTER.NAME}'}
            - name: '#{DATACENTER.NAME}'}
          templates:
            - name: 'VMware Hypervisor'
          macros:
            - macro: '#{VMWARE.HV.UUID}'}
              value: '#{HV.UUID}'}
              description: 'UUID of hypervisor.'
custom_interfaces: 'YES'
interfaces:
  -
ip: '{#HV.IP}'

uuid: 9fd559f4e88c4677a1b874634dd686f5
name: 'Discover VMware VMs'
type: SIMPLE
key: 'vmware.vm.discovery[{{VMWARE.URL}}]'
delay: 1h
username: '{${VMWARE.USERNAME}}'
password: '{${VMWARE.PASSWORD}}'
description: 'Discovery of guest virtual machines.'
host_prototypes:
  -
    uuid: 23b9ae9d6f33414880db1cb107115810
    host: '{#VM.UUID}'
    name: '{#VM.NAME}'
    group_links:
      -
        group:
          name: Templates/Applications
          group_prototypes:
            -
              name: '{#CLUSTER.NAME} (vm)'
            -
              name: '{#DATACENTER.NAME}/{#VM.FOLDER} (vm)'
            -
              name: '{#HV.NAME}'
    templates:
      -
        name: 'VMware Guest'
    macros:
      -
        macro: '{${VMWARE.VM.UUID}}'
        value: '{#VM.UUID}'
        description: 'UUID of guest virtual machine.'
    custom_interfaces: 'YES'
    interfaces:
      -
ip: '{#VM.IP}'
valuemaps:
  -
    uuid: 3c59c22905054d42ac4de8b72fe5f270
    name: 'VMware status'
    mappings:
      -
        value: '0'
        newvalue: gray
      -
        value: '1'
        newvalue: green
      -
        value: '2'
        newvalue: yellow
      -
        value: '3'
        newvalue: red

Element tags
Element tag values are explained in the table below.
Template tags
<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>template_groups</td>
<td>x</td>
<td></td>
<td></td>
<td>Root element for template groups.</td>
</tr>
<tr>
<td>uuid</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Unique identifier for this template group.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Template group name.</td>
</tr>
<tr>
<td>host_groups</td>
<td>x</td>
<td></td>
<td></td>
<td>Root element for host groups that are used by host prototypes.</td>
</tr>
<tr>
<td>uuid</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Unique identifier for this host group.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Host group name.</td>
</tr>
<tr>
<td>templates</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for templates.</td>
</tr>
<tr>
<td>uuid</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Unique identifier for this template.</td>
</tr>
<tr>
<td>template</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Unique template name.</td>
</tr>
<tr>
<td>name</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Visible template name.</td>
</tr>
<tr>
<td>description</td>
<td>-</td>
<td>text</td>
<td></td>
<td>Template description.</td>
</tr>
<tr>
<td>groups</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for template groups.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Template group name.</td>
</tr>
<tr>
<td>templates</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for linked templates.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Template name.</td>
</tr>
<tr>
<td>tags</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for template tags.</td>
</tr>
<tr>
<td>tag</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Tag name.</td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Tag value.</td>
</tr>
<tr>
<td>macros</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for template user macros.</td>
</tr>
<tr>
<td>macro</td>
<td>x</td>
<td>string</td>
<td></td>
<td>User macro name.</td>
</tr>
<tr>
<td>type</td>
<td>-</td>
<td>string</td>
<td>0 - TEXT (default) 1 - SECRET_TEXT 2 - VAULT</td>
<td>Type of the macro.</td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>string</td>
<td></td>
<td>User macro value.</td>
</tr>
<tr>
<td>description</td>
<td>-</td>
<td>string</td>
<td></td>
<td>User macro description.</td>
</tr>
<tr>
<td>valuemap</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for template value maps.</td>
</tr>
<tr>
<td>uuid</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Unique identifier for this value map.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Value map name.</td>
</tr>
<tr>
<td>mapping</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for mappings.</td>
</tr>
<tr>
<td>value</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Value of a mapping.</td>
</tr>
<tr>
<td>newvalue</td>
<td>x</td>
<td>string</td>
<td></td>
<td>New value of a mapping.</td>
</tr>
</tbody>
</table>

Template item tags

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>items</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for items.</td>
</tr>
<tr>
<td>uuid</td>
<td>x</td>
<td>string</td>
<td>0 - ZABBIX_PASSIVE (default) 2 - TRAP 3 - SIMPLE 5 - INTERNAL 7 - ZABBIX_ACTIVE 10 - EXTERNAL 11 - ODBC 12 - IPMI 13 - SSH 14 - TELNET 15 - CALCULATED 16 - JMX 17 - SNMP_TRAP 18 - DEPENDENT 19 - HTTP_AGENT 20 - SNMP_AGENT 21 - ITEM_TYPE_SCRIPT</td>
<td>Unique identifier for the item. Item name. Item type.</td>
</tr>
<tr>
<td>Element</td>
<td>property</td>
<td>RequiredType</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>--------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>snmp_oid</td>
<td></td>
<td>string</td>
<td></td>
<td>SNMP object ID.</td>
</tr>
<tr>
<td>key</td>
<td></td>
<td>string</td>
<td></td>
<td>Item key.</td>
</tr>
<tr>
<td>delay</td>
<td></td>
<td>string</td>
<td>Default: 1m</td>
<td>Update interval of the item. Accepts seconds or a time unit with suffix (30s, 1m, 2h, 1d). Optionally one or more custom intervals can be specified either as flexible intervals or scheduling. Multiple intervals are separated by a semicolon. User macros may be used. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. Flexible intervals may be written as two macros separated by a forward slash (e.g. <code>${FLEX_INTERVAL}/${FLEX_PERIOD}</code>).</td>
</tr>
<tr>
<td>history</td>
<td></td>
<td>string</td>
<td>Default: 90d</td>
<td>A time unit of how long the history data should be stored. Time unit with suffix, user macro or LLD macro.</td>
</tr>
<tr>
<td>trends</td>
<td></td>
<td>string</td>
<td>Default: 365d</td>
<td>A time unit of how long the trends data should be stored. Time unit with suffix, user macro or LLD macro.</td>
</tr>
<tr>
<td>status</td>
<td></td>
<td>string</td>
<td>0 - ENABLED (default) 1 - DISABLED</td>
<td>Item status.</td>
</tr>
<tr>
<td>value_type</td>
<td></td>
<td>string</td>
<td>0 - FLOAT 1 - CHAR 2 - LOG 3 - UNSIGNED (default) 4 - TEXT</td>
<td>Received value type.</td>
</tr>
<tr>
<td>allowed_hosts</td>
<td></td>
<td>string</td>
<td></td>
<td>List of IP addresses (comma delimitated) of hosts allowed sending data for the item. Used by trapper and HTTP agent items.</td>
</tr>
<tr>
<td>units</td>
<td></td>
<td>string</td>
<td></td>
<td>Units of returned values (bps, B, etc). Additional parameters depending on the type of the item: executed script for Script, SSH and Telnet items; SQL query for database monitor items; formula for calculated items.</td>
</tr>
<tr>
<td>params</td>
<td></td>
<td>text</td>
<td></td>
<td>IPMI sensor. Used only by IPMI items. Authentication type.</td>
</tr>
<tr>
<td>authtype</td>
<td></td>
<td>string</td>
<td>Authentication type for SSH agent items: 0 - PASSWORD (default) 1 - PUBLIC_KEY Authentication type for HTTP agent items: 0 - NONE (default) 1 - BASIC 2 - NTLM</td>
<td>Authentication type. Used only by SSH and HTTP agent items.</td>
</tr>
<tr>
<td>Element</td>
<td>property</td>
<td>RequiredType</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>--------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>username</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. Required by SSH and Telnet items. When used by JMX agent, password should also be specified together with the username or both properties should be left blank. When used by JMX agent, password should also be specified together with the username or both properties should be left blank.</td>
</tr>
<tr>
<td>password</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items.</td>
</tr>
<tr>
<td>publickey</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Name of the public key file. Required for SSH agent items. Name of the private key file.</td>
</tr>
<tr>
<td>privatekey</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Required for SSH agent items. Name of the private key file.</td>
</tr>
<tr>
<td>port</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Custom port monitored by the item. Can contain user macros.</td>
</tr>
<tr>
<td>description</td>
<td>-</td>
<td>text</td>
<td>0 - NONE</td>
<td>Used only by SNMP items. Item description.</td>
</tr>
<tr>
<td>inventory_link</td>
<td>-</td>
<td>string</td>
<td>Capitalized host inventory field name. Host inventory field that is populated by the item. For example: 4 - ALIAS 6 - OS_FULL 14 - HARDWARE etc.</td>
<td>Refer to the host inventory page for a list of supported host inventory fields and their IDs.</td>
</tr>
<tr>
<td>logtimefmt</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Format of the time in log entries. Used only by log items. JMX endpoint.</td>
</tr>
<tr>
<td>jmx_endpoint</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Used only by JMX agent items. URL string.</td>
</tr>
<tr>
<td>url</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Required only for HTTP agent items. Allow to populate value as in a trapper item. Follow HTTP response redirects while pooling data.</td>
</tr>
<tr>
<td>allow_traps</td>
<td>-</td>
<td>string</td>
<td>0 - NO (default) 1 - YES</td>
<td>Used only by HTTP agent items.</td>
</tr>
<tr>
<td>follow_redirects</td>
<td>string</td>
<td>0 - NO 1 - YES (default)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>headers</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for HTTP(S) request headers, where header name is used as key and header value as value. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Header name.</td>
</tr>
<tr>
<td>value</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Header value.</td>
</tr>
<tr>
<td>http_proxy</td>
<td>-</td>
<td>string</td>
<td></td>
<td>HTTP(S) proxy connection string. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>Element</td>
<td>Property</td>
<td>Required Type</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>---------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>output_format-</td>
<td>string</td>
<td>0 - RAW (default) 1 - JSON</td>
<td>How to process response.</td>
<td></td>
</tr>
<tr>
<td>post_type</td>
<td>string</td>
<td>0 - RAW (default) 2 - JSON 3 - XML</td>
<td>Type of post data body.</td>
<td></td>
</tr>
<tr>
<td>posts</td>
<td>string</td>
<td>Used only by HTTP agent items. HTTPS request body data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>query_fields</td>
<td>-</td>
<td>Used only by HTTP agent items. Root element for query parameters.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Parameter name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>Parameter value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>request_method</td>
<td>string</td>
<td>0 - GET (default) 1 - POST 2 - PUT 3 - HEAD.</td>
<td>Request method.</td>
<td></td>
</tr>
<tr>
<td>retrieve_mode-</td>
<td>string</td>
<td>0 - BODY (default) 1 - HEADERS 2 - BOTH.</td>
<td>What part of response should be stored.</td>
<td></td>
</tr>
<tr>
<td>ssl_cert_file</td>
<td>string</td>
<td>Used only by HTTP agent items. Public SSL Key file path.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssl_key_file</td>
<td>string</td>
<td>Used only by HTTP agent items. Private SSL Key file path.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssl_key_password</td>
<td>string</td>
<td>Used only by HTTP agent items. Password for SSL Key file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>status_codes</td>
<td>string</td>
<td>Ranges of required HTTP status codes separated by commas. Supports user macros. Example: 200,200-{$M},{$M},200-400.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>timeout</td>
<td>-</td>
<td>Item data polling request timeout. Supports user macros.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>verify_host</td>
<td>string</td>
<td>0 - NO (default) 1 - YES.</td>
<td>Validate if host name in URL is in Common Name field or a Subject Alternate Name field of host certificate.</td>
<td></td>
</tr>
<tr>
<td>verify_peer</td>
<td>string</td>
<td>0 - NO (default) 1 - YES.</td>
<td>Validate if host certificate is authentic.</td>
<td></td>
</tr>
<tr>
<td>parameters</td>
<td>-</td>
<td>Used only by HTTP agent items. Root element for user-defined parameters.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Used only by Script items. Parameter name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>Used only by Script items. Parameter value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value map</td>
<td>-</td>
<td>Used only by Script items. Value map.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the value map to use for the item.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>preprocessing</td>
<td>step</td>
<td>Used only by Script items. Individual item value preprocessing step.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Element property</td>
<td>RequiredType</td>
<td>Range</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>-------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>type x</td>
<td>string</td>
<td>1-MULTIPLIER RTRIM LTRIM TRIM REGEX BOOL_TO_DECIMAL OCTAL_TO_DECIMAL HEX_TO_DECIMAL SIMPLE_CHANGE CHANGE_PER_SECOND MATCHES_REGEX NOT_MATCHES_REGEX CHECK_JSON_ERROR CHECK_XML_ERROR CHECK_REGEX_ERROR DISCARD_UNCHANGED DISCARD_UNCHANGED_HEARTBEAT JAVASCRIPT PROMETHEUS_PATTERN PROMETHEUS_TO_JSON CSV_TO_JSON STR_REPLACE CHECK_NOT_SUPPORTED</td>
<td>Type of the item value preprocessing step.</td>
<td></td>
</tr>
<tr>
<td>parameters</td>
<td>-</td>
<td></td>
<td>Root element for parameters of the item value preprocessing step.</td>
<td></td>
</tr>
<tr>
<td>parameter x</td>
<td>string</td>
<td>0-ORIGINAL_ERROR DISCARD_VALUE CUSTOM_VALUE CUSTOM_ERROR</td>
<td>Individual parameter of the item value preprocessing step.</td>
<td></td>
</tr>
<tr>
<td>error_handler</td>
<td>string</td>
<td>0-ORIGINAL_ERROR (default) 1-DISCARD_VALUE 2-CUSTOM_VALUE 3-CUSTOM_ERROR</td>
<td>Action type used in case of preprocessing step failure.</td>
<td></td>
</tr>
<tr>
<td>error_handler_params</td>
<td>string</td>
<td></td>
<td>Error handler parameters used with ‘error_handler’.</td>
<td></td>
</tr>
<tr>
<td>master_item</td>
<td>-</td>
<td></td>
<td>Individual item master item.</td>
<td></td>
</tr>
</tbody>
</table>

Required by dependent items.
<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>Required Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>key</td>
<td>x</td>
<td>string</td>
<td>Dependent item master item key value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Recursion up to 3 dependent items and maximum count of dependent items equal to 29999 are allowed.</td>
</tr>
<tr>
<td>triggers</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for simple triggers.</td>
</tr>
<tr>
<td>tag</td>
<td>tag</td>
<td>x</td>
<td>string</td>
<td>Tag name.</td>
</tr>
<tr>
<td>value</td>
<td>value</td>
<td>-</td>
<td>string</td>
<td>Tag value.</td>
</tr>
</tbody>
</table>

Template low-level discovery rule tags

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>Required Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>type</td>
<td>-</td>
<td>string</td>
<td>Item type.</td>
</tr>
<tr>
<td>lifetime</td>
<td>lifetime</td>
<td>-</td>
<td>string</td>
<td>Time period after which items that are no longer discovered will be deleted. Seconds, time unit with suffix or user macro.</td>
</tr>
<tr>
<td>filter</td>
<td></td>
<td></td>
<td></td>
<td>Individual filter.</td>
</tr>
<tr>
<td>Element property</td>
<td>RequiredType</td>
<td>Range</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>-------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>evaltype</td>
<td>string</td>
<td>0-AND OR (default) 1-AND 2-OR 3-FORMULA</td>
<td>Logic to use for checking low-level discovery rule filter conditions.</td>
<td></td>
</tr>
<tr>
<td>formula</td>
<td>string</td>
<td></td>
<td>Custom calculation formula for filter conditions.</td>
<td></td>
</tr>
<tr>
<td>conditions</td>
<td></td>
<td></td>
<td>Root element for filter conditions.</td>
<td></td>
</tr>
<tr>
<td>macro</td>
<td>string</td>
<td></td>
<td>Low-level discovery macro name.</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td></td>
<td>Filter value: regular expression or global regular expression.</td>
<td></td>
</tr>
<tr>
<td>operator</td>
<td>string</td>
<td>8-MATCHES_REGEX (default) 9-NOT_MATCHES_REGEX</td>
<td>Condition operator.</td>
<td></td>
</tr>
<tr>
<td>formulaid</td>
<td>character</td>
<td></td>
<td>Arbitrary unique ID that is used to reference a condition from the custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.</td>
<td></td>
</tr>
<tr>
<td>lld_macro_paths</td>
<td></td>
<td></td>
<td>Root element for LLD macro paths.</td>
<td></td>
</tr>
<tr>
<td>lld_macro</td>
<td>string</td>
<td></td>
<td>Low-level discovery macro name.</td>
<td></td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td></td>
<td>Selector for value which will be assigned to the corresponding macro.</td>
<td></td>
</tr>
<tr>
<td>preprocessing</td>
<td></td>
<td></td>
<td>LLD rule value preprocessing.</td>
<td></td>
</tr>
<tr>
<td>step</td>
<td></td>
<td></td>
<td>Individual LLD rule value preprocessing step.</td>
<td></td>
</tr>
</tbody>
</table>

For most of the element tag values, see element tag values for a template item value preprocessing. Only the tags that are specific to template low-level discovery value preprocessing, are described below.
<table>
<thead>
<tr>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>x</td>
<td>string</td>
<td>Type of the item value preprocessing step.</td>
</tr>
<tr>
<td>trigger_prototypes</td>
<td>-</td>
<td></td>
<td>Root element for trigger prototypes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For trigger prototype element tag values, see regular template trigger tags.</td>
</tr>
<tr>
<td>graph_prototypes</td>
<td>-</td>
<td></td>
<td>Root element for graph prototypes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For graph prototype element tag values, see regular template graph tags.</td>
</tr>
<tr>
<td>host_prototypes</td>
<td>-</td>
<td></td>
<td>Root element for host prototypes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For host prototype element tag values, see regular host tags.</td>
</tr>
<tr>
<td>item_prototypes</td>
<td>-</td>
<td></td>
<td>Root element for item prototypes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For item prototype element tag values, see regular template item tags.</td>
</tr>
<tr>
<td>master_item</td>
<td>-</td>
<td></td>
<td>Individual item prototype master item/item prototype data.</td>
</tr>
<tr>
<td>key</td>
<td>x</td>
<td>string</td>
<td>Dependent item prototype master item/item prototype key value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Required for a dependent item.</td>
</tr>
</tbody>
</table>

Template trigger tags
<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>Required Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>triggers</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for triggers.</td>
</tr>
<tr>
<td></td>
<td>uuid</td>
<td>x</td>
<td>string</td>
<td>Unique identifier for this trigger.</td>
</tr>
<tr>
<td></td>
<td>expression</td>
<td>x</td>
<td>string</td>
<td>Trigger expression.</td>
</tr>
<tr>
<td></td>
<td>recovery_mode</td>
<td>string</td>
<td>0 - EXPRESSION (default) 1 - RECOVERY_EXPRESSION 2 - NONE</td>
<td>Basis for generating OK events.</td>
</tr>
<tr>
<td>recovery_expression</td>
<td>string</td>
<td></td>
<td></td>
<td>Trigger recovery expression.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Trigger name.</td>
</tr>
<tr>
<td>correlation_mode</td>
<td>string</td>
<td>0 - DISABLED (default) 1 - TAG_VALUE</td>
<td>Correlation mode (no event correlation or event correlation by tag).</td>
<td></td>
</tr>
<tr>
<td>correlation_tag</td>
<td>string</td>
<td></td>
<td></td>
<td>The tag name to be used for event correlation.</td>
</tr>
<tr>
<td>url</td>
<td>-</td>
<td>string</td>
<td></td>
<td>URL associated with the trigger.</td>
</tr>
<tr>
<td>status</td>
<td>-</td>
<td>string</td>
<td>0 - ENABLED (default) 1 - DISABLED</td>
<td>Trigger status.</td>
</tr>
<tr>
<td>priority</td>
<td>-</td>
<td>string</td>
<td>0 - NOT_CLASSIFIED (default) 1 - INFO 2 - WARNING 3 - AVERAGE 4 - HIGH 5 - DISASTER</td>
<td>Trigger severity.</td>
</tr>
<tr>
<td>description</td>
<td>-</td>
<td>text</td>
<td></td>
<td>Trigger description.</td>
</tr>
<tr>
<td>type</td>
<td>-</td>
<td>string</td>
<td>0 - SINGLE (default) 1 - MULTIPLE</td>
<td>Event generation type (single problem event or multiple problem events).</td>
</tr>
<tr>
<td>manual_close</td>
<td>-</td>
<td>string</td>
<td>0 - NO (default) 1 - YES</td>
<td>Manual closing of problem events.</td>
</tr>
<tr>
<td>dependencies</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for dependencies.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Dependency trigger name.</td>
</tr>
<tr>
<td>expression</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Dependency trigger expression.</td>
</tr>
<tr>
<td>recovery_expression</td>
<td>string</td>
<td></td>
<td></td>
<td>Dependency trigger recovery expression.</td>
</tr>
<tr>
<td>tags</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for trigger tags.</td>
</tr>
<tr>
<td>tag</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Tag name.</td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Tag value.</td>
</tr>
</tbody>
</table>

Template graph tags

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>Required Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphs</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for graphs.</td>
</tr>
<tr>
<td></td>
<td>uuid</td>
<td>x</td>
<td>string</td>
<td>Unique identifier for this graph.</td>
</tr>
<tr>
<td></td>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Graph name.</td>
</tr>
<tr>
<td></td>
<td>width</td>
<td>-</td>
<td>integer</td>
<td>20-65535 (default: 900)</td>
</tr>
<tr>
<td></td>
<td>height</td>
<td>-</td>
<td>integer</td>
<td>20-65535 (default: 200)</td>
</tr>
<tr>
<td></td>
<td>yaxismin</td>
<td>-</td>
<td>double</td>
<td>Default: 0</td>
</tr>
<tr>
<td></td>
<td>yaxismax</td>
<td>-</td>
<td>double</td>
<td>Default: 0</td>
</tr>
<tr>
<td></td>
<td>show_work_period</td>
<td>string</td>
<td>0 - NO 1 - YES (default)</td>
<td>Highlight non-working hours.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Used by normal and stacked graphs.</td>
</tr>
<tr>
<td>Element property</td>
<td>RequiredType</td>
<td>Range</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>show_triggers</td>
<td>string</td>
<td>0-NO 1-YES (default)</td>
<td>Display simple trigger values as a line. Used by normal and stacked graphs.</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>0-NORMAL (default) 1-STACKED 2-PIE 3-EXPLODED</td>
<td>Graph type.</td>
<td></td>
</tr>
<tr>
<td>show_legend</td>
<td>string</td>
<td>0-NO 1-YES (default)</td>
<td>Display graph legend.</td>
<td></td>
</tr>
<tr>
<td>show_3d</td>
<td>string</td>
<td>0-NO (default) 1-YES</td>
<td>Enable 3D style. Used by pie and exploded pie graphs. Show the percentile line for left axis.</td>
<td></td>
</tr>
<tr>
<td>percent_left</td>
<td>double</td>
<td>Default:0</td>
<td>Used only for normal graphs. Show the percentile line for left axis.</td>
<td></td>
</tr>
<tr>
<td>percent_right</td>
<td>double</td>
<td>Default:0</td>
<td>Used only for normal graphs. Show the percentile line for right axis.</td>
<td></td>
</tr>
<tr>
<td>ymin_type_1</td>
<td>string</td>
<td>0-CALCULATED (default) 1-FIXED 2-ITEM</td>
<td>Minimum value of Y axis. Used by normal and stacked graphs.</td>
<td></td>
</tr>
<tr>
<td>ymax_type_1</td>
<td>string</td>
<td>0-CALCULATED (default) 1-FIXED 2-ITEM</td>
<td>Maximum value of Y axis. Used by normal and stacked graphs.</td>
<td></td>
</tr>
<tr>
<td>ymin_item_1</td>
<td>-</td>
<td></td>
<td>Individual item details. Required if 'ymin_type_1' is ITEM.</td>
<td></td>
</tr>
<tr>
<td>host x</td>
<td>string</td>
<td></td>
<td>Item host.</td>
<td></td>
</tr>
<tr>
<td>key x</td>
<td>string</td>
<td></td>
<td>Item key.</td>
<td></td>
</tr>
<tr>
<td>ymax_item_1</td>
<td>-</td>
<td></td>
<td>Individual item details. Required if 'ymax_type_1' is ITEM.</td>
<td></td>
</tr>
<tr>
<td>host x</td>
<td>string</td>
<td></td>
<td>Item host.</td>
<td></td>
</tr>
<tr>
<td>key x</td>
<td>string</td>
<td></td>
<td>Item key.</td>
<td></td>
</tr>
<tr>
<td>graph_items x</td>
<td>string</td>
<td></td>
<td>Root element for graph items.</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>integer</td>
<td></td>
<td>Draw order. The smaller value is drawn first. Can be used to draw lines or regions behind (or in front of) another.</td>
<td></td>
</tr>
<tr>
<td>drawtype x</td>
<td>string</td>
<td>0-SINGLE_LINE (default) 1-FILLED_REGION 2-BOLD_LINE 3-DOTTED_LINE 4-DASHED_LINE 5-GRADIENT_LINE</td>
<td>Draw style of the graph item. Used only by normal graphs.</td>
<td></td>
</tr>
<tr>
<td>color x</td>
<td>string</td>
<td></td>
<td>Element color (6 symbols, hex).</td>
<td></td>
</tr>
<tr>
<td>yaxisside x</td>
<td>string</td>
<td>0-LEFT (default) 1-RIGHT</td>
<td>Side of the graph where the graph item’s Y scale will be drawn. Used by normal and stacked graphs.</td>
<td></td>
</tr>
<tr>
<td>Element</td>
<td>Element property</td>
<td>Required</td>
<td>Type</td>
<td>Range</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------</td>
<td>----------</td>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td>calc_fnc</td>
<td></td>
<td>string</td>
<td>1 - MIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 - AVG (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4 - MAX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7 - ALL (minimum, average and maximum; used only by simple graphs)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9 - LAST (used only by pie and exploded pie graphs)</td>
</tr>
<tr>
<td></td>
<td>type</td>
<td></td>
<td>string</td>
<td>0 - SIMPLE (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 - GRAPH_SUM (value of the item represents the whole pie; used only by pie and exploded pie graphs)</td>
</tr>
<tr>
<td>item</td>
<td></td>
<td>x</td>
<td>string</td>
<td>Individual item.</td>
</tr>
<tr>
<td>host</td>
<td></td>
<td>x</td>
<td>string</td>
<td>Item host.</td>
</tr>
<tr>
<td>key</td>
<td></td>
<td>x</td>
<td>string</td>
<td>Item key.</td>
</tr>
</tbody>
</table>

Template web scenario tags

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>Required</th>
<th>Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>httptests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Root element for web scenarios.</td>
</tr>
<tr>
<td></td>
<td>uuid</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Unique identifier for this web scenario.</td>
</tr>
<tr>
<td></td>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Web scenario name.</td>
</tr>
<tr>
<td></td>
<td>delay</td>
<td>-</td>
<td>string</td>
<td>Default: 1m</td>
<td>Frequency of executing the web scenario.</td>
</tr>
<tr>
<td></td>
<td>attempts</td>
<td>-</td>
<td>integer</td>
<td>1-10 (default: 1)</td>
<td>The number of attempts for executing web scenario steps.</td>
</tr>
<tr>
<td></td>
<td>agent</td>
<td>-</td>
<td>string</td>
<td>Default: Zabbix</td>
<td>Client agent. Zabbix will pretend to be the selected browser. This is useful when a website returns different content for different browsers.</td>
</tr>
<tr>
<td></td>
<td>http_proxy</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Specify an HTTP proxy to use, using the format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[username[:password]@[proxy.example.com[:port]]]</td>
<td></td>
</tr>
<tr>
<td>variables</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Root element for scenario-level variables (macros) that may be used in scenario steps.</td>
</tr>
<tr>
<td></td>
<td>name</td>
<td>x</td>
<td>text</td>
<td></td>
<td>Variable name.</td>
</tr>
<tr>
<td></td>
<td>value</td>
<td>x</td>
<td>text</td>
<td></td>
<td>Variable value.</td>
</tr>
<tr>
<td>headers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Root element for HTTP headers that will be sent when performing a request. Headers should be listed using the same syntax as they would appear in the HTTP protocol.</td>
</tr>
<tr>
<td></td>
<td>name</td>
<td>x</td>
<td>text</td>
<td></td>
<td>Header name.</td>
</tr>
<tr>
<td></td>
<td>value</td>
<td>x</td>
<td>text</td>
<td></td>
<td>Header value.</td>
</tr>
<tr>
<td></td>
<td>status</td>
<td>-</td>
<td>string</td>
<td>0 - ENABLED (default)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 - DISABLED</td>
<td>Web scenario status.</td>
</tr>
<tr>
<td></td>
<td>authentication</td>
<td></td>
<td>string</td>
<td>0 - NONE (default)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 - BASIC</td>
<td>Authentication method.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 - NTLM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>http_user</td>
<td>-</td>
<td>string</td>
<td></td>
<td>User name used for basic, HTTP or NTLM authentication.</td>
</tr>
<tr>
<td></td>
<td>http_password</td>
<td></td>
<td>string</td>
<td></td>
<td>Password used for basic, HTTP or NTLM authentication.</td>
</tr>
<tr>
<td></td>
<td>verify_peer</td>
<td>-</td>
<td>string</td>
<td>0 - NO (default)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 - YES</td>
<td>Verify the SSL certificate of the web server.</td>
</tr>
<tr>
<td>Element</td>
<td>Element property</td>
<td>Required</td>
<td>Type</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
<td>----------</td>
<td>------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>verify_host</td>
<td>-</td>
<td>string</td>
<td>0 - NO (default) 1 - YES</td>
<td>Verify that the Common Name field or the Subject Alternate Name field of the web server certificate matches.</td>
<td></td>
</tr>
<tr>
<td>ssl_cert_file</td>
<td>-</td>
<td>string</td>
<td>Name of the SSL certificate file used for client authentication (must be in PEM format).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssl_key_file</td>
<td>-</td>
<td>string</td>
<td>Name of the SSL private key file used for client authentication (must be in PEM format).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ssl_key_password</td>
<td>-</td>
<td>string</td>
<td>SSL private key file password.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>steps</td>
<td>x</td>
<td></td>
<td></td>
<td>Root element for web scenario steps.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Web scenario step name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>url</td>
<td>x</td>
<td>string</td>
<td>URL for monitoring.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>query_fields</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for query fields - an array of HTTP fields that will be added to the URL when performing a request.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Query field name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>string</td>
<td>Query field value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>posts</td>
<td>-</td>
<td></td>
<td></td>
<td>HTTP POST variables as a string (raw post data) or as an array of HTTP fields (form field data).</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Post field name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>x</td>
<td>string</td>
<td>Post field value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>variables</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element of step-level variables (macros) that should be applied after this step.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Variable name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>x</td>
<td>string</td>
<td>Variable value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>headers</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for HTTP headers that will be sent when performing a request. Headers should be listed using the same syntax as they would appear in the HTTP protocol.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Header name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>x</td>
<td>string</td>
<td>Header value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>follow_redirects</td>
<td>string</td>
<td>0 - NO 1 - YES (default)</td>
<td>Follow HTTP redirects.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>retrieve_mode-</td>
<td>string</td>
<td>0 - BODY (default) 1 - HEADERS 2 - BOTH</td>
<td>HTTP response retrieve mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>timeout</td>
<td>-</td>
<td>string</td>
<td>Default: 15s</td>
<td>Timeout of step execution. Seconds, time unit with suffix or user macro.</td>
<td></td>
</tr>
<tr>
<td>required</td>
<td>-</td>
<td>string</td>
<td>Text that must be present in the response. Ignored if empty.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>status_codes</td>
<td>-</td>
<td>string</td>
<td>A comma delimited list of accepted HTTP status codes. Ignored if empty. For example: 200-201,210-299</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for web scenario tags.</td>
<td></td>
</tr>
<tr>
<td>tag</td>
<td>x</td>
<td>string</td>
<td>Tag name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>string</td>
<td>Tag value.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Template dashboard tags

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>Required</th>
<th>Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dashboards</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for template dashboards.</td>
<td></td>
</tr>
<tr>
<td>Element</td>
<td>Property</td>
<td>Required</td>
<td>Type</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>uuid</td>
<td>x</td>
<td>string</td>
<td></td>
<td></td>
<td>Unique identifier for this dashboard.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td></td>
<td>Template dashboard name.</td>
</tr>
<tr>
<td>display</td>
<td></td>
<td>integer</td>
<td></td>
<td></td>
<td>Display period of dashboard pages.</td>
</tr>
<tr>
<td>period</td>
<td></td>
<td>integer</td>
<td></td>
<td></td>
<td>Slideshow auto start.</td>
</tr>
<tr>
<td>auto_start</td>
<td></td>
<td>string</td>
<td>0-no</td>
<td>1-yes</td>
<td>Root element for template dashboard pages.</td>
</tr>
<tr>
<td>pages</td>
<td></td>
<td>string</td>
<td></td>
<td></td>
<td>Page name.</td>
</tr>
<tr>
<td>display</td>
<td></td>
<td>integer</td>
<td></td>
<td></td>
<td>Page display period.</td>
</tr>
<tr>
<td>period</td>
<td></td>
<td>integer</td>
<td></td>
<td></td>
<td>Page sorting order.</td>
</tr>
<tr>
<td>widgets</td>
<td></td>
<td>string</td>
<td></td>
<td></td>
<td>Root element for template dashboard widgets.</td>
</tr>
<tr>
<td>type</td>
<td>x</td>
<td>string</td>
<td></td>
<td></td>
<td>Widget type.</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>string</td>
<td></td>
<td></td>
<td>Widget name.</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>integer</td>
<td>0-23</td>
<td></td>
<td>Horizontal position from the left side of the template dashboard.</td>
</tr>
<tr>
<td>y</td>
<td></td>
<td>integer</td>
<td>0-62</td>
<td></td>
<td>Vertical position from the top of the template dashboard.</td>
</tr>
<tr>
<td>width</td>
<td></td>
<td>integer</td>
<td>1-24</td>
<td></td>
<td>Widget width.</td>
</tr>
<tr>
<td>height</td>
<td></td>
<td>integer</td>
<td>2-32</td>
<td></td>
<td>Widget height.</td>
</tr>
<tr>
<td>hide_header</td>
<td></td>
<td>string</td>
<td>0-no</td>
<td>1-yes</td>
<td>Hide widget header.</td>
</tr>
<tr>
<td>fields</td>
<td></td>
<td>string</td>
<td></td>
<td></td>
<td>Root element for the template dashboard widget fields.</td>
</tr>
<tr>
<td>type</td>
<td>x</td>
<td>string</td>
<td>0-INTEGER</td>
<td>1-STRING</td>
<td>3-HOST</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td></td>
<td>Widget field name.</td>
</tr>
<tr>
<td>value</td>
<td>x</td>
<td>mixed</td>
<td></td>
<td></td>
<td>Widget field value, depending on the field type.</td>
</tr>
</tbody>
</table>

Footnotes

1 For string values, only the string will be exported (e.g. “ZABBIX_ACTIVE”) without the numbering used in this table. The numbers for range values (corresponding to the API values) in this table is used for ordering only.

4 Hosts

Overview

Hosts are exported with many related objects and object relations.

Host export contains:
- linked host groups
- host data
- template linkage
- host group linkage
- host interfaces
- directly linked items
- directly linked triggers
- directly linked graphs
- directly linked discovery rules with all prototypes
- directly linked web scenarios
- host macros
• host inventory data
• value maps

Exporting

To export hosts, do the following:

• Go to: Configuration → Hosts
• Mark the checkboxes of the hosts to export
• Click on Export below the list

Depending on the selected format, hosts are exported to a local file with a default name:

• zabbix_export_hosts.yaml - in YAML export (default option for export)
• zabbix_export_hosts.xml - in XML export
• zabbix_export_hosts.json - in JSON export

Importing

To import hosts, do the following:

• Go to: Configuration → Hosts
• Click on Import to the right
• Select the import file
• Mark the required options in import rules
• Click on import
A success or failure message of the import will be displayed in the frontend.

Import rules:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update existing</td>
<td>Existing elements will be updated with data taken from the import file. Otherwise they will not be updated.</td>
</tr>
<tr>
<td>Create new</td>
<td>The import will add new elements using data from the import file. Otherwise it will not add them.</td>
</tr>
<tr>
<td>Delete missing</td>
<td>The import will remove existing elements not present in the import file. Otherwise it will not remove them.</td>
</tr>
<tr>
<td></td>
<td>If Delete missing is marked for template linkage, existing template linkage not present in the import file will be removed from the host along with all entities inherited from the potentially unlinked templates (items, triggers, etc).</td>
</tr>
</tbody>
</table>

Export format

Export format in YAML:

```
zabbix_export:
  version: '6.2'
  date: '2021-09-28T12:20:07Z'
  host_groups:
    -
      uuid: f2481361f99448eea617b7b1d4765566
      name: 'Discovered hosts'
    -
      uuid: 6f6799aa66e844b4b3918f779f2abf08
      name: 'Zabbix servers'
  hosts:
    -
      host: 'Zabbix server 1'
      name: 'Main Zabbix server'
      templates:
        -
          name: 'Linux by Zabbix agent'
```
name: 'Zabbix server health'
groups:

- name: 'Discovered hosts'

- name: 'Zabbix servers'

interfaces:

- ip: 192.168.1.1
  interface_ref: if1

items:

- name: 'Zabbix trap'
type: TRAP
key: trap
delay: '0'
history: 1w
preprocessing:

  - type: MULTIPLIER
    parameters:
      - '8'

tags:

- tag: Application
  value: 'Zabbix server'

triggers:

- expression: 'last(/Zabbix server 1/trap)=0'
  name: 'Last value is zero'
priority: WARNING
tags:

  - tag: Process
    value: 'Internal test'

tags:

- tag: Process
  value: Zabbix

macros:

- macro: '{$HOST.MACRO}'
  value: '123'

- macro: '{$PASSWORD1}'
  type: SECRET_TEXT

inventory:

  type: 'Zabbix server'
  name: yyyyyy-HP-Pro-3010-Small-Form-Factor-PC
  os: 'Linux yyyyyy-HP-Pro-3010-Small-Form-Factor-PC 4.4.0-165-generic #193-Ubuntu SMP Tue Sep 17 17:42:52 UTC 2019 x86_64'
inventory_mode: AUTOMATIC

graphs:

- name: 'CPU utilization server'
  show_work_period: 'NO'
  show_triggers: 'NO'
  graph_items:

    - drawtype: FILLED_REGION
      color: FF5555
      item:
        host: 'Zabbix server 1'
Element tags

Element tag values are explained in the table below.

Host tags

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>Required Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>host_groups</td>
<td>x</td>
<td></td>
<td></td>
<td>Root element for host groups.</td>
</tr>
<tr>
<td>uuid</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Unique identifier for this host group.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Host group name.</td>
</tr>
<tr>
<td>Element</td>
<td>Element property</td>
<td>Required Type</td>
<td>Range¹</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------</td>
<td>---------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>hosts</td>
<td>host</td>
<td>x</td>
<td>string</td>
<td>Unique host name.</td>
</tr>
<tr>
<td></td>
<td>name</td>
<td>-</td>
<td>string</td>
<td>Visible host name.</td>
</tr>
<tr>
<td></td>
<td>description</td>
<td>-</td>
<td>text</td>
<td>Host description.</td>
</tr>
</tbody>
</table>
|           | status           | -             | string | 0 - ENABLED (default)  
<p>|           |                  |               |        | 1 - DISABLED       |
|           | ipmi_authtype    |               | string | IPMI session authentication type. |
|           | ipmi_privilege   |               | string | IPMI session privilege level. |
|           | ipmi_username    |               | string | Username for IPMI checks. |
|           | ipmi_password    |               | string | Password for IPMI checks. |
|           | proxy            |               | -      | Proxy. |
|           | name             | x             | string | Name of the proxy (if any) that monitors the host. |
| templates | name             | -             | string | Template name. |
| interfaces| default          | -             | string | Whether this is the primary host interface. |
|           | type             | -             | string | Interface type. |
|           | useip            | -             | string | Whether to use IP as the interface for connecting to the host (if not, DNS will be used). |
|           | ip               | -             | string | IP address, can be either IPv4 or IPv6. |
|           | dns              | -             | string | Required if the connection is made via IP. DNS name. |
|           | port             | -             | string | Required if the connection is made via DNS. Port number. Supports user macros. |
| details   | interface_ref    | x             | string | Format: if&lt;\n&gt; Interface reference name to be used in items. Root element for interface details. |
|           | version          | -             | string | Use this SNMP version. |
|           | community        | -             | string | SNMP community. |
|           | contextname      | -             | string | Required by SNMPv1 and SNMPv2 items. SNMPv3 context name. |
|           | securityname     | -             | string | Used only by SNMPv3 items. SNMPv3 security name. |
|           | securitylevel    |                | string | Used only by SNMPv3 items. SNMPv3 security level. |</p>
<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>authprotocol</td>
<td>string</td>
<td>0 - MDS (default)</td>
<td></td>
<td>SNMPv3 authentication protocol.</td>
</tr>
<tr>
<td>authpassphrase</td>
<td>string</td>
<td>Used only by SNMPv3 items.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>privprotocol</td>
<td>string</td>
<td>0 - DES (default)</td>
<td></td>
<td>SNMPv3 privacy protocol.</td>
</tr>
<tr>
<td>privpassphrase</td>
<td>string</td>
<td>Used only by SNMPv3 items.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bulk</td>
<td>string</td>
<td>0 - NO</td>
<td>Used only by SNMPv3 items.</td>
<td></td>
</tr>
<tr>
<td>items</td>
<td>-</td>
<td>Root element for items.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>-</td>
<td>Root element for host tags.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tag</td>
<td>x</td>
<td>string</td>
<td>Tag name.</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>string</td>
<td>Tag value.</td>
<td></td>
</tr>
<tr>
<td>macros</td>
<td>-</td>
<td>Root element for macros.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>macro</td>
<td>x</td>
<td>string</td>
<td>User macro name.</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>-</td>
<td>string</td>
<td>Type of the macro.</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>string</td>
<td>User macro value.</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>-</td>
<td>string</td>
<td>User macro description.</td>
<td></td>
</tr>
<tr>
<td>inventory</td>
<td>-</td>
<td>Root element for host inventory.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;inventory_property&gt;</td>
<td></td>
<td>Individual inventory property.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>inventory_mode</td>
<td>-</td>
<td>string</td>
<td>Inventory mode.</td>
<td></td>
</tr>
<tr>
<td>valuemaps</td>
<td>-</td>
<td>Root element for host value maps.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Value map name.</td>
<td></td>
</tr>
<tr>
<td>mapping</td>
<td>-</td>
<td>Root element for mappings.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>x</td>
<td>string</td>
<td>Value of a mapping.</td>
<td></td>
</tr>
<tr>
<td>newvalue</td>
<td>x</td>
<td>string</td>
<td>New value of a mapping.</td>
<td></td>
</tr>
</tbody>
</table>

Host item tags

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>items</td>
<td>-</td>
<td>Root element for items.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Item name.</td>
<td></td>
</tr>
<tr>
<td>Element</td>
<td>Element property</td>
<td>Required Type</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>---------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>snmp_oid</td>
<td>-</td>
<td>string</td>
<td></td>
<td>SNMP object ID. Required by SNMP items.</td>
</tr>
<tr>
<td>key</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Item key.</td>
</tr>
<tr>
<td>delay</td>
<td>-</td>
<td>string</td>
<td>Default: 1m</td>
<td>Update interval of the item. Note that delay will be always ‘0’ for trapper items. Accepts seconds or a time unit with suffix (30s, 1m, 2h, 1d). Optionally one or more custom intervals can be specified either as flexible intervals or scheduling. Multiple intervals are separated by a semicolon. User macros may be used. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. Flexible intervals may be written as two macros separated by a forward slash (e.g. {$FLEX_INTERVAL$}/{$FLEX_PERIOD$}).</td>
</tr>
<tr>
<td>history</td>
<td>-</td>
<td>string</td>
<td>Default: 90d</td>
<td>A time unit of how long the history data should be stored. Time unit with suffix, user macro or LLD macro.</td>
</tr>
<tr>
<td>trends</td>
<td>-</td>
<td>string</td>
<td>Default: 365d</td>
<td>A time unit of how long the trends data should be stored. Time unit with suffix, user macro or LLD macro.</td>
</tr>
<tr>
<td>status</td>
<td>-</td>
<td>string</td>
<td>0 - ENABLED (default) 1 - DISABLED</td>
<td>Item status.</td>
</tr>
<tr>
<td>value_type</td>
<td>-</td>
<td>string</td>
<td>0 - FLOAT 1 - CHAR 2 - LOG 3 - UNSIGNED (default) 4 - TEXT</td>
<td>Received value type.</td>
</tr>
<tr>
<td>allowed_hosts</td>
<td>-</td>
<td>string</td>
<td></td>
<td>List of IP addresses (comma delimited) of hosts allowed sending data for the item.</td>
</tr>
<tr>
<td>units</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Units of returned values (bps, B, etc).</td>
</tr>
<tr>
<td>Element</td>
<td>property</td>
<td>RequiredType</td>
<td>Range¹</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>--------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>params</td>
<td>-</td>
<td>text</td>
<td></td>
<td>Additional parameters depending on the type of the item:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- executed script for Script, SSH and Telnet items;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- SQL query for database monitor items;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- formula for calculated items.</td>
</tr>
<tr>
<td>ipmi_sensor</td>
<td>-</td>
<td>string</td>
<td></td>
<td>IPMI sensor.</td>
</tr>
<tr>
<td>authtype</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Authentication type for SSH agent items:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 - PASSWORD (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 - PUBLIC_KEY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Authentication type for HTTP agent items:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 - NONE (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 - BASIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 - NTLM</td>
</tr>
<tr>
<td>username</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Username for authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Required by SSH and Telnet items.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When used by JMX agent, password should also be specified together with the username or both properties should be left blank.</td>
</tr>
<tr>
<td>password</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Password for authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When used by JMX agent, username should also be specified together with the password or both properties should be left blank.</td>
</tr>
<tr>
<td>publickey</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Name of the public key file.</td>
</tr>
<tr>
<td>privatekey</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Required for SSH agent items.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Name of the private key file.</td>
</tr>
<tr>
<td>description</td>
<td>-</td>
<td>text</td>
<td>0 - NONE</td>
<td>Required for SSH agent items.</td>
</tr>
<tr>
<td>inventory_link</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Item description.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Host inventory field that is populated by the item.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Capitalized host inventory field name. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4 - ALIAS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6 - OS_FULL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14 - HARDWARE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>etc.</td>
</tr>
<tr>
<td>logtimefmt</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Format of the time in log entries.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Used only by log items.</td>
</tr>
<tr>
<td>interface_ref</td>
<td>-</td>
<td>string</td>
<td>Format: if&lt;N&gt;</td>
<td>Reference to the host interface.</td>
</tr>
<tr>
<td>jmx_endpoint</td>
<td>-</td>
<td>string</td>
<td></td>
<td>JMX endpoint.</td>
</tr>
<tr>
<td>url</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Required only for HTTP agent items.</td>
</tr>
<tr>
<td>Element</td>
<td>Property</td>
<td>Required Type</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>---------------</td>
<td>--------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>allow_traps</td>
<td></td>
<td>string</td>
<td>0-NO (default) 1-YES</td>
<td>Allow to populate value as in a trapper item. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>follow_redirects</td>
<td></td>
<td>string</td>
<td>0-NO 1-YES (default)</td>
<td>Follow HTTP response redirects while pooling data. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>headers</td>
<td></td>
<td></td>
<td></td>
<td>Used only by HTTP agent items. Root element for HTTP(S) request headers, where header name is used as key and header value as value. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Header name. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>value</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Header value. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>http_proxy</td>
<td></td>
<td>string</td>
<td></td>
<td>HTTP(S) proxy connection string. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>output_format</td>
<td></td>
<td>string</td>
<td>0-RAW (default) 1-JSON</td>
<td>How to process response. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>post_type</td>
<td></td>
<td>string</td>
<td>0-RAW (default) 2-JSON 3-XML</td>
<td>Type of post data body. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>posts</td>
<td></td>
<td>string</td>
<td></td>
<td>HTTP(S) request body data. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>query_fields</td>
<td></td>
<td></td>
<td></td>
<td>Used only by HTTP agent items. Root element for query parameters.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Parameter name. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>value</td>
<td></td>
<td>string</td>
<td></td>
<td>Parameter value. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>request_method</td>
<td></td>
<td>string</td>
<td>0-GET (default) 1-POST 2-PUT 3-HEAD</td>
<td>Request method. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>retrieve_mode</td>
<td></td>
<td>string</td>
<td>0-BODY (default) 1-HEADERS 2-BOTH</td>
<td>What part of response should be stored. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>ssl_cert_file</td>
<td></td>
<td>string</td>
<td></td>
<td>Public SSL Key file path. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>ssl_key_file</td>
<td></td>
<td>string</td>
<td></td>
<td>Private SSL Key file path. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>ssl_key_password</td>
<td></td>
<td>string</td>
<td></td>
<td>Password for SSL Key file. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>status_codes</td>
<td></td>
<td>string</td>
<td></td>
<td>Ranges of required HTTP status codes separated by commas. Supports user macros. Example: 200,200-{$M},{$M},200-400 Used only by HTTP agent items.</td>
</tr>
<tr>
<td>timeout</td>
<td></td>
<td>string</td>
<td></td>
<td>Item data polling request timeout. Supports user macros.</td>
</tr>
<tr>
<td>verify_host</td>
<td></td>
<td>string</td>
<td>0-NO (default) 1-YES</td>
<td>Validate if host name in URL is in Common Name field or a Subject Alternate Name field of host certificate. Used only by HTTP agent items.</td>
</tr>
<tr>
<td>Element</td>
<td>Property</td>
<td>Required</td>
<td>Type</td>
<td>Range</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>verify_peer</td>
<td>-</td>
<td>string</td>
<td>0 - NO (default) 1 - YES</td>
<td>Validate if host certificate is authentic.</td>
</tr>
<tr>
<td>parameters</td>
<td>-</td>
<td></td>
<td></td>
<td>Used only by HTTP agent items. Root element for user-defined parameters.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Used only by Script items. Parameter name.</td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Used only by Script items. Parameter value.</td>
</tr>
<tr>
<td>value map</td>
<td>-</td>
<td></td>
<td></td>
<td>Used only by Script items. Value map.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Name of the value map to use for the item.</td>
</tr>
<tr>
<td>preprocessing</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for item value preprocessing step.</td>
</tr>
<tr>
<td>step</td>
<td>-</td>
<td></td>
<td></td>
<td>Individual item value preprocessing step.</td>
</tr>
<tr>
<td>type</td>
<td>x</td>
<td>string</td>
<td>1 - MULTIPLIER 2 - RTRIM 3 - LTRIM 4 - TRIM 5 - REGEX 6 - BOOL_TO_DECIMAL 7 - OCTAL_TO_DECIMAL 8 - HEX_TO_DECIMAL 9 - SIMPLE_CHANGE (calculated as (received value-previous value)) 10 - CHANGE_PER_SECOND (calculated as (received value-previous value)/(time now-time of last check)) 11 - XMLPATH 12 - JSONPATH 13 - IN_RANGE 14 - MATCHES_REGEX 15 - NOT_MATCHES_REGEX 16 - CHECK_JSON_ERROR 17 - CHECK_XML_ERROR 18 - CHECK_REGEX_ERROR 19 - DISCARD_UNCHANGED 20 - DISCARD_UNCHANGED_HEARTBEAT 21 - JAVASCRIPT 22 - PROMETHEUS_PATTERN 23 - PROMETHEUS_TO_JSON 24 - CSV_TO_JSON 25 - STR_REPLACE 26 - CHECK_NOT_SUPPORTED 27 - XML_TO_JSON</td>
<td>Type of the item value preprocessing step.</td>
</tr>
<tr>
<td>Element</td>
<td>Element property</td>
<td>RequiredType</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
<td>--------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>parameters</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for parameters of the item value preprocessing step.</td>
</tr>
<tr>
<td>parameter</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Individual parameter of the item value preprocessing step.</td>
</tr>
<tr>
<td>error_handler</td>
<td>-</td>
<td>string</td>
<td>0 - ORIGINAL_ERROR (default) 1 - DISCARD_VALUE 2 - CUSTOM_VALUE 3 - CUSTOM_ERROR</td>
<td>Action type used in case of preprocessing step failure.</td>
</tr>
<tr>
<td>error_handler_params</td>
<td>string</td>
<td></td>
<td></td>
<td>Error handler parameters used with 'error_handler'.</td>
</tr>
<tr>
<td>master_item</td>
<td>-</td>
<td></td>
<td></td>
<td>Individual item master item.</td>
</tr>
<tr>
<td>key</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Required by dependent items.</td>
</tr>
<tr>
<td>triggers</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for simple triggers.</td>
</tr>
<tr>
<td>triggers</td>
<td>For trigger element tag values, see host trigger tags.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for item tags.</td>
</tr>
<tr>
<td>tag</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Tag name.</td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Tag value.</td>
</tr>
</tbody>
</table>

**Host low-level discovery rule tags**

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
</table>

For most of the element tag values, see element tag values for a regular item. Only the tags that are specific to low-level discovery rules, are described below.
<table>
<thead>
<tr>
<th>Element property</th>
<th>Required Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>string</td>
<td>0-ZABBIX_PASSIVE (default) 2-TRAP 3-SIMPLE 5-INTERNAL 7-ZABBIX_ACTIVE 10-EXTERNAL 11-ODBC 12-IPMI 13-SSH 14-TELNET 16-JMX 18-DEPENDENT 19-HTTP_AGENT 20-SNMP_AGENT</td>
<td>Item type.</td>
</tr>
<tr>
<td>lifetime</td>
<td>string</td>
<td>Default: 30d</td>
<td>Time period after which items that are no longer discovered will be deleted. Seconds, time unit with suffix or user macro.</td>
</tr>
<tr>
<td>filter evaltype</td>
<td>string</td>
<td>0-AND_OR (default) 1-AND 2-OR 3-FORMULA</td>
<td>Logic to use for checking low-level discovery rule filter conditions.</td>
</tr>
<tr>
<td>conditions</td>
<td>-</td>
<td></td>
<td>Root element for filter conditions.</td>
</tr>
<tr>
<td>macro value</td>
<td>string</td>
<td></td>
<td>Low-level discovery macro name.</td>
</tr>
<tr>
<td>operator</td>
<td>string</td>
<td>8-MATCHES_REGEX (default) 9-NOT_MATCHES_REGEX</td>
<td>Condition operator.</td>
</tr>
<tr>
<td>formulaid</td>
<td>character</td>
<td></td>
<td>Arbitrary unique ID that is used to reference a condition from the custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.</td>
</tr>
<tr>
<td>lld_macro_paths</td>
<td>-</td>
<td></td>
<td>Root element for LLD macro paths.</td>
</tr>
<tr>
<td>lld_macro path</td>
<td>string</td>
<td></td>
<td>Low-level discovery macro name.</td>
</tr>
<tr>
<td>preprocessing</td>
<td>-</td>
<td></td>
<td>LLD rule value preprocessing.</td>
</tr>
<tr>
<td>step</td>
<td>-</td>
<td></td>
<td>Individual LLD rule value preprocessing step.</td>
</tr>
<tr>
<td>Element property</td>
<td>RequiredType</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>trigger_prototypes</td>
<td>x</td>
<td>string</td>
<td>5 - REGEX, 11 - XMLPATH, 12 - JSONPATH, 15 - NOT_MATCHES_REGEX, 16 - CHECK_JSON_ERROR, 17 - CHECK_XML_ERROR, 20 - DISCARD_UNCHANGED_HEARTBEAT, 21 - JAVASCRIPT, 23 - PROMETHEUS_TO_JSON, 24 - CSV_TO_JSON, 25 - STR_REPLACE, 27 - XML_TO_JSON</td>
</tr>
<tr>
<td>graph_prototypes</td>
<td>-</td>
<td></td>
<td>Root element for graph prototypes.</td>
</tr>
<tr>
<td>host_prototypes</td>
<td>-</td>
<td></td>
<td>Root element for host prototypes.</td>
</tr>
</tbody>
</table>

For most of the element tag values, see element tag values for a host item value preprocessing. Only the tags that are specific to low-level discovery value preprocessing, are described below.

Type of the item value preprocessing step.
<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>item_prototypes</td>
<td>-</td>
<td></td>
<td>-</td>
<td>Root element for item prototypes.</td>
</tr>
<tr>
<td></td>
<td>For item prototype element tag values, see regular host item tags.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>master_item</td>
<td>-</td>
<td></td>
<td>-</td>
<td>Individual item prototype master item/item prototype data.</td>
</tr>
<tr>
<td></td>
<td>key</td>
<td>x</td>
<td>string</td>
<td>Dependent item prototype master item/item prototype key value. Required for a dependent item.</td>
</tr>
</tbody>
</table>

Host trigger tags

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>triggers</td>
<td>-</td>
<td></td>
<td>-</td>
<td>Root element for triggers.</td>
</tr>
<tr>
<td></td>
<td>expression</td>
<td>x</td>
<td>string</td>
<td>Trigger expression.</td>
</tr>
<tr>
<td></td>
<td>recovery_mode</td>
<td>string</td>
<td>0 - EXPRESSION (default) 1 - RECOVERY_EXPRESSION 2 - NONE</td>
<td>Basis for generating OK events.</td>
</tr>
<tr>
<td></td>
<td>recovery_expression</td>
<td>string</td>
<td></td>
<td>Trigger recovery expression.</td>
</tr>
<tr>
<td></td>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Trigger name.</td>
</tr>
<tr>
<td></td>
<td>correlation_mode</td>
<td>string</td>
<td>0 - DISABLED (default) 1 - TAG_VALUE</td>
<td>Correlation mode (no event correlation or event correlation by tag).</td>
</tr>
<tr>
<td></td>
<td>correlation_tag</td>
<td>string</td>
<td></td>
<td>The tag name to be used for event correlation.</td>
</tr>
<tr>
<td></td>
<td>url</td>
<td>-</td>
<td>string</td>
<td>URL associated with the trigger.</td>
</tr>
<tr>
<td></td>
<td>status</td>
<td>-</td>
<td>string</td>
<td>0 - ENABLED (default) 1 - DISABLED</td>
</tr>
<tr>
<td></td>
<td>priority</td>
<td>-</td>
<td>string</td>
<td>0 - NOT_CLASSIFIED (default) 1 - INFO 2 - WARNING 3 - AVERAGE 4 - HIGH 5 - DISASTER</td>
</tr>
<tr>
<td></td>
<td>description</td>
<td>-</td>
<td>text</td>
<td>Trigger description.</td>
</tr>
<tr>
<td></td>
<td>type</td>
<td>-</td>
<td>string</td>
<td>0 - SINGLE (default) 1 - MULTIPLE</td>
</tr>
<tr>
<td></td>
<td>manual_close</td>
<td>-</td>
<td>string</td>
<td>0 - NO (default) 1 - YES</td>
</tr>
<tr>
<td></td>
<td>dependencies</td>
<td></td>
<td>-</td>
<td>Root element for dependencies.</td>
</tr>
<tr>
<td></td>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Dependency trigger name.</td>
</tr>
<tr>
<td></td>
<td>expression</td>
<td>x</td>
<td>string</td>
<td>Dependency trigger expression.</td>
</tr>
<tr>
<td></td>
<td>recovery_expression</td>
<td>string</td>
<td></td>
<td>Dependency trigger recovery expression.</td>
</tr>
<tr>
<td></td>
<td>tags</td>
<td>-</td>
<td>-</td>
<td>Root element for event tags.</td>
</tr>
<tr>
<td></td>
<td>tag</td>
<td>x</td>
<td>string</td>
<td>Tag name.</td>
</tr>
<tr>
<td></td>
<td>value</td>
<td>-</td>
<td>string</td>
<td>Tag value.</td>
</tr>
</tbody>
</table>

Host graph tags
<table>
<thead>
<tr>
<th>Element</th>
<th>property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphs</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Root element for graphs.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>-</td>
<td>Graph name.</td>
</tr>
<tr>
<td>width</td>
<td>-</td>
<td>integer</td>
<td>20-65535 (default: 900)</td>
<td>Graph width, in pixels. Used for preview and for pie/explored graphs.</td>
</tr>
<tr>
<td>height</td>
<td>-</td>
<td>integer</td>
<td>20-65535 (default: 200)</td>
<td>Graph height, in pixels. Used for preview and for pie/explored graphs.</td>
</tr>
<tr>
<td>yaxismin</td>
<td>-</td>
<td>double</td>
<td>Default: 0</td>
<td>Value of Y axis minimum.</td>
</tr>
<tr>
<td>yaxismax</td>
<td>-</td>
<td>double</td>
<td>Default: 0</td>
<td>Used if <code>ymin_type_1</code> is FIXED. Value of Y axis maximum.</td>
</tr>
<tr>
<td>show_work_period</td>
<td>string</td>
<td>0 - NO</td>
<td>1 - YES (default)</td>
<td>Highlight non-working hours.</td>
</tr>
<tr>
<td>show_triggers</td>
<td>string</td>
<td>0 - NO</td>
<td>1 - YES (default)</td>
<td>Used by normal and stacked graphs. Display simple trigger values as a line.</td>
</tr>
<tr>
<td>type</td>
<td>-</td>
<td>string</td>
<td>0 - NORMAL (default)</td>
<td>1 - STACKED 2 - PIE 3 - EXPLODED</td>
</tr>
<tr>
<td>show_legend</td>
<td>string</td>
<td>0 - NO</td>
<td>1 - YES (default)</td>
<td>Display graph legend.</td>
</tr>
<tr>
<td>show_3d</td>
<td>string</td>
<td>0 - NO (default)</td>
<td>1 - YES</td>
<td>Enable 3D style.</td>
</tr>
<tr>
<td>percent_left</td>
<td>-</td>
<td>double</td>
<td>Default: 0</td>
<td>Used by pie and exploded pie graphs. Show the percentile line for left axis.</td>
</tr>
<tr>
<td>percent_right</td>
<td>-</td>
<td>double</td>
<td>Default: 0</td>
<td>Used only for normal graphs. Show the percentile line for right axis.</td>
</tr>
<tr>
<td>ymin_type_1</td>
<td>string</td>
<td>0 - CALCULATED (default)</td>
<td>1 - FIXED 2 - ITEM</td>
<td>Minimum value of Y axis.</td>
</tr>
<tr>
<td>ymax_type_1</td>
<td>string</td>
<td>0 - CALCULATED (default)</td>
<td>1 - FIXED 2 - ITEM</td>
<td>Maximum value of Y axis.</td>
</tr>
<tr>
<td>ymin_item_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Individual item details.</td>
</tr>
<tr>
<td>host</td>
<td>x</td>
<td>string</td>
<td>-</td>
<td>Required if <code>ymin_type_1</code> is ITEM. Item host.</td>
</tr>
<tr>
<td>key</td>
<td>x</td>
<td>string</td>
<td>-</td>
<td>Item key.</td>
</tr>
<tr>
<td>ymax_item_1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Individual item details.</td>
</tr>
<tr>
<td>host</td>
<td>x</td>
<td>string</td>
<td>-</td>
<td>Required if <code>ymax_type_1</code> is ITEM. Item host.</td>
</tr>
<tr>
<td>key</td>
<td>x</td>
<td>string</td>
<td>-</td>
<td>Item key.</td>
</tr>
<tr>
<td>graph_items</td>
<td>x</td>
<td>-</td>
<td>-</td>
<td>Root element for graph items.</td>
</tr>
<tr>
<td>sortorder</td>
<td>-</td>
<td>integer</td>
<td>-</td>
<td>Draw order. The smaller value is drawn first. Can be used to draw lines or regions behind (or in front of) another.</td>
</tr>
<tr>
<td>Element property</td>
<td>Required</td>
<td>Type</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>--------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>drawtype</td>
<td>-</td>
<td>string</td>
<td>0 - SINGLE_LINE (default) 1 - FILLED_REGION 2 - BOLD_LINE 3 - DOTTED_LINE 4 - DASHED_LINE 5 - GRADIENT_LINE</td>
<td>Draw style of the graph item. Used only by normal graphs.</td>
</tr>
<tr>
<td>color</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Element color (6 symbols, hex).</td>
</tr>
<tr>
<td>yaxisside</td>
<td>-</td>
<td>string</td>
<td>0 - LEFT (default) 1 - RIGHT</td>
<td>Side of the graph where the graph item's Y scale will be drawn. Used by normal and stacked graphs.</td>
</tr>
<tr>
<td>calc_fnc</td>
<td>-</td>
<td>string</td>
<td>1 - MIN 2 - AVG (default) 4 - MAX 7 - ALL (minimum, average and maximum; used only by simple graphs) 9 - LAST (used only by pie and exploded pie graphs)</td>
<td>Data to draw if more than one value exists for an item.</td>
</tr>
<tr>
<td>type</td>
<td>-</td>
<td>string</td>
<td>0 - SIMPLE (default) 2 - GRAPH_SUM (value of the item represents the whole pie; used only by pie and exploded pie graphs)</td>
<td>Graph item type.</td>
</tr>
</tbody>
</table>

| item | x | Individual item. |
| host | x | string | Item host. |
| key  | x | string | Item key. |

Host web scenario tags

<table>
<thead>
<tr>
<th>Element property</th>
<th>Required</th>
<th>Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>httptests</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for web scenarios.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Web scenario name.</td>
</tr>
<tr>
<td>delay</td>
<td>-</td>
<td>string</td>
<td>Default: 1m</td>
<td>Frequency of executing the web scenario. Seconds, time unit with suffix or user macro.</td>
</tr>
<tr>
<td>attempts</td>
<td>-</td>
<td>integer</td>
<td>1-10 (default: 1)</td>
<td>The number of attempts for executing web scenario steps.</td>
</tr>
<tr>
<td>agent</td>
<td>-</td>
<td>string</td>
<td>Default: Zabbix</td>
<td>Client agent. Zabbix will pretend to be the selected browser. This is useful when a website returns different content for different browsers.</td>
</tr>
<tr>
<td>http_proxy</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Specify an HTTP proxy to use, using the format: http://[username[:password]@]proxy.example.com:port</td>
</tr>
</tbody>
</table>

variables

<table>
<thead>
<tr>
<th>Element property</th>
<th>Required</th>
<th>Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>x</td>
<td>text</td>
<td></td>
<td>Variable name.</td>
</tr>
<tr>
<td>value</td>
<td>x</td>
<td>text</td>
<td></td>
<td>Variable value.</td>
</tr>
</tbody>
</table>

headers

<table>
<thead>
<tr>
<th>Element property</th>
<th>Required</th>
<th>Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>x</td>
<td>text</td>
<td></td>
<td>Header name.</td>
</tr>
<tr>
<td>value</td>
<td>x</td>
<td>text</td>
<td></td>
<td>Header value.</td>
</tr>
<tr>
<td>Element</td>
<td>Property</td>
<td>Required Type</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------</td>
<td>---------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>status</td>
<td>-</td>
<td>string</td>
<td>0 - ENABLED (default) 1 - DISABLED</td>
<td>Web scenario status.</td>
</tr>
<tr>
<td>authentication</td>
<td>-</td>
<td>string</td>
<td>0 - NONE (default) 1 - BASIC 2 - NTLM</td>
<td>Authentication method.</td>
</tr>
<tr>
<td>http_user</td>
<td>-</td>
<td>string</td>
<td>User name used for basic, HTTP or NTLM authentication.</td>
<td></td>
</tr>
<tr>
<td>http_password</td>
<td>-</td>
<td>string</td>
<td>Verify the SSL certificate of the web server.</td>
<td></td>
</tr>
<tr>
<td>verify_peer</td>
<td>-</td>
<td>string</td>
<td>0 - NO (default) 1 - YES</td>
<td>Verify that the Common Name field or the Subject Alternate Name field of the web server certificate matches.</td>
</tr>
<tr>
<td>verify_host</td>
<td>-</td>
<td>string</td>
<td>0 - NO (default) 1 - YES</td>
<td>Name of the SSL certificate file used for client authentication (must be in PEM format).</td>
</tr>
<tr>
<td>ssl_cert_file</td>
<td>-</td>
<td>string</td>
<td>SSL private key file password.</td>
<td></td>
</tr>
<tr>
<td>ssl_key_file</td>
<td>-</td>
<td>string</td>
<td>Name of the SSL private key file used for client authentication (must be in PEM format).</td>
<td></td>
</tr>
<tr>
<td>ssl_key_password</td>
<td>-</td>
<td>string</td>
<td>SSL private key file password.</td>
<td></td>
</tr>
<tr>
<td>steps</td>
<td>-</td>
<td>x string</td>
<td>Root element for web scenario steps.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Web scenario step name.</td>
<td></td>
</tr>
<tr>
<td>url</td>
<td>x</td>
<td>string</td>
<td>URL for monitoring.</td>
<td></td>
</tr>
<tr>
<td>query_fields</td>
<td>-</td>
<td>x string</td>
<td>Root element for query fields - an array of HTTP fields that will be added to the URL when performing a request.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Query field name.</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>string</td>
<td>Query field value.</td>
<td></td>
</tr>
<tr>
<td>posts</td>
<td>-</td>
<td>x string</td>
<td>HTTP POST variables as a string (raw post data) or as an array of HTTP fields (form field data).</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Post field name.</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>x</td>
<td>string</td>
<td>Post field value.</td>
<td></td>
</tr>
<tr>
<td>variables</td>
<td>-</td>
<td></td>
<td>Root element of step-level variables (macros) that should be applied after this step.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Variable name.</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>x</td>
<td>string</td>
<td>Variable value.</td>
<td></td>
</tr>
<tr>
<td>headers</td>
<td>-</td>
<td></td>
<td>Root element for HTTP headers that will be sent when performing a request. Headers should be listed using the same syntax as they would appear in the HTTP protocol.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Header name.</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>x</td>
<td>string</td>
<td>Header value.</td>
<td></td>
</tr>
<tr>
<td>follow_redirects</td>
<td></td>
<td>string</td>
<td>0 - NO 1 - YES (default)</td>
<td>Follow HTTP redirects.</td>
</tr>
<tr>
<td>retrieve_mode</td>
<td></td>
<td>string</td>
<td>0 - BODY (default) 1 - HEADERS 2 - BOTH</td>
<td>HTTP response retrieve mode.</td>
</tr>
<tr>
<td>timeout</td>
<td>-</td>
<td>string</td>
<td>Default: 15s</td>
<td>Timeout of step execution. Seconds, time unit with suffix or user macro.</td>
</tr>
<tr>
<td>required</td>
<td>-</td>
<td>string</td>
<td>Text that must be present in the response. Ignored if empty.</td>
<td></td>
</tr>
<tr>
<td>status_codes</td>
<td>-</td>
<td>string</td>
<td>A comma delimited list of accepted HTTP status codes. Ignored if empty. For example: 200-201,210-299</td>
<td></td>
</tr>
<tr>
<td>Element property</td>
<td>RequiredType</td>
<td>Range</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>-------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>-</td>
<td>string</td>
<td>Root element for web scenario tags.</td>
<td></td>
</tr>
<tr>
<td>tag</td>
<td>x</td>
<td>string</td>
<td>Tag name.</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>string</td>
<td>Tag value.</td>
<td></td>
</tr>
</tbody>
</table>

Footnotes

1 For string values, only the string will be exported (e.g. “ZABBIX_ACTIVE”) without the numbering used in this table. The numbers for range values (corresponding to the API values) in this table is used for ordering only.

5 Network maps

Overview

Network map export contains:

- all related images
- map structure - all map settings, all contained elements with their settings, map links and map link status indicators

Any host groups, hosts, triggers, other maps or other elements that may be related to the exported map are not exported. Thus, if at least one of the elements the map refers to is missing, importing it will fail.

Network map export/import is supported since Zabbix 1.8.2.

Exporting

To export network maps, do the following:

- Go to: Monitoring → Maps
- Mark the checkboxes of the network maps to export
- Click on Export below the list

Depending on the selected format, maps are exported to a local file with a default name:

- zabbix_export_maps.yaml - in YAML export (default option for export)
- zabbix_export_maps.xml - in XML export
- zabbix_export_maps.json - in JSON export
Importing

To import network maps, do the following:

- Go to: Monitoring → Maps
- Click on Import to the right
- Select the import file
- Mark the required options in import rules
- Click on Import

All mandatory input fields are marked with a red asterisk.

A success or failure message of the import will be displayed in the frontend.

Import rules:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update existing</td>
<td>Existing maps will be updated with data taken from the import file. Otherwise they will not be updated.</td>
</tr>
<tr>
<td>Create new</td>
<td>The import will add new maps using data from the import file. Otherwise it will not add them.</td>
</tr>
</tbody>
</table>

If you uncheck both map options and check the respective options for images, images only will be imported. Image importing is only available to Super Admin users.

If replacing an existing image, it will affect all maps that are using this image.

Export format

Export to YAML:

```yaml
zabbix_export:

  version: '6.2'
  date: '2021-08-31T12:55:10Z'

images:
  -
    name: Zabbix_server_3D_(128)
    imagetype: '1'
    encodedImage: iVBOR...5CYII=

maps:
  -
    name: 'Local network'
    width: '680'
    height: '200'
    label_type: '0'
    label_location: '0'
    highlight: '1'
    expandproblem: '1'
    markelements: '1'
    show_unack: '0'
```
severity_min: '0'
show_suppressed: '0'
grid_size: '50'
grid_show: '1'
grid_align: '1'
label_format: '0'
label_type_host: '2'
label_type_hostgroup: '2'
label_type_trigger: '2'
label_type_map: '2'
label_type_image: '2'
label_string_host: ''
label_string_hostgroup: ''
label_string_trigger: ''
label_string_map: ''
label_string_image: ''
expand_macros: '1'
background: {}
iconmap: {}
urls: {}
selements:
  -
    elementtype: '0'
elements:
  -
    host: 'Zabbix server'
label: |
    {HOST.NAME}
    {HOST.NAME}
label_location: '0'
x: '111'
y: '61'
elementssubtype: '0'
areatype: '0'
width: '200'
height: '200'
viewtype: '0'
use_iconmap: '0'
selementid: '1'
icon_off:
    name: Zabbix_server_3D_(128)
icon_on: {}
icon_disabled: {}
icon_maintenance: {}
urls: {}
evaltype: '0'
shapes:
  -
    type: '0'
x: '0'
y: '0'
width: '680'
height: '15'
text: '{MAP_NAME}'
font: '9'
font_size: '11'
font_color: '000000'
text_halign: '0'
text_valign: '0'
border_type: '0'
border_width: '0'
border_color: '000000'
Element tags

Element tag values are explained in the table below.

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>images</td>
<td>name</td>
<td>string</td>
<td>Unique image name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>imagetype</td>
<td>integer</td>
<td>1 - image, 2 - background</td>
<td>Image type.</td>
</tr>
<tr>
<td></td>
<td>encodedImage</td>
<td></td>
<td>Base64 encoded image.</td>
<td></td>
</tr>
<tr>
<td>maps</td>
<td>name</td>
<td>string</td>
<td>Unique map name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>width</td>
<td>integer</td>
<td>Map width, in pixels.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>height</td>
<td>integer</td>
<td>Map height, in pixels.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>label_type</td>
<td>integer</td>
<td>0 - label, 1 - host IP address, 2 - element name, 3 - status only, 4 - nothing</td>
<td>Map element label type.</td>
</tr>
<tr>
<td></td>
<td>label_location</td>
<td>integer</td>
<td>0 - bottom, 1 - left, 2 - right, 3 - top</td>
<td>Map element label location by default.</td>
</tr>
<tr>
<td></td>
<td>highlight</td>
<td>integer</td>
<td>0 - no, 1 - yes</td>
<td>Enable icon highlighting for active triggers and host statuses.</td>
</tr>
<tr>
<td></td>
<td>expandproblem</td>
<td>integer</td>
<td>0 - no, 1 - yes</td>
<td>Display problem trigger for elements with a single problem.</td>
</tr>
<tr>
<td></td>
<td>markelements</td>
<td>integer</td>
<td>0 - no, 1 - yes</td>
<td>Highlight map elements that have recently changed their status.</td>
</tr>
<tr>
<td></td>
<td>show_unack</td>
<td>integer</td>
<td>0 - count of all problems, 1 - count of unacknowledged problems, 2 - count of acknowledged and unacknowledged problems separately</td>
<td>Problem display.</td>
</tr>
<tr>
<td></td>
<td>severity_min</td>
<td>integer</td>
<td>0 - not classified, 1 - information, 2 - warning, 3 - average, 4 - high, 5 - disaster</td>
<td>Minimum trigger severity to show on the map by default.</td>
</tr>
<tr>
<td></td>
<td>show_suppressed</td>
<td>integer</td>
<td>0 - no, 1 - yes</td>
<td>Display problems which would otherwise be suppressed (not shown) because of host maintenance.</td>
</tr>
<tr>
<td></td>
<td>grid_size</td>
<td>integer</td>
<td>20, 40, 50, 75 or 100</td>
<td>Cell size of a map grid in pixels, if “grid_show=1”</td>
</tr>
<tr>
<td></td>
<td>grid_show</td>
<td>integer</td>
<td>0 - yes, 1 - no</td>
<td>Display a grid in map configuration.</td>
</tr>
<tr>
<td></td>
<td>grid_align</td>
<td>integer</td>
<td>0 - yes, 1 - no</td>
<td>Automatically align icons in map configuration.</td>
</tr>
<tr>
<td></td>
<td>label_format</td>
<td>integer</td>
<td>0 - no, 1 - yes</td>
<td>Use advanced label configuration.</td>
</tr>
<tr>
<td>Element</td>
<td>Element property</td>
<td>Type</td>
<td>Range</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
<td>------</td>
<td>-------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| label_type_host | integer | 0 - label  
1 - host IP address  
2 - element name  
3 - status only  
4 - nothing  
5 - custom label | Display as host label, if "label_format=1" |
| label_type_host | integer | 0 - label  
2 - element name  
3 - status only  
4 - nothing  
5 - custom label | Display as host group label, if "label_format=1" |
| label_type_trigger | integer | 0 - label  
2 - element name  
3 - status only  
4 - nothing  
5 - custom label | Display as trigger label, if "label_format=1" |
| label_type_map | integer | 0 - label  
2 - element name  
3 - status only  
4 - nothing  
5 - custom label | Display as map label, if "label_format=1" |
| label_type_image | integer | 0 - label  
2 - element name  
4 - nothing  
5 - custom label | Display as image label, if "label_format=1" |
| label_string_host | string | 2 - element name  
5 - custom label | Custom label for host elements, if "label_type_host=5" |
| label_string_hostgroup | string | 2 - element name  
5 - custom label | Custom label for host group elements, if "label_type_hostgroup=5" |
| label_string_trigger | string | 2 - element name  
5 - custom label | Custom label for trigger elements, if "label_type_trigger=5" |
| label_string_map | string | 2 - element name  
5 - custom label | Custom label for map elements, if "label_type_map=5" |
| label_string_image | string | 2 - element name  
5 - custom label | Custom label for image elements, if "label_type_image=5" |
| expand_macros | integer | 0 - no  
1 - yes | Expand macros in labels in map configuration. |
| background | id | | ID of the background image (if any), if "imagetype=2" |
| iconmap | id | | ID of the icon mapping (if any). |
| urls | | | Used by maps or each map element. |
| name | string | | Link name. |
| url | string | | Link URL. |
| elementtype | integer | 0 - host  
1 - map  
2 - trigger  
3 - host group  
4 - image | Map item type the link belongs to. |
| elements | | | Map element type. |
| label | string | | Icon label. |
| label_location | integer | -1 - use map default  
0 - bottom  
1 - left  
2 - right  
3 - top | Location on the X axis. |
<p>| x | integer | | |</p>
<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>element subtype</td>
<td>integer</td>
<td>0 - single host group</td>
<td>Location on the Y axis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - all host groups</td>
<td></td>
</tr>
<tr>
<td>areatype</td>
<td>element subtype</td>
<td>integer</td>
<td>0 - same as whole map</td>
<td>Element subtype, if &quot;elementtype=3&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - custom size</td>
<td></td>
</tr>
<tr>
<td>width</td>
<td>area size</td>
<td>integer</td>
<td>0 - place evenly in the area</td>
<td>Area size, if &quot;areatype=1&quot;</td>
</tr>
<tr>
<td>height</td>
<td>area size</td>
<td>integer</td>
<td>1 - custom size</td>
<td></td>
</tr>
<tr>
<td>viewtype</td>
<td>area placement</td>
<td>integer</td>
<td>0 - place evenly in the area</td>
<td>Height of area, if &quot;areatype=1&quot;</td>
</tr>
<tr>
<td></td>
<td>algorithm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>use_iconmap</td>
<td>use icon mapping</td>
<td>integer</td>
<td>0 - no</td>
<td>Use icon mapping for this element. Relevant only if icon mapping is activated on map level.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - yes</td>
<td></td>
</tr>
<tr>
<td>elementid</td>
<td>element</td>
<td>id</td>
<td></td>
<td>Unique element record ID.</td>
</tr>
<tr>
<td>evaltype</td>
<td>problem type</td>
<td>integer</td>
<td>0 - place evenly in the area</td>
<td>Evaluation type for tags.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>problem tags</td>
<td>integer</td>
<td>0 - place evenly in the area</td>
<td>Problem tags (for host and host group elements). If tags are given, only problems with these tags will be displayed on the map.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - place evenly in the area</td>
<td></td>
</tr>
<tr>
<td>tag</td>
<td>tag name</td>
<td>string</td>
<td></td>
<td>Tag name.</td>
</tr>
<tr>
<td>value</td>
<td>tag value</td>
<td>string</td>
<td></td>
<td>Tag value.</td>
</tr>
<tr>
<td>operator</td>
<td>operator</td>
<td></td>
<td></td>
<td>Operator.</td>
</tr>
<tr>
<td>elements</td>
<td>zabbix entities</td>
<td></td>
<td></td>
<td>Zabbix entities that are represented on the map (host, host group, map etc).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>host</td>
<td>icon to use</td>
<td></td>
<td></td>
<td>Image to use when element is in 'OK' status.</td>
</tr>
<tr>
<td>icon_off</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>icon_on</td>
<td></td>
<td></td>
<td></td>
<td>Image to use when element is in 'Problem' status.</td>
</tr>
<tr>
<td>icon_disabled</td>
<td></td>
<td></td>
<td></td>
<td>Image to use when element is disabled.</td>
</tr>
<tr>
<td>icon_maintenance</td>
<td></td>
<td></td>
<td></td>
<td>Image to use when element is in maintenance.</td>
</tr>
<tr>
<td>name</td>
<td>unique image</td>
<td>string</td>
<td></td>
<td>Unique image name.</td>
</tr>
<tr>
<td>shapes</td>
<td>shape type</td>
<td>integer</td>
<td>0 - rectangle</td>
<td>Shape type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - ellipse</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>coordinates</td>
<td>integer</td>
<td></td>
<td>X coordinates of the shape in pixels.</td>
</tr>
<tr>
<td>y</td>
<td>coordinates</td>
<td>integer</td>
<td></td>
<td>Y coordinates of the shape in pixels.</td>
</tr>
<tr>
<td>width</td>
<td>shape width</td>
<td>integer</td>
<td></td>
<td>Shape width.</td>
</tr>
<tr>
<td>height</td>
<td>shape height</td>
<td>integer</td>
<td></td>
<td>Shape height.</td>
</tr>
<tr>
<td>border_type</td>
<td>type of border</td>
<td>integer</td>
<td>0 - none</td>
<td>Type of the border for the shape.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - bold line</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 - dotted line</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 - dashed line</td>
<td></td>
</tr>
<tr>
<td>border_width</td>
<td>width of border</td>
<td>integer</td>
<td></td>
<td>Width of the border in pixels.</td>
</tr>
<tr>
<td>border_color</td>
<td>color</td>
<td>string</td>
<td></td>
<td>Border color represented in hexadecimal code.</td>
</tr>
<tr>
<td>text</td>
<td>text inside</td>
<td>string</td>
<td></td>
<td>Text inside of shape.</td>
</tr>
<tr>
<td>Element property</td>
<td>Type</td>
<td>Range</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| `font`           | integer | 0 - "Georgia, serif"
1 - "Palatino Linotype",
"Book Antiqua",
Palatino, serif
2 - "Times New Roman",
Times, serif
3 - Arial, Helvetica,
sans-serif
4 - "Arial Black",
Gadget, sans-serif
5 - "Comic Sans MS",
cursive, sans-serif
6 - Impact, Charcoal,
sans-serif
7 - "Lucida Sans
Unicode", "Lucida
Grande", sans-serif
8 - Tahoma, Geneva,
sans-serif
9 - "Trebuchet MS",
Helvetica, sans-serif
10 - Verdana, Geneva,
sans-serif
11 - "Courier New",
Courier, monospace
12 - "Lucida Console",
Monaco, monospace | Text font style.                                                     |
| `font_size`      | integer |                                                                      | Font size in pixels.                                                        |
| `font_color`     | string  |                                                                      | Font color represented in hexadecimal code.                                |
| `text_halign`    | integer | 0 - center
1 - left
2 - right | Horizontal alignment of text.                                        |
| `text_valign`    | integer | 0 - middle
1 - top
2 - bottom | Vertical alignment of text.                                          |
| `background_color` | string  |                                                                      | Background (fill) color represented in hexadecimal code.                    |
| `zindex`         | integer |                                                                      | Value used to order all shapes and lines (z-index).                         |
| `x1`             | integer |                                                                      | X coordinates of the line point 1 in pixels.                                |
| `y1`             | integer |                                                                      | Y coordinates of the line point 1 in pixels.                                |
| `x2`             | integer |                                                                      | X coordinates of the line point 2 in pixels.                                |
| `y2`             | integer |                                                                      | Y coordinates of the line point 2 in pixels.                                |
| `line_type`      | integer | 0 - none
1 - bold line
2 - dotted line
3 - dashed line | Line type.                                                           |
| `line_width`     | integer |                                                                      | Line width in pixels.                                                       |
| `line_color`     | string  |                                                                      | Line color represented in hexadecimal code.                                |
| `zindex`         | integer |                                                                      | Value used to order all shapes and lines (z-index).                         |
| `drawtype`       | integer | 0 - line
2 - bold line
3 - dotted line
4 - dashed line | Link style.                                                          |
<p>| <code>color</code>          | string  |                                                                      | Link color (6 symbols, hex).                                               |
| <code>label</code>          | string  |                                                                      | Link label.                                                               |
| <code>selementid1</code>    | id      |                                                                      | ID of one element to connect.                                              |
| <code>selementid2</code>    | id      |                                                                      | ID of the other element to connect.                                        |
| <code>linktriggers</code>   |         |                                                                      | Link status indicators.                                                    |</p>
<table>
<thead>
<tr>
<th>Element property</th>
<th>Type</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>drawtype</td>
<td>integer</td>
<td>0 - line</td>
<td>Link style when trigger is in the ‘Problem’ state.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - bold line</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - dotted line</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - dashed line</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>string</td>
<td>Link color (6 symbols, hex) when trigger is in the ‘Problem’ state.</td>
<td></td>
</tr>
<tr>
<td>trigger</td>
<td></td>
<td>Trigger used for indicating link status.</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Trigger name.</td>
<td></td>
</tr>
<tr>
<td>expression</td>
<td>string</td>
<td>Trigger expression.</td>
<td></td>
</tr>
<tr>
<td>recovery_expression</td>
<td>string</td>
<td>Trigger recovery expression.</td>
<td></td>
</tr>
</tbody>
</table>

### 6 Media types

**Overview**

Media types are exported with all related objects and object relations.

**Exporting**

To export media types, do the following:

- Go to: Administration → Media types
- Mark the checkboxes of the media types to export
- Click on Export below the list

Depending on the selected format, media types are exported to a local file with a default name:

- `zabbix_export_mediatypes.yaml` - in YAML export (default option for export)
- `zabbix_export_mediatypes.xml` - in XML export
- `zabbix_export_mediatypes.json` - in JSON export

**Importing**

To import media types, do the following:

- Go to: Administration → Media types
• Click on Import to the right
• Select the import file
• Mark the required options in import rules
• Click on Import

A success or failure message of the import will be displayed in the frontend.

Import rules:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update existing</td>
<td>Existing elements will be updated with data taken from the import file. Otherwise they will not be updated.</td>
</tr>
<tr>
<td>Create new</td>
<td>The import will add new elements using data from the import file. Otherwise it will not add them.</td>
</tr>
<tr>
<td>Delete missing</td>
<td>The import will remove existing elements not present in the import file. Otherwise it will not remove them.</td>
</tr>
</tbody>
</table>

Export format

Export to YAML:

```yaml
zabbix_export:
  version: '6.2'
  date: '2021-08-31T13:34:17Z'
  media_types:
    - name: Pushover
      type: WEBHOOK
      parameters:
        - name: endpoint
          value: 'https://api.pushover.net/1/messages.json'
        - name: eventid
          value: '{EVENT.ID}'
        - name: event_nseverity
          value: '{EVENT.NSEVERITY}'
        - name: event_source
          value: '{EVENT.SOURCE}'
        - name: event_value
          value: '{EVENT.VALUE}'
        - name: expire
          value: '1200'
        - name: message
```

592
value: '{ALERT.MESSAGE}'
-
name: priority_average
value: '0'
-
name: priority_default
value: '0'
-
name: priority_disaster
value: '0'
-
name: priority_high
value: '0'
-
name: priority_information
value: '0'
-
name: priority_not_classified
value: '0'
-
name: priority_warning
value: '0'
-
name: retry
value: '60'
-
name: title
value: '{ALERT.SUBJECT}'
-
name: token
value: '<PUSHOVER TOKEN HERE>'
-
name: triggerid
value: '{TRIGGER.ID}'
-
name: url
value: '{$ZABBIX.URL}'
-
name: url_title
value: Zabbix
-
name: user
value: '{ALERT.SENDTO}'
max_sessions: '0'

script: |
  try {
    var params = JSON.parse(value),
    request = new HttpRequest(),
    data,
    response,
    severities = [
      {name: 'not_classified', color: '#97AAB3'},
      {name: 'information', color: '#7499FF'},
      {name: 'warning', color: '#FFC859'},
      {name: 'average', color: '#FFA059'},
      {name: 'high', color: '#E97659'},
      {name: 'disaster', color: '#E45959'},
      {name: 'resolved', color: '#009900'},
      {name: 'default', color: '#000000'}
    ],
    priority;
if (typeof params.HTTPProxy === 'string' && params.HTTPProxy.trim() !== '') {
    request.setProxy(params.HTTPProxy);
}

if ([0, 1, 2, 3].indexOf(parseInt(params.event_source)) === -1) {
    throw 'Incorrect "event_source" parameter given: "' + params.event_source + '"
Must be 0 or 1.
}

if (params.event_value !== '0' && params.event_value !== '1' && (params.event_source === '0' || params.event_source === '3')) {
    throw 'Incorrect "event_value" parameter given: ' + params.event_value + '
Must be 0 or 1.';
}

if ([0, 1, 2, 3, 4, 5].indexOf(parseInt(params.event_nseverity)) === -1) {
    params.event_nseverity = '7';
}

if (params.event_value === '0') {
    params.event_nseverity = '6';
}

priority = params['priority_' + severities[params.event_nseverity].name] || params.priority_default;

if (isNaN(priority) || priority < -2 || priority > 2) {
    throw "priority" should be -2..2';
}

if (params.event_source === '0' && isNaN(params.triggerid)) {
    throw 'field "triggerid" is not a number';
}

if (isNaN(params.eventid)) {
    throw 'field "eventid" is not a number';
}

if (typeof params.message !== 'string' || params.message.trim() === '') {
    throw 'field "message" cannot be empty';
}

data = {
    token: params.token,
    user: params.user,
    title: params.title,
    message: params.message,
    url: (params.event_source === '0')
        ? params.url + '/tr_events.php?triggerid=' + params.triggerid + '&eventid=' + params.eventid
        : params.url,
    url_title: params.url_title,
    priority: priority
};

if (priority == 2) {
    if (isNaN(params.retry) || params.retry < 30) {
        throw 'field "retry" should be a number with value of at least 30 if "priority" is set.
    }

    if (isNaN(params.expire) || params.expire > 10800) {
        throw 'field "expire" should be a number with value of at most 10800 if "priority" is set.
    }

    data.retry = params.retry;
    data.expire = params.expire;
data = JSON.stringify(data);
Zabbix.log(4, ' [ Pushover Webhook ] Sending request: ' + params.endpoint + '\n' + data);

request.addHeader('Content-Type: application/json');
response = request.post(params.endpoint, data);

Zabbix.log(4, ' [ Pushover Webhook ] Received response with status code ' + request.getStatus());

if (response !== null) {
  try {
    response = JSON.parse(response);
  }
  catch (error) {
    Zabbix.log(4, ' [ Pushover Webhook ] Failed to parse response received from Pushover');
    response = null;
  }
}

if (request.getStatus() != 200 || response === null || typeof response !== 'object' || response.status !== 1) {
  if (response !== null && typeof response === 'object' && typeof response.errors === 'object' && typeof response.errors[0] === 'string') {
    throw response.errors[0];
  } else {
    throw 'Unknown error. Check debug log for more information.';
  }
}

return 'OK';
}

} catch (error) {
  Zabbix.log(4, ' [ Pushover Webhook ] Pushover notification failed: ' + error);
  throw 'Pushover notification failed: ' + error;
}

description:
Please refer to setup guide here: https://git.zabbix.com/projects/ZBX/repos/zabbix/browse/templates/media/pushover

Set token parameter with to your Pushover application key.
When assigning Pushover media to the Zabbix user - add user key into send to field.

message_templates:

  - event_source: TRIGGERS
    operation_mode: PROBLEM
    subject: 'Problem: {EVENT.NAME}'
    message: |
      Problem started at {EVENT.TIME} on {EVENT.DATE}
      Problem name: {EVENT.NAME}
      Host: {HOST.NAME}
      Severity: {EVENT.SEVERITY}
      Operational data: {EVENT.OPDATA}
      Original problem ID: {EVENT.ID}
      {TRIGGER.URL}

  - event_source: TRIGGERS
    operation_mode: RECOVERY
    subject: 'Resolved in {EVENT.DURATION}: {EVENT.NAME}'
    message: |
      Problem has been resolved at {EVENT.RECOVERY.TIME} on {EVENT.RECOVERY.DATE}
      Problem name: {EVENT.NAME}
      Problem duration: {EVENT.DURATION}
Host: {HOST.NAME}
Severity: {EVENT.SEVERITY}
Original problem ID: {EVENT.ID}
{TRIGGER.URL}

- event_source: TRIGGERS
operation_mode: UPDATE
subject: 'Updated problem in {EVENT.AGE}: {EVENT.NAME}'
message: |
 {USER.FULLNAME} {EVENT.UPDATE.ACTION} problem at {EVENT.UPDATE.DATE} {EVENT.UPDATE.TIME}.
{EVENT.UPDATE.MESSAGE}

Current problem status is {EVENT.STATUS}, age is {EVENT.AGE}, acknowledged: {EVENT.ACK.STATUS}

- event_source: DISCOVERY
operation_mode: PROBLEM
subject: 'Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IPADDRESS}'
message: |
Discovery rule: {DISCOVERY.RULE.NAME}
Device IP: {DISCOVERY.DEVICE.IPADDRESS}
Device DNS: {DISCOVERY.DEVICE.DNS}
Device status: {DISCOVERY.DEVICE.STATUS}
Device uptime: {DISCOVERY.DEVICE.UPTIME}
Device service name: {DISCOVERY.SERVICE.NAME}
Device service port: {DISCOVERY.SERVICE.PORT}
Device service status: {DISCOVERY.SERVICE.STATUS}
Device service uptime: {DISCOVERY.SERVICE.UPTIME}

- event_source: AUTOREGISTRATION
operation_mode: PROBLEM
subject: 'Autoregistration: {HOST.HOST}'
message: |
Host name: {HOST.HOST}
Host IP: {HOST.IP}
Agent port: {HOST.PORT}

Element tags

Element tag values are explained in the table below.

<table>
<thead>
<tr>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>media_types</td>
<td>-</td>
<td></td>
<td>Root element for media_types.</td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Media type name.</td>
</tr>
<tr>
<td>type</td>
<td>x</td>
<td>string</td>
<td>Transport used by the media type.</td>
</tr>
<tr>
<td>status</td>
<td>-</td>
<td>string</td>
<td>Whether the media type is enabled.</td>
</tr>
<tr>
<td>max_sessions</td>
<td>-</td>
<td>integer</td>
<td>The maximum number of alerts that can be processed in parallel.</td>
</tr>
<tr>
<td>attempts</td>
<td>-</td>
<td>integer</td>
<td>The maximum number of attempts to send an alert.</td>
</tr>
<tr>
<td>attempt_interval</td>
<td>-</td>
<td>integer</td>
<td>The interval between retry attempts.</td>
</tr>
</tbody>
</table>

Accepts seconds and time unit with suffix.
<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Media type description.</td>
</tr>
<tr>
<td>message_templates</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for media type message templates.</td>
</tr>
<tr>
<td>event_source</td>
<td>x</td>
<td>string</td>
<td>0 - TRIGGERS, 1 - DISCOVERY, 2 - AUTOREGISTRATION, 3 - INTERNAL</td>
<td>Event source.</td>
</tr>
<tr>
<td>operation_mode</td>
<td>string</td>
<td></td>
<td>0 - PROBLEM, 1 - RECOVERY, 2 - UPDATE</td>
<td>Operation mode.</td>
</tr>
<tr>
<td>subject</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Message subject.</td>
</tr>
<tr>
<td>message</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Message body.</td>
</tr>
</tbody>
</table>

Used only by e-mail media type

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smtp_server</td>
<td>x</td>
<td>string</td>
<td></td>
<td>SMTP server.</td>
</tr>
<tr>
<td>smtp_port</td>
<td>-</td>
<td>integer</td>
<td>Default: 25</td>
<td>SMTP server port to connect to.</td>
</tr>
<tr>
<td>smtp_helo</td>
<td>x</td>
<td>string</td>
<td></td>
<td>SMTP helo.</td>
</tr>
<tr>
<td>smtp_email</td>
<td>x</td>
<td>string</td>
<td>0 - NONE (default), 1 - STARTTLS, 2 - SSL_OR_TLS</td>
<td>Email address from which notifications will be sent.</td>
</tr>
<tr>
<td>smtp_security</td>
<td>-</td>
<td>string</td>
<td>0 - NONE (default), 1 - STARTTLS, 2 - SSL_OR_TLS</td>
<td>SMTP connection security level to use.</td>
</tr>
<tr>
<td>smtp_verify_host</td>
<td>x</td>
<td>string</td>
<td>0 - NO (default), 1 - YES</td>
<td>SSL verify host for SMTP. Optional if smtp_security is STARTTLS or SSL_OR_TLS.</td>
</tr>
<tr>
<td>smtp_verify_peer</td>
<td>x</td>
<td>string</td>
<td>0 - NO (default), 1 - YES</td>
<td>SSL verify peer for SMTP. Optional if smtp_security is STARTTLS or SSL_OR_TLS.</td>
</tr>
<tr>
<td>smtp_authentication</td>
<td>string</td>
<td></td>
<td>0 - NONE (default), 1 - PASSWORD</td>
<td>SMTP authentication method to use.</td>
</tr>
<tr>
<td>username</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Username.</td>
</tr>
<tr>
<td>password</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Authentication password.</td>
</tr>
<tr>
<td>content_type</td>
<td>-</td>
<td>string</td>
<td>0 - TEXT, 1 - HTML (default)</td>
<td>Message format.</td>
</tr>
</tbody>
</table>

Used only by SMS media type

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gsm_modem</td>
<td>x</td>
<td>string</td>
<td></td>
<td>Serial device name of the GSM modem.</td>
</tr>
</tbody>
</table>

Used only by script media type

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameters</td>
<td>-</td>
<td></td>
<td></td>
<td>Root element for script parameters.</td>
</tr>
</tbody>
</table>

Used only by webhook media type

<table>
<thead>
<tr>
<th>Element</th>
<th>Element property</th>
<th>RequiredType</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>script</td>
<td>x</td>
<td>string</td>
<td>1-60s (default: 30s)</td>
<td>Javascript script HTTP request timeout interval.</td>
</tr>
<tr>
<td>timeout</td>
<td>-</td>
<td>string</td>
<td></td>
<td>Whether to process returned tags.</td>
</tr>
<tr>
<td>process_tags</td>
<td>-</td>
<td>string</td>
<td>0 - NO (default), 1 - YES</td>
<td>If {EVENT.TAGS.*} were successfully resolved in event_menu_url and event_menu_name fields, this field indicates presence of entry in the event menu.</td>
</tr>
<tr>
<td>show_event_menu</td>
<td>string</td>
<td></td>
<td>0 - NO (default), 1 - YES</td>
<td>URL of the event menu entry. Supports {EVENT.TAGS.*} macro.</td>
</tr>
<tr>
<td>event_menu_url</td>
<td>string</td>
<td></td>
<td></td>
<td>Name of the event menu entry. Supports {EVENT.TAGS.*} macro.</td>
</tr>
<tr>
<td>Element property</td>
<td>RequiredType</td>
<td>Range</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>-------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>parameters</td>
<td>-</td>
<td>-</td>
<td>Root element for webhook media type parameters.</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>x</td>
<td>string</td>
<td>Webhook parameter name.</td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>-</td>
<td>string</td>
<td>Webhook parameter value.</td>
<td></td>
</tr>
</tbody>
</table>

Footnotes

1 For string values, only the string will be exported (e.g. "EMAIL") without the numbering used in this table. The numbers for range values (corresponding to the API values) in this table is used for ordering only.

15. Discovery

Please use the sidebar to access content in the Discovery section.

1 Network discovery

Overview

Zabbix offers automatic network discovery functionality that is effective and very flexible.

With network discovery properly set up you can:

• speed up Zabbix deployment
• simplify administration
• use Zabbix in rapidly changing environments without excessive administration

Zabbix network discovery is based on the following information:

• IP ranges
• Availability of external services (FTP, SSH, WEB, POP3, IMAP, TCP, etc)
• Information received from Zabbix agent (only unencrypted mode is supported)
• Information received from SNMP agent

It does NOT provide:

• Discovery of network topology

Network discovery basically consists of two phases: discovery and actions.

Discovery

Zabbix periodically scans the IP ranges defined in network discovery rules. The frequency of the check is configurable for each rule individually.

Note that one discovery rule will always be processed by a single discoverer process. The IP range will not be split between multiple discoverer processes.

Each rule has a set of service checks defined to be performed for the IP range.

Discovery checks are processed independently from the other checks. If any checks do not find a service (or fail), other checks will still be processed.

Every check of a service and a host (IP) performed by the network discovery module generates a discovery event.

<table>
<thead>
<tr>
<th>Event</th>
<th>Check of service result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Discovered</td>
<td>The service is ‘up’ after it was ‘down’ or when discovered for the first time.</td>
</tr>
<tr>
<td>Service Up</td>
<td>The service is ‘up’, after it was already ‘up’.</td>
</tr>
<tr>
<td>Service Lost</td>
<td>The service is ‘down’ after it was ‘up’.</td>
</tr>
<tr>
<td>Service Down</td>
<td>The service is ‘down’, after it was already ‘down’.</td>
</tr>
<tr>
<td>Host Discovered</td>
<td>At least one service of a host is ‘up’ after all services of that host were ‘down’ or a service is discovered which belongs to a not registered host.</td>
</tr>
<tr>
<td>Host Up</td>
<td>At least one service of a host is ‘up’, after at least one service was already ‘up’.</td>
</tr>
<tr>
<td>Event</td>
<td>Check of service result</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>Host Lost</td>
<td>All services of a host are ‘down’ after at least one was ‘up’</td>
</tr>
<tr>
<td>Host Down</td>
<td>All services of a host are ‘down’, after they were already ‘down’</td>
</tr>
</tbody>
</table>

**Actions**

Discovery events can be the basis of relevant actions, such as:

- Sending notifications
- Adding/removing hosts
- Enabling/disabling hosts
- Adding hosts to a group
- Removing hosts from a group
- Linking hosts to/unlinking from a template
- Executing remote scripts

These actions can be configured with respect to the device type, IP, status, uptime/downtime, etc. For full details on configuring actions for network-discovery based events, see action operation and conditions pages.

Since network discovery actions are event-based, they will be triggered both when a discovered host is online and when it is offline.

It is highly recommended to add an action condition Discovery status: up to avoid such actions as Add host being triggered upon Service Lost/Service Down events. Otherwise, if a discovered host is manually removed, it will still generate Service Lost/Service Down events and will be recreated during the next discovery cycle.

Linking a discovered host to templates will fail collectively if any of the linkable templates has a unique entity (e.g. item key) that is the same as a unique entity (e.g. item key) already existing on the host or on another of the linkable templates.

**Host creation**

A host is added if the Add host operation is selected. A host is also added, even if the Add host operation is missing, if you select operations resulting in actions on a host. Such operations are:

- enable host
- disable host
- add host to a host group
- link template to a host

Created hosts are added to the Discovered hosts group (by default, configurable in Administration → General → Other). If you wish hosts to be added to another group, add a Remove from host groups operation (specifying “Discovered hosts”) and also add an Add to host groups operation (specifying another host group), because a host must belong to a host group.

**Host naming**

When adding hosts, a host name is the result of reverse DNS lookup or IP address if reverse lookup fails. Lookup is performed from the Zabbix server or Zabbix proxy, depending on which is doing the discovery. If lookup fails on the proxy, it is not retried on the server. If the host with such a name already exists, the next host would get _2 appended to the name, then _3 and so on.

It is also possible to override DNS/IP lookup and instead use an item value for host name, for example:

- You may discover multiple servers with Zabbix agent running using a Zabbix agent item for discovery and assign proper names to them automatically, based on the string value returned by this item
- You may discover multiple SNMP network devices using an SNMP agent item for discovery and assign proper names to them automatically, based on the string value returned by this item

If the host name has been set using an item value, it is not updated during the following discovery checks. If it is not possible to set host name using an item value, default value (DNS name) is used.

If a host already exists with the discovered IP address, a new host is not created. However, if the discovery action contains operations (link template, add to host group, etc), they are performed on the existing host.

**Host removal**

Hosts discovered by a network discovery rule are removed automatically from Monitoring → Discovery if a discovered entity is not in the rule’s IP range any more. Hosts are removed immediately.

**Interface creation when adding hosts**

When hosts are added as a result of network discovery, they get interfaces created according to these rules:

- the services detected - for example, if an SNMP check succeeded, an SNMP interface will be created
- if a host responded both to Zabbix agent and SNMP requests, both types of interfaces will be created
• if uniqueness criteria are Zabbix agent or SNMP-returned data, the first interface found for a host will be created as the default one. Other IP addresses will be added as additional interfaces. Action’s conditions (such as Host IP) do not impact adding interfaces. Note that this will work if all interfaces are discovered by the same discovery rule. If a different discovery rule discovers a different interface of the same host, an additional host will be added.
• if a host responded to agent checks only, it will be created with an agent interface only. If it would start responding to SNMP later, additional SNMP interfaces would be added.
• if 3 separate hosts were initially created, having been discovered by the “IP” uniqueness criteria, and then the discovery rule is modified so that hosts A, B and C have identical uniqueness criteria result, B and C are created as additional interfaces for A, the first host. The individual hosts B and C remain. In Monitoring → Discovery the added interfaces will be displayed in the “Discovered device” column, in black font and indented, but the ”Monitored host” column will only display A, the first created host. ”Uptime/Downtime” is not measured for IPs that are considered to be additional interfaces.

Changing proxy setting

The hosts discovered by different proxies are always treated as different hosts. While this allows to perform discovery on matching IP ranges used by different subnets, changing proxy for an already monitored subnet is complicated because the proxy changes must be also applied to all discovered hosts.

For example the steps to replace proxy in a discovery rule:

1. disable discovery rule
2. sync proxy configuration
3. replace the proxy in the discovery rule
4. replace the proxy for all hosts discovered by this rule
5. enable discovery rule

1 Configuring a network discovery rule

Overview

To configure a network discovery rule used by Zabbix to discover hosts and services:

• Go to Configuration → Discovery
• Click on Create rule (or on the rule name to edit an existing one)
• Edit the discovery rule attributes

Rule attributes
All mandatory input fields are marked with a red asterisk.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Unique name of the rule. For example, “Local network”.</td>
</tr>
<tr>
<td>Discovery by proxy</td>
<td>What performs discovery:</td>
</tr>
<tr>
<td></td>
<td><strong>no proxy</strong> - Zabbix server is doing discovery</td>
</tr>
<tr>
<td></td>
<td><strong>&lt;proxy name&gt;</strong> - this proxy performs discovery</td>
</tr>
<tr>
<td>IP range</td>
<td>The range of IP addresses for discovery. It may have the following formats:</td>
</tr>
<tr>
<td></td>
<td>Single IP: 192.168.1.33</td>
</tr>
<tr>
<td></td>
<td>Range of IP addresses: 192.168.1.10-1.255. The range is limited by the total number of covered addresses (less than 64K).</td>
</tr>
<tr>
<td></td>
<td>IP mask: 192.168.4.0/24</td>
</tr>
<tr>
<td></td>
<td>supported IP masks: /16 - /30 for IPv4 addresses /112 - /128 for IPv6 addresses</td>
</tr>
<tr>
<td></td>
<td>List: 192.168.1.1-255, 192.168.2.1-100, 192.168.2.200, 192.168.4.0/24</td>
</tr>
<tr>
<td></td>
<td>Since Zabbix 3.0.0 this field supports spaces, tabulation and multiple lines.</td>
</tr>
<tr>
<td>Update interval</td>
<td>This parameter defines how often Zabbix will execute the rule. The interval is measured after the execution of previous discovery instance ends so there is no overlap.</td>
</tr>
<tr>
<td></td>
<td><em>Time suffixes</em> are supported, e.g. 30s, 1m, 2h, 1d, since Zabbix 3.4.0.</td>
</tr>
<tr>
<td></td>
<td><em>User macros</em> are supported, since Zabbix 3.4.0.</td>
</tr>
<tr>
<td></td>
<td>Note that if a user macro is used and its value is changed (e.g. 1w → 1h), the next check will be executed according to the previous value (far in the future with the example values).</td>
</tr>
<tr>
<td>Checks</td>
<td>Zabbix will use this list of checks for discovery. Click on <strong>Add</strong> to configure a new check in a popup window.</td>
</tr>
<tr>
<td></td>
<td>Supported checks: SSH, LDAP, SMTP, FTP, HTTP, HTTPS, POP, NNTP, IMAP, TCP, Telnet, Zabbix agent, SNMPv1 agent, SNMPv2 agent, SNMPv3 agent, ICMP ping.</td>
</tr>
<tr>
<td></td>
<td>A protocol-based discovery uses the <strong>net.tcp.service[]</strong> functionality to test each host, except for SNMP which queries an SNMP OID. Zabbix agent is tested by querying an item in unencrypted mode. Please see <em>agent items</em> for more details.</td>
</tr>
<tr>
<td></td>
<td>The ‘Ports’ parameter may be one of following:</td>
</tr>
<tr>
<td></td>
<td>Single port: 22</td>
</tr>
<tr>
<td></td>
<td>Range of ports: 22-45</td>
</tr>
<tr>
<td></td>
<td>List: 22-45,55,60-70</td>
</tr>
<tr>
<td>Device uniqueness criteria</td>
<td>Uniqueness criteria may be:</td>
</tr>
<tr>
<td></td>
<td><strong>IP address</strong> - no processing of multiple single-IP devices. If a device with the same IP already exists it will be considered already discovered and a new host will not be added.</td>
</tr>
<tr>
<td></td>
<td><strong>&lt;discovery check&gt;</strong> - either Zabbix agent or SNMP agent check.</td>
</tr>
<tr>
<td>Host name</td>
<td>Set the technical host name of a created host using:</td>
</tr>
<tr>
<td></td>
<td><strong>DNS name</strong> - DNS name (default)</td>
</tr>
<tr>
<td></td>
<td><strong>IP address</strong> - IP address</td>
</tr>
<tr>
<td></td>
<td><strong>&lt;discovery check&gt;</strong> - received string value of the discovery check (e.g. Zabbix agent, SNMP agent check)</td>
</tr>
<tr>
<td></td>
<td>See also: <em>Host naming</em>.</td>
</tr>
<tr>
<td></td>
<td>This option is supported since 4.2.0.</td>
</tr>
<tr>
<td>Visible name</td>
<td>Set the visible host name of a created host using:</td>
</tr>
<tr>
<td></td>
<td><strong>Host name</strong> - technical host name (default)</td>
</tr>
<tr>
<td></td>
<td><strong>DNS name</strong> - DNS name</td>
</tr>
<tr>
<td></td>
<td><strong>IP address</strong> - IP address</td>
</tr>
<tr>
<td></td>
<td><strong>&lt;discovery check&gt;</strong> - received string value of the discovery check (e.g. Zabbix agent, SNMP agent check)</td>
</tr>
<tr>
<td></td>
<td>See also: <em>Host naming</em>.</td>
</tr>
<tr>
<td></td>
<td>This option is supported since 4.2.0.</td>
</tr>
<tr>
<td>Enabled</td>
<td>With the check-box marked the rule is active and will be executed by Zabbix server.</td>
</tr>
<tr>
<td></td>
<td>If unmarked, the rule is not active. It won’t be executed.</td>
</tr>
</tbody>
</table>

A real life scenario

In this example, we would like to set up network discovery for the local network having an IP range of 192.168.1.1-192.168.1.254.

In our scenario we want to:

- discover those hosts that have Zabbix agent running
- run discovery every 10 minutes
• add a host to monitoring if the host uptime is more than 1 hour
• remove hosts if the host downtime is more than 24 hours
• add Linux hosts to the “Linux servers” group
• add Windows hosts to the “Windows servers” group
• use the template Linux for Linux hosts
• use the template Windows for Windows hosts

Step 1

Defining a network discovery rule for our IP range.

Zabbix will try to discover hosts in the IP range of 192.168.1.1-192.168.1.254 by connecting to Zabbix agents and getting the value from the system.uname key. The value received from the agent can be used to name the hosts and also to apply different actions for different operating systems. For example, link Windows servers to the template Windows, Linux servers to the template Linux.

The rule will be executed every 10 minutes.

When this rule is added, Zabbix will automatically start the discovery and generation of the discovery-based events for further
Step 2

Defining a discovery action for adding the discovered Linux servers to the respective group/template.

<table>
<thead>
<tr>
<th>Action</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Add discovered Linux servers</td>
</tr>
<tr>
<td>Type of calculation</td>
<td>And</td>
</tr>
<tr>
<td>A</td>
<td>Received value contains Linux</td>
</tr>
<tr>
<td>B</td>
<td>Discovery status equals Up</td>
</tr>
<tr>
<td>C</td>
<td>Service type equals Zabbix agent</td>
</tr>
<tr>
<td>D</td>
<td>Uptime/Downtime is greater than or equals 3600</td>
</tr>
<tr>
<td>Add</td>
<td></td>
</tr>
</tbody>
</table>

The action will be activated if:
- the “Zabbix agent” service is “up”
- the value of system.uname (the Zabbix agent key we used in rule definition) contains “Linux”
- Uptime is 1 hour (3600 seconds) or more

<table>
<thead>
<tr>
<th>Action</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default subject</td>
<td>Discovery: (DISCOVERY.DEVICE.STATUS), (DISCOVERY.DEVICE.IPADDRESS)</td>
</tr>
<tr>
<td>Default message</td>
<td>Discovery rule: (DISCOVERY.RULE.NAME)</td>
</tr>
<tr>
<td>Operations</td>
<td>Details</td>
</tr>
<tr>
<td>Add to host groups</td>
<td>Linux servers</td>
</tr>
<tr>
<td>Link to templates</td>
<td>Linux</td>
</tr>
<tr>
<td>Add</td>
<td></td>
</tr>
</tbody>
</table>

The action will execute the following operations:
- add the discovered host to the “Linux servers” group (and also add host if it wasn’t added previously)
- link host to the Linux template. Zabbix will automatically start monitoring the host using items and triggers from the “Linux” template.

Step 3

Defining a discovery action for adding the discovered Windows servers to the respective group/template.
### Step 4

Defining a discovery action for removing lost servers.

<table>
<thead>
<tr>
<th>Action</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Add discovered Windows servers</td>
</tr>
<tr>
<td>Type of calculation</td>
<td>And A and B and C and D</td>
</tr>
<tr>
<td>Conditions</td>
<td>Label</td>
</tr>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default subject</td>
<td>Discovery: <code>{DISCOVERY.DEVICE.STATUS}</code>, <code>{DISCOVERY.DEVICE.IPADDRESS}</code></td>
</tr>
<tr>
<td>Default message</td>
<td>Discovery rule: <code>{DISCOVERY.RULE.NAME}</code></td>
</tr>
<tr>
<td></td>
<td>Device IP: <code>{DISCOVERY.DEVICE.IPADDRESS}</code></td>
</tr>
<tr>
<td></td>
<td>Device DNS: <code>{DISCOVERY.DEVICE.DNS}</code></td>
</tr>
<tr>
<td></td>
<td>Device status: <code>{DISCOVERY.DEVICE.STATUS}</code></td>
</tr>
<tr>
<td></td>
<td>Device uptime: <code>{DISCOVERY.DEVICE.UPTIME}</code></td>
</tr>
<tr>
<td></td>
<td>Device service name: <code>{DISCOVERY.SERVICE.NAME}</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operations</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add to host groups</td>
<td>Windows servers</td>
</tr>
<tr>
<td>Link to templates</td>
<td>Windows</td>
</tr>
</tbody>
</table>

Add
A server will be removed if “Zabbix agent” service is ‘down’ for more than 24 hours (86400 seconds).

2 Active agent autoregistration

Overview

It is possible to allow active Zabbix agent autoregistration, after which the server can start monitoring them. This way new hosts can be added for monitoring without configuring them manually on the server.

Autoregistration can happen when a previously unknown active agent asks for checks.

The feature might be very handy for automatic monitoring of new Cloud nodes. As soon as you have a new node in the Cloud Zabbix will automatically start the collection of performance and availability data of the host.

Active agent autoregistration also supports the monitoring of added hosts with passive checks. When the active agent asks for checks, providing it has the ‘ListenIP’ or ‘ListenPort’ configuration parameters defined in the configuration file, these are sent along to the server. (If multiple IP addresses are specified, the first one is sent to the server.)

Server, when adding the new autoregistered host, uses the received IP address and port to configure the agent. If no IP address value is received, the one used for the incoming connection is used. If no port value is received, 10050 is used.

It is possible to specify that the host should be autoregistered with a DNS name as the default agent interface.
Autoregistration is rerun:

- if host *metadata* information changes:
  - due to HostMetadata changed and agent restarted
  - due to value returned by HostMetadataItem changed
- for manually created hosts with metadata missing
- if a host is manually changed to be monitored by another Zabbix proxy
- if autoregistration for the same host comes from a new Zabbix proxy

Configuration

Specify server

Make sure you have the Zabbix server identified in the agent *configuration file* - `zabbix_agentd.conf`

```
ServerActive=10.0.0.1
```

Unless you specifically define a Hostname in `zabbix_agentd.conf`, the system hostname of agent location will be used by server for naming the host. The system hostname in Linux can be obtained by running the ‘hostname’ command.

If Hostname is defined in Zabbix agent configuration as a comma-delimited list of hosts, hosts will be created for all listed host-names.

Restart the agent after making any changes to the configuration file.

Action for active agent autoregistration

When server receives an autoregistration request from an agent it calls an *action*. An action of event source “Autoregistration” must be configured for agent autoregistration.

Setting up network discovery is not required to have active agents autoregister.

In the Zabbix frontend, go to Configuration → Actions, select Autoregistration as the event source and click on Create action:

- In the Action tab, give your action a name
- Optionally specify *conditions*. You can do a substring match or regular expression match in the conditions for host name/host metadata. If you are going to use the “Host metadata” condition, see the next section.
- In the Operations tab, add relevant operations, such as - ‘Add host’, ‘Add to host group’ (for example, Discovered hosts), ‘Link to templates’, etc.

If the hosts that will be autoregistering are likely to be supported for active monitoring only (such as hosts that are firewalled from your Zabbix server) then you might want to create a specific template like Template_Linux-active to link to.

Created hosts are added to the Discovered hosts group (by default, configurable in Administration → General → Other). If you wish hosts to be added to another group, add a Remove from host group operation (specifying “Discovered hosts”) and also add an Add to host group operation (specifying another host group), because a host must belong to a host group.

Secure autoregistration

A secure way of autoregistration is possible by configuring PSK-based authentication with encrypted connections.

The level of encryption is configured globally in Administration → General, in the Autoregistration section accessible through the dropdown to the right. It is possible to select no encryption, TLS encryption with PSK authentication or both (so that some hosts may register without encryption while others through encryption).

Authentication by PSK is verified by Zabbix server before adding a host. If successful, the host is added and Connections from/to host are set to ‘PSK’ only with identity/pre-shared key the same as in the global autoregistration setting.

To ensure security of autoregistration on installations using proxies, encryption between Zabbix server and proxy should be enabled.

Using DNS as default interface

HostInterface and HostInterfaceItem *configuration parameters* allow to specify a custom value for the host interface during autoregistration.

More specifically, they are useful if the host should be autoregistered with a DNS name as the default agent interface rather than its IP address. In that case the DNS name should be specified or returned as the value of either HostInterface or HostInterfaceItem parameters. Note that if the value of one of the two parameters changes, the autoregistered host interface is updated. So it is possible to update the default interface to another DNS name or update it to an IP address. For the changes to take effect though, the agent has to be restarted.

If HostInterface or HostInterfaceItem parameters are not configured, the listen_dns parameter is resolved from the IP address. If such resolving is configured incorrectly, it may break autoregistration because of invalid hostname.

Using host metadata
When agent is sending an autoregistration request to the server it sends its hostname. In some cases (for example, Amazon cloud nodes) a hostname is not enough for Zabbix server to differentiate discovered hosts. Host metadata can be optionally used to send other information from an agent to the server.

Host metadata is configured in the agent configuration file - `zabbix_agentd.conf`. There are 2 ways of specifying host metadata in the configuration file:

- `HostMetadata`
- `HostMetadataItem`

See the description of the options in the link above.

An autoregistration attempt happens every time an active agent sends a request to refresh active checks to the server. The delay between requests is specified in the `RefreshActiveChecks` parameter of the agent. The first request is sent immediately after the agent is restarted.

Example 1

Using host metadata to distinguish between Linux and Windows hosts.

Say you would like the hosts to be autoregistered by the Zabbix server. You have active Zabbix agents (see “Configuration” section above) on your network. There are Windows hosts and Linux hosts on your network and you have “Linux by Zabbix agent” and “Windows by Zabbix agent” templates available in your Zabbix frontend. So at host registration, you would like the appropriate Linux/Windows template to be applied to the host being registered. By default, only the hostname is sent to the server at autoregistration, which might not be enough. In order to make sure the proper template is applied to the host you should use host metadata.

Frontend configuration

The first thing to do is to configure the frontend. Create 2 actions. The first action:

- **Name:** Linux host autoregistration
- **Conditions:** Host metadata contains Linux
- **Operations:** Link to templates: Linux

You can skip an “Add host” operation in this case. Linking to a template requires adding a host first so the server will do that automatically.

The second action:

- **Name:** Windows host autoregistration
- **Conditions:** Host metadata contains Windows
- **Operations:** Link to templates: Windows

Agent configuration

Now you need to configure the agents. Add the next line to the agent configuration files:

```
HostMetadataItem=system.uname
```

This way you make sure host metadata will contain “Linux” or “Windows” depending on the host an agent is running on. An example of host metadata in this case:

Linux: Linux server3 3.2.0-4-686-pae #1 SMP Debian 3.2.41-2 i686 GNU/Linux
Windows: Windows WIN-0PXGGSTYNHO 6.0.6001 Windows Server 2008 Service Pack 1 Intel IA-32

Do not forget to restart the agent after making any changes to the configuration file.

Example 2

Step 1

Using host metadata to allow some basic protection against unwanted hosts registering.

Frontend configuration

Create an action in the frontend, using some hard-to-guess secret code to disallow unwanted hosts:

- **Name:** Autoregistration action Linux
- **Conditions:**
  - *Type of calculation:* AND
  - *Condition (A):* Host metadata contains //Linux//
  - *Condition (B):* Host metadata contains //21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e//
- **Operations:**
  - *Send message to users:* Admin via all media
* Add to host groups: Linux servers
* Link to templates: Linux

Please note that this method alone does not provide strong protection because data is transmitted in plain text. Configuration cache reload is required for changes to have an immediate effect.

Agent configuration

Add the next line to the agent configuration file:

```
HostMetadata=Linux 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
```

where “Linux” is a platform, and the rest of the string is the hard-to-guess secret text.

Do not forget to restart the agent after making any changes to the configuration file.

**Step 2**

It is possible to add additional monitoring for an already registered host.

Frontend configuration

Update the action in the frontend:

- **Name:** Autoregistration action Linux
- **Conditions:**
  - **Type of calculation:** AND
  - **Condition (A):** Host metadata contains Linux
  - **Condition (B):** Host metadata contains 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
- **Operations:**
  - Send message to users: Admin via all media
  - Add to host groups: Linux servers
  - Link to templates: Linux
  - Link to templates: MySQL by Zabbix Agent

Agent configuration

Update the next line in the agent configuration file:

```
HostMetadata=MySQL on Linux 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
```

Do not forget to restart the agent after making any changes to the configuration file.

## 3 Low-level discovery

### Overview

Low-level discovery provides a way to automatically create items, triggers, and graphs for different entities on a computer. For instance, Zabbix can automatically start monitoring file systems or network interfaces on your machine, without the need to create items for each file system or network interface manually. Additionally, it is possible to configure Zabbix to remove unneeded entities automatically based on actual results of periodically performed discovery.

A user can define their own types of discovery, provided they follow a particular JSON protocol.

The general architecture of the discovery process is as follows.

First, a user creates a discovery rule in “Configuration” → “Templates” → “Discovery” column. A discovery rule consists of (1) an item that discovers the necessary entities (for instance, file systems or network interfaces) and (2) prototypes of items, triggers, and graphs that should be created based on the value of that item.

An item that discovers the necessary entities is like a regular item seen elsewhere: the server asks a Zabbix agent (or whatever the type of the item is set to) for a value of that item, the agent responds with a textual value. The difference is that the value the agent responds with should contain a list of discovered entities in a JSON format. While the details of this format are only important for implementers of custom discovery checks, it is necessary to know that the returned value contains a list of macro → value pairs. For instance, item “net.if.discovery” might return two pairs: “{#IFNAME}” → “lo” and “{#IFNAME}” → “eth0”.

These macros are used in names, keys and other prototype fields where they are then substituted with the received values for creating real items, triggers, graphs or even hosts for each discovered entity. See the full list of options for using LLD macros.

When the server receives a value for a discovery item, it looks at the macro → value pairs and for each pair generates real items, triggers, and graphs, based on their prototypes. In the example with “net.if.discovery” above, the server would generate one set of items, triggers, and graphs for the loopback interface “lo”, and another set for interface “eth0”.

609
Note that since Zabbix 4.2, the format of the JSON returned by low-level discovery rules has been changed. It is no longer expected that the JSON will contain the "data" object. Low-level discovery will now accept a normal JSON containing an array, in order to support new features such as the item value preprocessing and custom paths to low-level discovery macro values in a JSON document.

Built-in discovery keys have been updated to return an array of LLD rows at the root of JSON document. Zabbix will automatically extract a macro and value if an array field uses the {#MACRO} syntax as a key. Any new native discovery checks will use the new syntax without the "data" elements. When processing a low-level discovery value first the root is located (array at $. or $.data).

While the "data" element has been removed from all native items related to discovery, for backward compatibility Zabbix will still accept the JSON notation with a "data" element, though its use is discouraged. If the JSON contains an object with only one "data" array element, then it will automatically extract the content of the element using JSONPath $.data. Low-level discovery now accepts optional user-defined LLD macros with a custom path specified in JSONPath syntax.

As a result of the changes above, newer agents no longer will be able to work with an older Zabbix server.

See also: Discovered entities

**Configuring low-level discovery** We will illustrate low-level discovery based on an example of file system discovery.

To configure the discovery, do the following:

- Go to: Configuration → Templates or Hosts
- Click on Discovery in the row of an appropriate template/host
- Click on Create discovery rule in the upper right corner of the screen
- Fill in the discovery rule form with the required details

**Discovery rule**

The discovery rule form contains five tabs, representing, from left to right, the data flow during discovery:

- Discovery rule - specifies, most importantly, the built-in item or custom script to retrieve discovery data
- Preprocessing - applies some preprocessing to the discovered data
- LLD macros - allows to extract some macro values to use in discovered items, triggers, etc
- Filters - allows to filter the discovered values
- Overrides - allows to modify items, triggers, graphs or host prototypes when applying to specific discovered objects

The **Discovery rule** tab contains the item key to use for discovery (as well as some general discovery rule attributes):
All mandatory input fields are marked with a red asterisk.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of discovery rule.</td>
</tr>
<tr>
<td>Type</td>
<td>The type of check to perform discovery.</td>
</tr>
<tr>
<td></td>
<td>In this example we are using a Zabbix agent item key.</td>
</tr>
<tr>
<td></td>
<td>The discovery rule can also be a dependent item, depending on a regular item.</td>
</tr>
<tr>
<td></td>
<td>It cannot depend on another discovery rule. For a dependent item, select the</td>
</tr>
<tr>
<td></td>
<td>respective type (Dependent item) and specify the master item in the 'Master</td>
</tr>
<tr>
<td></td>
<td>item' field. The master item must exist.</td>
</tr>
<tr>
<td>Key</td>
<td>Enter the discovery item key (up to 2048 characters).</td>
</tr>
<tr>
<td></td>
<td>For example, you may use the built-in &quot;vfs.fs.discovery&quot; item key to return</td>
</tr>
<tr>
<td></td>
<td>a JSON with the list of file systems present on the computer and their types.</td>
</tr>
<tr>
<td></td>
<td>Note that another option for filesystem discovery is using discovery results</td>
</tr>
<tr>
<td></td>
<td>by the &quot;vfs.fs.get&quot; agent key, supported since Zabbix 4.4.5 (see example).</td>
</tr>
<tr>
<td>Update interval</td>
<td>This field specifies how often Zabbix performs discovery. In the beginning,</td>
</tr>
<tr>
<td></td>
<td>when you are just setting up file system discovery, you might wish to set it</td>
</tr>
<tr>
<td></td>
<td>to a small interval, but once you know it works you can set it to 30 minutes</td>
</tr>
<tr>
<td></td>
<td>or more, because file systems usually do not change very often.</td>
</tr>
<tr>
<td></td>
<td>Time suffixes are supported, e.g. 30s, 1m, 2h, 1d, since Zabbix 3.4.0.</td>
</tr>
<tr>
<td></td>
<td>User macros are supported, since Zabbix 3.4.0.</td>
</tr>
<tr>
<td></td>
<td>Note: The update interval can only be set to '0' if custom intervals exist</td>
</tr>
<tr>
<td></td>
<td>with a non-zero value. If set to '0', and a custom interval (flexible or</td>
</tr>
<tr>
<td></td>
<td>scheduled) exists with a non-zero value, the item will be polled during the</td>
</tr>
<tr>
<td></td>
<td>custom interval duration. New discovery rules will be checked within 60</td>
</tr>
<tr>
<td></td>
<td>seconds of their creation, unless they have Scheduling or Flexible update</td>
</tr>
<tr>
<td></td>
<td>interval and the Update interval is set to 0.</td>
</tr>
<tr>
<td></td>
<td>Note that for an existing discovery rule the discovery can be performed immediately by pushing the Execute now button.</td>
</tr>
</tbody>
</table>
Parameter Description

Custom intervals You can create custom rules for checking the item:

- **Flexible** - create an exception to the Update interval (interval with different frequency)
- **Scheduling** - create a custom polling schedule.

For detailed information see Custom intervals. Scheduling is supported since Zabbix 3.0.0.

Keep lost resources period This field allows you to specify the duration for how long the discovered entity will be retained (won’t be deleted) once its discovery status becomes "Not discovered anymore” (between 1 hour to 25 years; or "0").

- **Time suffixes** are supported, e.g. 2h, 1d, since Zabbix 3.4.0.
- **User macros** are supported, since Zabbix 3.4.0.

Note: If set to "0", entities will be deleted immediately. Using "0" is not recommended, since just wrongly editing the filter may end up in the entity being deleted with all the historical data.

Description Enter a description.

Enabled If checked, the rule will be processed.

---

Discovery rule history is not preserved.

Preprocessing

The **Preprocessing** tab allows to define transformation rules to apply to the result of discovery. One or several transformations are possible in this step. Transformations are executed in the order in which they are defined. All preprocessing is done by Zabbix server.

See also:

- Preprocessing details
- Preprocessing testing

<table>
<thead>
<tr>
<th>Type</th>
<th>Transformation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Regular expression</td>
<td>Match the received value to the &lt;pattern&gt; regular expression and replace value with the extracted &lt;output&gt;. The regular expression supports extraction of maximum 10 captured groups with the \N sequence. Parameters: &lt;br&gt;<strong>pattern</strong> - regular expression &lt;br&gt;<strong>output</strong> - output formatting template. An \N (where N=1…9) escape sequence is replaced with the Nth matched group. A \0 escape sequence is replaced with the matched text. &lt;br&gt;If you mark the Custom on fail checkbox, it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| Replace      | Find the search string and replace it with another (or nothing). All occurrences of the search string will be replaced. Parameters:  
- **search string**: the string to find and replace, case-sensitive (required)  
- **replacement**: the string to replace the search string with. The replacement string may also be empty effectively allowing to delete the search string when found.  
It is possible to use escape sequences to search for or replace line breaks, carriage return, tabs and spaces "\n \r \t \s"; backslash can be escaped as "\" and escape sequences can be escaped as "\n". Escaping of line breaks, carriage return, tabs is automatically done during low-level discovery. Supported since 5.0.0. |
| Structured data |                                                                                                       |
| JSONPath     | Extract value or fragment from JSON data using JSONPath functionality. If you mark the Custom on fail checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message. Supported since 4.4.0. |
| XML XPath    | Extract value or fragment from XML data using XPath functionality. For this option to work, Zabbix server must be compiled with libxml support. Examples:  
- `number(/document/item/value)` will extract 10 from `<document><item><value>10</value></item></document>`  
- `number(/document/item/@attribute)` will extract 10 from `<document><item attribute="10"></item></document>`  
- `/document/item` will extract `<item><value>10</value></item>` from `<document><item><value>10</value></item></document>`  
Note that namespaces are not supported. |
| CSV to JSON   | Convert CSV file data into JSON format. For more information, see: CSV to JSON preprocessing. Supported since 4.4.0. |
| XML to JSON   | Convert data in XML format to JSON. For more information, see: Serialization rules. If you mark the Custom on fail checkbox, it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message. |
| Custom scripts |                                                                                                       |
| JavaScript    | Enter JavaScript code in the block that appears when clicking in the parameter field or on Open. Note that available JavaScript length depends on the database used. For more information, see: Javascript preprocessing |
| Validation    |                                                                                                       |
| Does not match regular expression | Specify a regular expression that a value must not match. E.g. `Error: (.*)\.`. If you mark the Custom on fail checkbox, it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message. |
| Check for error in JSON | Check for an application-level error message located at JSONPath. Stop processing if succeeded and message is not empty; otherwise continue processing with the value that was before this preprocessing step. Note that these external service errors are reported to user as is, without adding preprocessing step information. E.g. `$.errors`. If a JSON like `{"errors": "e1"}` is received, the next preprocessing step will not be executed. If you mark the Custom on fail checkbox, it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message. |
Type

**Check for error in XML**
Check for an application-level error message located at xpath. Stop processing if succeeded and message is not empty; otherwise continue processing with the value that was before this preprocessing step. Note that these external service errors are reported to user as is, without adding preprocessing step information.
No error will be reported in case of failing to parse invalid XML.
Supported since 4.4.0.
If you mark the Custom on fail checkbox, it is possible to specify custom error handling options: either to discard the value, set a specified value or set a specified error message.

**Throttling**

**Discard unchanged with heartbeat**
Discard a value if it has not changed within the defined time period (in seconds).
Positive integer values are supported to specify the seconds (minimum - 1 second). Time suffixes can be used in this field (e.g. 30s, 1m, 2h, 1d). User macros and low-level discovery macros can be used in this field.
Only one throttling option can be specified for a discovery item.
E.g. 1m. If identical text is passed into this rule twice within 60 seconds, it will be discarded.
Note: Changing item prototypes does not reset throttling. Throttling is reset only when preprocessing steps are changed.

**Prometheus**

**Prometheus to JSON**
Convert required Prometheus metrics to JSON.
See Prometheus checks for more details.

Note that if the discovery rule has been applied to the host via template then the content of this tab is read-only.

Custom macros

The **LLD macros** tab allows to specify custom low-level discovery macros.

Custom macros are useful in cases when the returned JSON does not have the required macros already defined. So, for example:

- The native `vfs.fs.discovery` key for filesystem discovery returns a JSON with some pre-defined LLD macros such as `{#FSNAME}`, `{#FSTYPE}`. These macros can be used in item, trigger prototypes (see subsequent sections of the page) directly; defining custom macros is not needed;
- The `vfs.fs.get` agent item also returns a JSON with `filesystem data`, but without any pre-defined LLD macros. In this case you may define the macros yourself, and map them to the values in the JSON using JSONPath:

![LLD macros tab](image)

The extracted values can be used in discovered items, triggers, etc. Note that values will be extracted from the result of discovery and any preprocessing steps so far.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLD macro</td>
<td>Name of the low-level discovery macro, using the following syntax: <code>{#MACRO}</code>. Path that is used to extract LLD macro value from a LLD row, using JSONPath syntax. For example, <code>.foo</code> will extract &quot;bar&quot; and &quot;baz&quot; from this JSON: <code>{&quot;foo&quot;:&quot;bar&quot;},{&quot;foo&quot;:&quot;baz&quot;}</code>. The values extracted from the returned JSON are used to replace the LLD macros in item, trigger, etc. prototype fields. JSONPath can be specified using the dot notation or the bracket notation. Bracket notation should be used in case of any special characters and Unicode, like <code>$[&quot;unicode + special chars #1&quot;]</code>[<code>unicode + special chars #2</code>].</td>
</tr>
</tbody>
</table>
Filter

A filter can be used to generate real items, triggers, and graphs only for entities that match the criteria. The Filters tab contains discovery rule filter definitions allowing to filter discovery values:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of calculation</td>
<td>The following options for calculating filters are available:</td>
</tr>
<tr>
<td></td>
<td><strong>And</strong> - all filters must be passed;</td>
</tr>
<tr>
<td></td>
<td><strong>Or</strong> - enough if one filter is passed;</td>
</tr>
<tr>
<td></td>
<td><strong>And/Or</strong> - uses And with different macro names and Or with the same macro name;</td>
</tr>
<tr>
<td></td>
<td><strong>Custom expression</strong> - offers the possibility to define a custom calculation of filters. The formula must include all filters in the list. Limited to 255 symbols.</td>
</tr>
<tr>
<td>Filters</td>
<td>The following filter condition operators are available: matches, does not match, exists, does not exist.</td>
</tr>
</tbody>
</table>

Matches and does not match operators expect a Perl Compatible Regular Expression (PCRE). For instance, if you are only interested in C:, D:, and E: file systems, you could put {#FSNAME} into "Macro" and "^C|^D|^E" regular expression into "Regular expression" text fields. Filtering is also possible by file system types using {#FSTYPE} macro (e.g. "^ext|^reiserfs") and by drive types (supported only by Windows agent) using {#FSDRIVETYPE} macro (e.g., "fixed").

You can enter a regular expression or reference a global regular expression in "Regular expression" field.

In order to test a regular expression you can use "grep -E", for example:
```
for f in ext2 nfs reiserfs smbfs; do echo $f | grep -E "^ext|\^reiserfs" || echo "SKIP: $f"; done
```

(#FSDRIVETYPE) macro on Windows is supported since Zabbix 3.0.0.

Exists and does not exist operators allow to filter entities based on the presence or absence of the specified LLD macro in the response (supported since version 5.4.0).

Defining several filters is supported since Zabbix 2.4.0.

Note that if a macro from the filter is missing in the response, the found entity will be ignored, unless a "does not exist" condition is specified for this macro.

A mistake or a typo in the regular expression used in the LLD rule (for example, an incorrect "File systems for discovery" regular expression) may cause deletion of thousands of configuration elements, historical values, and events for many hosts.

Zabbix database in MySQL must be created as case-sensitive if file system names that differ only by case are to be discovered correctly.

Override

The Override tab allows setting rules to modify the list of item, trigger, graph and host prototypes or their attributes for discovered objects that meet given criteria.
Overrides (if any) are displayed in a reorderable drag-and-drop list and executed in the order in which they are defined. To configure details of a new override, click on in the Overrides block. To edit an existing override, click on the override name. A popup window will open allowing to edit the override rule details.

All mandatory parameters are marked with red asterisks.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>A unique (per LLD rule) override name.</td>
</tr>
</tbody>
</table>
| If filter matches | Defines whether next overrides should be processed when filter conditions are met:  
  Continue overrides - subsequent overrides will be processed.  
  Stop processing - operations from preceding (if any) and this override will be executed, subsequent overrides will be ignored for matched LLD rows. |
| Filters       | Determines to which discovered entities the override should be applied. Override filters are processed after discovery rule filters and have the same functionality. |
| Operations    | Override operations are displayed with these details:  
  Condition - an object type (item prototype/trigger prototype/graph prototype/host prototype) and a condition to be met (equals/does not equal/contains/does not contain/matches/does not match)  
  Action - links for editing and removing an operation are displayed. |

**Configuring an operation**

To configure details of a new operation, click on in the Operations block. To edit an existing operation, click on next to the operation. A popup window where you can edit the operation details will open.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Four types of objects are available: Item prototype, Trigger prototype, Graph prototype, Host prototype</td>
</tr>
<tr>
<td>Condition</td>
<td>Allows filtering entities to which the operation should be applied. Supported operators: equals - apply to this prototype, does not equal - apply to all prototypes, except this, contains - apply, if prototype name contains this string, does not contain - apply, if prototype name does not contain this string, matches - apply, if prototype name matches regular expression, does not match - apply, if prototype name does not match regular expression</td>
</tr>
<tr>
<td>Operator</td>
<td>A regular expression or a string to search for.</td>
</tr>
<tr>
<td>Create enabled</td>
<td>When the checkbox is marked, the buttons will appear, allowing to override original item prototype settings: Yes - the item will be added in an enabled state, No - the item will be added to a discovered entity but in a disabled state.</td>
</tr>
<tr>
<td>Discover</td>
<td>When the checkbox is marked, the buttons will appear, allowing to override original item prototype settings: Yes - the item will be added, No - the item will not be added.</td>
</tr>
<tr>
<td>Update interval</td>
<td>When the checkbox is marked, two options will appear, allowing to set different interval for the item: Delay - item update interval. User macros and time suffixes (e.g. 30s, 1m, 2h, 1d) are supported. Should be set to 0 if Custom interval is used. Custom interval - click to specify flexible/scheduling intervals. For detailed information see Custom intervals.</td>
</tr>
<tr>
<td>History storage period</td>
<td>When the checkbox is marked, the buttons will appear, allowing to set different history storage period for the item: Do not keep history - if selected, the history will not be stored. Storage period - if selected, an input field for specifying storage period will appear to the right. User macros and LLD macros are supported.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Trend storage period      | When the checkbox is marked, the buttons will appear, allowing to set different trend storage period for the item:  
Do not keep trends - if selected, the trends will not be stored.  
Storage period - if selected, an input field for specifying storage period will appear to the right. User macros and LLD macros are supported. |
| Tags                      | When the checkbox is marked, a new block will appear, allowing to specify tag-value pairs.  
These tags will be appended to the tags specified in the item prototype, even if the tag names match. |
| Object: Trigger prototype | Create enabled When the checkbox is marked, the buttons will appear, allowing to override original trigger prototype settings:  
Yes - the trigger will be added in an enabled state.  
No - the trigger will be added to a discovered entity, but in a disabled state.  
Discover When the checkbox is marked, the buttons will appear, allowing to override original trigger prototype settings:  
Yes - the trigger will be added.  
No - the trigger will not be added.  
Severity When the checkbox is marked, trigger severity buttons will appear, allowing to modify trigger severity.  
Tags When the checkbox is marked, a new block will appear, allowing to specify tag-value pairs.  
These tags will be appended to the tags specified in the trigger prototype, even if the tag names match. |
| Object: Graph prototype   | Discover When the checkbox is marked, the buttons will appear, allowing to override original graph prototype settings:  
Yes - the graph will be added.  
No - the graph will not be added. |
| Object: Host prototype    | Create enabled When the checkbox is marked, the buttons will appear, allowing to override original host prototype settings:  
Yes - the host will be created in an enabled state.  
No - the host will be created in a disabled state.  
Discover When the checkbox is marked, the buttons will appear, allowing to override original host prototype settings:  
Yes - the host will be discovered.  
No - the host will not be discovered.  
Link templates When the checkbox is marked, an input field for specifying templates will appear. Start typing the template name or click on Select next to the field and select templates from the list in a popup window.  
All templates linked to a host prototype will be replaced by templates from this override.  
Tags When the checkbox is marked, a new block will appear, allowing to specify tag-value pairs.  
These tags will be appended to the tags specified in the host prototype, even if the tag names match. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host inventory</td>
<td>When the checkbox is marked, the buttons will appear, allowing to select different inventory mode for the host prototype:</td>
</tr>
<tr>
<td></td>
<td>Disabled - do not populate host inventory</td>
</tr>
<tr>
<td></td>
<td>Manual - provide details manually</td>
</tr>
<tr>
<td></td>
<td>Automated - auto-fill host inventory data based on collected metrics.</td>
</tr>
</tbody>
</table>

**Form buttons**

Buttons at the bottom of the form allow to perform several operations.

- **Add**
  - Add a discovery rule. This button is only available for new discovery rules.

- **Update**
  - Update the properties of a discovery rule. This button is only available for existing discovery rules.

- **Clone**
  - Create another discovery rule based on the properties of the current discovery rule.

- **Check now**
  - Perform discovery based on the discovery rule immediately. The discovery rule must already exist. See more details.
  - Note that when performing discovery immediately, configuration cache is not updated, thus the result will not reflect very recent changes to discovery rule configuration.

- **Delete**
  - Delete the discovery rule.

- **Cancel**
  - Cancel the editing of discovery rule properties.

**Discovered entities** The screenshots below illustrate how discovered items, triggers, and graphs look like in the host’s configuration. Discovered entities are prefixed with an orange link to a discovery rule they come from.

Note that discovered entities will not be created in case there are already existing entities with the same uniqueness criteria, for example, an item with the same key or graph with the same name. An error message is displayed in this case in the frontend that the low-level discovery rule could not create certain entities. The discovery rule itself, however, will not turn unsupported because some entity could not be created and had to be skipped. The discovery rule will go on creating/updating other entities.

Items (similarly, triggers and graphs) created by a low-level discovery rule will be deleted automatically if a discovered entity (file system, interface, etc) stops being discovered (or does not pass the filter anymore). In this case the items, triggers and graphs will be deleted after the days defined in the Keep lost resources period field pass.

When discovered entities become ‘Not discovered anymore’, a lifetime indicator is displayed in the item list. Move your mouse pointer over it and a message will be displayed indicating how many days are left until the item is deleted.
If entities were marked for deletion, but were not deleted at the expected time (disabled discovery rule or item host), they will be deleted the next time the discovery rule is processed.

Entities containing other entities, which are marked for deletion, will not update if changed on the discovery rule level. For example, LLD-based triggers will not update if they contain items that are marked for deletion.

Other types of discovery

More detail and how-tos on other types of out-of-the-box discovery is available in the following sections:

- discovery of network interfaces;
- discovery of CPUs and CPU cores;
- discovery of SNMP OIDs;
- discovery of JMX objects;
- discovery using ODBC SQL queries;
- discovery of Windows services;
- discovery of host interfaces in Zabbix.

For more detail on the JSON format for discovery items and an example of how to implement your own file system discoverer as a Perl script, see creating custom LLD rules.

Creating custom LLD rules

It is also possible to create a completely custom LLD rule, discovering any type of entities - for example, databases on a database server.

To do so, a custom item should be created that returns JSON, specifying found objects and optionally - some properties of them. The amount of macros per entity is not limited - while the built-in discovery rules return either one or two macros (for example, two for filesystem discovery), it is possible to return more.

The required JSON format is best illustrated with an example. Suppose we are running an old Zabbix 1.8 agent (one that does not support "vfs.fs.discovery"), but we still need to discover file systems. Here is a simple Perl script for Linux that discovers mounted

```
Mounted filesystem discovery: Disk space usage /
```
filesystems and outputs JSON, which includes both filesystem name and type. One way to use it would be as a UserParameter with key "vfs.fs.discovery_perl":

```perl
#!/usr/bin/perl

$first = 1;

print "[";

for (`cat /proc/mounts`) {
    ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;

    print "\t,\n" if not $first;
    $first = 0;

    print "\t\n";
    print "\t\t"."fsname":"$fsname",\n"fstype":"$fstype";
    print "\t\n"
}

print "]
";
```

Allowed symbols for LLD macro names are 0-9, A-Z, _, .

Lowercase letters are not supported in the names.

An example of its output (reformatted for clarity) is shown below. JSON for custom discovery checks has to follow the same format.

```json
[
    { "fsname": "/", "fstype": "rootfs" },
    { "fsname": "/sys", "fstype": "sysfs" },
    { "fsname": "/proc", "fstype": "proc" },
    { "fsname": "/dev", "fstype": "devtmpfs" },
    { "fsname": "/dev/pts", "fstype": "devpts" },
    { "fsname": "/lib/init/rw", "fstype": "tmpfs" },
    { "fsname": "/dev/shm", "fstype": "tmpfs" },
    { "fsname": "/home", "fstype": "ext3" },
    { "fsname": "/tmp", "fstype": "ext3" },
    { "fsname": "/usr", "fstype": "ext3" },
    { "fsname": "/var", "fstype": "ext3" },
    { "fsname": "/sys/fs/fuse/connections", "fstype": "fusectl" }
]
```

In previous example it is required that the keys match the LLD macro names used in prototypes, the alternative is to extract LLD macro values using JSONPath `{#FSNAME}` -> $fsname and `{#FSTYPE}` -> $fstype, thus making such script possible:

```perl
#!/usr/bin/perl

$first = 1;

print "[";

for (`cat /proc/mounts`) {
    ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;

    print "\t,\n" if not $first;
    $first = 0;

    print "\t\n";
    print "\t\t"."fsname":"$fsname",\n"fstype":"$fstype";
    print "\t\n"
}

print "]
";
```
An example of its output (reformatted for clarity) is shown below. JSON for custom discovery checks has to follow the same format.

```json
[
    {
        "fsname": "/",
        "ftype": "rootfs"
    },
    {
        "fsname": "/sys",
        "ftype": "sysfs"
    },
    {
        "fsname": "/proc",
        "ftype": "proc"
    },
    {
        "fsname": "/dev",
        "ftype": "devtmpfs"
    },
    {
        "fsname": "/dev/pts",
        "ftype": "devpts"
    },
    {
        "fsname": "/lib/init/rw",
        "ftype": "tmpfs"
    },
    {
        "fsname": "/dev/shm",
        "ftype": "tmpfs"
    },
    {
        "fsname": "/home",
        "ftype": "ext3"
    },
    {
        "fsname": "/tmp",
        "ftype": "ext3"
    },
    {
        "fsname": "/usr",
        "ftype": "ext3"
    },
    {
        "fsname": "/var",
        "ftype": "ext3"
    },
    {
        "fsname": "/sys/fs/fuse/connections",
        "ftype": "fusectl"
    }
]
```

Then, in the discovery rule’s “Filter” field, we could specify “{#FSTYPE}” as a macro and “rootfs|ext3” as a regular expression.

You don’t have to use macro names `FSNAME/FSTYPE` with custom LLD rules, you are free to use whatever names you like. In case JSONPath is used then LLD row will be an array element that can be an object, but it can be also another array or a value.

Note that, if using a user parameter, the return value is limited to 512 KB. For more details, see data limits for LLD return values.

---

1 Item prototypes

Once a rule is created, go to the items for that rule and press “Create item prototype” to create an item prototype. Note how macro `{#FSNAME}` is used where a file system name is required. When the discovery rule is processed, this macro will be substituted with the discovered file system.
Low-level discovery macros and user macros may be used in item prototype configuration and item value preprocessing parameters. Note that when used in update intervals, a single macro has to fill the whole field. Multiple macros in one field or macros mixed with text are not supported.

Context-specific escaping of low-level discovery macros is performed for safe use in regular expression and XPath preprocessing parameters.

Attributes that are specific for item prototypes:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create enabled</td>
<td>If checked the item will be added in an enabled state.</td>
</tr>
<tr>
<td></td>
<td>If unchecked, the item will be added to a discovered entity, but in a disabled state.</td>
</tr>
<tr>
<td>Discover</td>
<td>If checked (default) the item will be added to a discovered entity.</td>
</tr>
<tr>
<td></td>
<td>If unchecked, the item will not be added to a discovered entity, unless this setting is overridden in the discovery rule.</td>
</tr>
</tbody>
</table>

We can create several item prototypes for each file system metric we are interested in:
Click on the three-dot icon to open the menu for the specific item prototype with these options:<br> - Create trigger prototype<br> - create a trigger prototype based on this item prototype<br> - Trigger prototypes - click to see a list with links to already-configured trigger prototypes of this item prototype<br> - Create dependent item - create a dependent item for this item prototype

**Mass update** option is available if you want to update properties of several item prototypes at once.

### 2 Trigger prototypes

We create trigger prototypes in a similar way as item prototypes:
Attributes that are specific for trigger prototypes:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create enabled</td>
<td>If checked the trigger will be added in an enabled state. If unchecked, the trigger will be added to a discovered entity, but in a disabled state.</td>
</tr>
</tbody>
</table>
Parameter Description

Discover
If checked (default) the trigger will be added to a discovered entity.
If unchecked, the trigger will not be added to a discovered entity, unless this setting is overridden in the discovery rule.

When real triggers are created from the prototypes, there may be a need to be flexible as to what constant (‘20’ in our example) is used for comparison in the expression. See how user macros with context can be useful to accomplish such flexibility.

You can define dependencies between trigger prototypes as well (supported since Zabbix 3.0). To do that, go to the Dependencies tab. A trigger prototype may depend on another trigger prototype from the same low-level discovery (LLD) rule or on a regular trigger. A trigger prototype may not depend on a trigger prototype from a different LLD rule or on a trigger created from trigger prototype. Host trigger prototype cannot depend on a trigger from a template.

3 Graph prototypes

We can create graph prototypes, too:
Attributes that are specific for graph prototypes:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discover</td>
<td>If checked (default) the graph will be added to a discovered entity. If unchecked, the graph will not be added to a discovered entity, unless this setting is overridden in the discovery rule.</td>
</tr>
</tbody>
</table>

Finally, we have created a discovery rule that looks as shown below. It has five item prototypes, two trigger prototypes, and one graph prototype.
4 Host prototypes

Host prototypes can be created with the low-level discovery rule. When matching entities are discovered, these prototypes become real hosts. Discovered hosts belong to an existing host and are prefixed with the discovery rule name.

Prototypes, before becoming discovered, cannot have their own items and triggers, other than those from the linked templates.

**Host prototype configuration**  To create a host prototype, press on the Host prototypes hyperlink for the required discovery rule, then press Create host prototype button in the upper right corner.

In the new window, specify host prototype parameters. Host prototypes have the same parameters as regular hosts, with the following exceptions:

- Host name must contain at least one low-level discovery macro to ensure that hosts created from the prototype have unique host names.
- Interfaces defines whether discovered hosts should inherit the IP of a host the discovery rule belongs to (default) or get custom interfaces.
- Group prototypes allows specifying host group prototypes by using LLD macros.
- Create enabled sets the status of discovered hosts, if the checkbox is unmarked the hosts will be created, but disabled.
- Discover - if the checkbox is unmarked, the hosts will not be created from the host prototype, unless this setting is overridden in the discovery rule.
- Value maps are not supported for host prototypes.

LLD macros can be used for host name, visible name, host group prototype, interfaces, tag values, or values of host prototype user macros.

**Host interfaces**

To add custom interfaces, switch the Interface selector from Inherit to Custom mode, then press **Add** and select the required interface type from the menu.
A host prototype may have any of the supported interface types: Zabbix agent, SNMP, JMX, IPMI.

Low-level discovery macros and user macros are supported.

If several custom interfaces are specified, use the Default column to specify the primary interface.

Notes:

• If Custom is selected, but no interfaces have been specified, the hosts will be created without interfaces.
• If Inherit is selected for a host prototype that belongs to a template, discovered hosts will inherit the interface of a host to which the template is linked to.

A host will not be created, if a host interface contains incorrect data.

**Discovered hosts** In the host list, discovered hosts are prefixed with the name of the discovery rule that created them.

The following discovered host parameters are customizable:

• Templates - it is possible to link additional templates to these hosts or unlink manually added templates. Templates inherited from a host prototype cannot be unlinked.
• Description
• Status - a host can be manually enabled/disabled.
• Tags - host tags can be added manually, alongside the tags inherited from the host prototype. Neither manual nor inherited tags can be duplicate, i.e. have the same name and value. If an inherited tag has the same name and value as a manual tag, it will replace the manual tag during discovery.
• Macros - host macros can be added manually, alongside the macros inherited from the host prototype. For inherited macros, it is possible to change macro value and type on the host level.
• Host inventory fields

Other parameters are inherited from the host prototype as read-only.

Discovered hosts can be deleted manually. Hosts that are no longer discovered, will be deleted automatically, based on the Keep lost resources period (in days) value of the discovery rule.

5 Notes on low-level discovery

Using LLD macros in user macro contexts

LLD macros may be used inside user macro context, for example, in trigger prototypes.

Multiple LLD rules for the same item

Since Zabbix agent version 3.2 it is possible to define several low-level discovery rules with the same discovery item.

To do that you need to define the Alias agent parameter, allowing to use altered discovery item keys in different discovery rules, for example vfs.fs.discovery[foo], vfs.fs.discovery[bar], etc.

Data limits for return values

There is no limit for low-level discovery rule JSON data if it is received directly by Zabbix server, because return values are processed without being stored in a database. There’s also no limit for custom low-level discovery rules, however, if it is intended to acquire custom LLD data using a user parameter, then the user parameter return value limit applies (512 KB).

If data has to go through Zabbix proxy it has to store this data in database so database limits apply.

6 Discovery rules

Please use the sidebar to see discovery rule configuration examples for various cases.

1 Discovery of mounted filesystems

Overview

It is possible to discover mounted filesystems and their properties (mountpoint name, mountpoint type, filesystem size and inode statistics).

To do that, you may use a combination of:

• the vfs.fs.get agent item as the master item
• dependent low-level discovery rule and item prototypes
Configuration

Master item

Create a Zabbix agent item using the following key:

\texttt{vfs.fs.get}

Set the type of information to "Text" for possibly big JSON data.

The data returned by this item will contain something like the following for a mounted filesystem:

```json
{
    "fsname": "/",
    "fstype": "rootfs",
    "bytes": {
        "total": 1000,
        "free": 500,
        "used": 500,
        "pfree": 50.00,
        "pused": 50.00
    },
    "inodes": {
        "total": 1000,
        "free": 500,
        "used": 500,
        "pfree": 50.00,
        "pused": 50.00
    }
}
```

Dependent LLD rule

Create a low-level discovery rule as "Dependent item" type:
As master item select the `vfs.fs.get` item we created.

In the "LLD macros" tab define custom macros with the corresponding JSONPath:

```
{#FSNAME}
{#FSTYPE}
```

Dependent item prototype

Create an item prototype with "Dependent item" type in this LLD rule. As master item for this prototype select the `vfs.fs.get` item we created.

```
Name: Free disk space on {#FSNAME}, type: {#FSTYPE}
```

Note the use of custom macros in the item prototype name and key:

- Name: Free disk space on {#FSNAME}, type: {#FSTYPE}
• Key: Free[#{FSNAME}]

As type of information, use:

• Numeric (unsigned) for metrics like ‘free’, ‘total’, ‘used’
• Numeric (float) for metrics like ‘pfree’, ‘pused’ (percentage)

In the item prototype "Preprocessing" tab select JSONPath and use the following JSONPath expression as parameter:

$.?[@.fsname=='{#FSNAME}')].bytes.free.first()

When discovery starts, one item per each mountpoint will be created. This item will return the number of free bytes for the given mountpoint.

2 Discovery of network interfaces

In a similar way as file systems are discovered, it is possible to also discover network interfaces.

Item key

The item key to use in the discovery rule is

net.if.discovery

This item is supported since Zabbix agent 2.0.

Supported macros

You may use the {#IFNAME} macro in the discovery rule filter and prototypes of items, triggers and graphs.

Examples of item prototypes that you might wish to create based on "net.if.discovery":

• “net.if.in[#{IFNAME},bytes]”.
• “net.if.out[#{IFNAME},bytes]”.

Note that on Windows {#IFGUID} is also returned.

3 Discovery of CPUs and CPU cores

In a similar way as file systems are discovered, it is possible to also discover CPUs and CPU cores.

Item key

The item key to use in the discovery rule is

system.cpu.discovery

This item is supported since Zabbix agent 2.4.

Supported macros

This discovery key returns two macros - {#CPU.NUMBER} and {#CPU.STATUS} identifying the CPU order number and status respectively. Note that a clear distinction cannot be made between actual, physical processors, cores and hyperthreads. {#CPU.STATUS} on Linux, UNIX and BSD systems returns the status of the processor, which can be either “online” or “offline”. On Windows systems, this same macro may represent a third value - “unknown” - which indicates that a processor has been detected, but no information has been collected for it yet.

CPU discovery relies on the agent’s collector process to remain consistent with the data provided by the collector and save resources on obtaining the data. This has the effect of this item key not working with the test (-t) command line flag of the agent binary, which will return a NOT_SUPPORTED status and an accompanying message indicating that the collector process has not been started.

Item prototypes that can be created based on CPU discovery include, for example:
4 Discovery of SNMP OIDs

Overview

In this section we will perform an SNMP discovery on a switch.

Item key

Unlike with file system and network interface discovery, the item does not necessarily have to have an "snmp.discovery" key - item type of SNMP agent is sufficient.

Discovery of SNMP OIDs is supported since Zabbix server/proxy 2.0.

To configure the discovery rule, do the following:

- Go to: Configuration → Templates
- Click on Discovery in the row of an appropriate template

- Click on Create discovery rule in the upper right corner of the screen
- Fill in the discovery rule form with the required details as in the screenshot below
All mandatory input fields are marked with a red asterisk.

The OIDs to discover are defined in SNMP OID field in the following format: discovery[#{MACRO1}, oid1, #{MACRO2}, oid2, ...],

where #{MACRO1}, #{MACRO2} ... are valid lld macro names and oid1, oid2... are OIDs capable of generating meaningful values for these macros. A built-in macro #{SNMPINDEX} containing index of the discovered OID is applied to discovered entities. The discovered entities are grouped by #{SNMPINDEX} macro value.

To understand what we mean, let us perform few snmpwalks on our switch:

```
$ snmpwalk -v 2c -c public 192.168.1.1 IF-MIB::ifDescr
IF-MIB::ifDescr.1 = STRING: WAN
IF-MIB::ifDescr.2 = STRING: LAN1
IF-MIB::ifDescr.3 = STRING: LAN2

$ snmpwalk -v 2c -c public 192.168.1.1 IF-MIB::ifPhysAddress
IF-MIB::ifPhysAddress.1 = STRING: 8:0:27:90:7a:75
IF-MIB::ifPhysAddress.2 = STRING: 8:0:27:90:7a:76
IF-MIB::ifPhysAddress.3 = STRING: 8:0:27:2b:af:9e
```

And set SNMP OID to: discovery[#{IFDESCR}, ifDescr, #{IFPHYSADDRESS}, ifPhysAddress]

Now this rule will discover entities with #{IFDESCR} macros set to **WAN, LAN1** and **LAN2**, #{IFPHYSADDRESS} macros set to **8:0:27:90:7a:75, 8:0:27:90:7a:76, and 8:0:27:2b:af:9e**, #{SNMPINDEX} macros set to the discovered OIDs indexes 1, 2 and 3:

```
[
  {
    "#{SNMPINDEX}": "1",
```
If an entity does not have the specified OID, then the corresponding macro will be omitted for this entity. For example if we have
the following data:

```plaintext
ifDescr.1 "Interface #1"
ifDescr.2 "Interface #2"
ifDescr.4 "Interface #4"
ifAlias.1 "eth0"
ifAlias.2 "eth1"
ifAlias.3 "eth2"
ifAlias.5 "eth4"
```

Then in this case SNMP discovery discovery[{#IFDESCR}, ifDescr, {#IFALIAS}, ifAlias] will return the following structure:

```plaintext
[
    {
        "{#SNMPINDEX}": 1,
        "{#IFDESCR}": "Interface #1",
        "{#IFALIAS}": "eth0"
    },
    {
        "{#SNMPINDEX}": 2,
        "{#IFDESCR}": "Interface #2",
        "{#IFALIAS}": "eth1"
    },
    {
        "{#SNMPINDEX}": 3,
        "{#IFALIAS}": "eth2"
    },
    {
        "{#SNMPINDEX}": 4,
        "{#IFDESCR}": "Interface #4"
    },
    {
        "{#SNMPINDEX}": 5,
        "{#IFALIAS}": "eth4"
    }
]
```

**Item prototypes**

The following screenshot illustrates how we can use these macros in item prototypes:
Again, creating as many item prototypes as needed:

### Item prototypes

<table>
<thead>
<tr>
<th>Item prototype</th>
<th>Tags</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td>Incoming traffic on interface <code>{#IFDESCR}</code></td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td>SNMP agent</td>
</tr>
<tr>
<td>Key</td>
<td></td>
<td><code>ifInOctets[#IFDESCR]</code></td>
</tr>
<tr>
<td>SNMP OID</td>
<td></td>
<td><code>IF-MIB::ifInOctets.#SNMPINDEX</code></td>
</tr>
<tr>
<td>SNMP community</td>
<td></td>
<td><code>{$SNMP_COMMUNITY}</code></td>
</tr>
<tr>
<td>Port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type of information</td>
<td>Numeric (unsigned)</td>
<td></td>
</tr>
<tr>
<td>Units</td>
<td></td>
<td>bps</td>
</tr>
<tr>
<td>Update interval</td>
<td></td>
<td>1m</td>
</tr>
</tbody>
</table>

**Trigger prototypes**

The following screenshot illustrates how we can use these macros in trigger prototypes:
<table>
<thead>
<tr>
<th><strong>Trigger prototype</strong></th>
<th><strong>Tags</strong></th>
<th><strong>Dependencies</strong></th>
</tr>
</thead>
</table>

| **Name**  | Interface `{IFDESCR}: Link down` |
| Event name | Interface `{IFDESCR}: Link down` |
| Operational data | Current state: `{ITEM.LASTVALUE1}` |
| **Severity** | Not classified | Information | Warning | Average | High | Disaster |

**Problem expression**

```
{$IFCONTROL:"{#IFNAME}"}={1} and ({Template Module Interfaces Simple SNMPv1:net.if.status[ifOperStatus.#{SNMPINDEX}].last()}{2} and {Template Module Interfaces Simple SNMPv1:net.if.status[ifOperStatus.#{SNMPINDEX}].diff()}{1})
```

**Expression constructor**

**OK event generation**

- Expression
- Recovery expression
- None

**Recovery expression**

```
{Template Module Interfaces Simple SNMPv1:net.if.status[ifOperStatus.#{SNMPINDEX}].last()}{2}
```

**Expression constructor**

**PROBLEM event generation mode**

- Single
- Multiple

**OK event closes**

- All problems
- All problems if tag values match

**Allow manual close**

- ✔

**URL**

- |

**Description**

This trigger expression works as follows:

1. Can be triggered if operations status is down.
2. \{$IFCONTROL:"{#IFNAME}"\}={1} - user can redefine Context macro to value 0. That marks this interface as not important. No new trigger will be fired if this interface is down.
3. `{TEMPLATE_NAME:METRIC.diff()}{1}` - trigger fires only if operational status was up(1) sometime before. (So, do not fire 'ethernal off' interfaces.)

**WARNING:** if closed manually - won't fire again on next poll, because of .diff.

**Create enabled**

- ✔

**Discover**

- ✔
Graph prototypes

The following screenshot illustrates how we can use these macros in graph prototypes:
A summary of our discovery rule:

**Discovered entities**

When server runs, it will create real items, triggers and graphs based on the values the SNMP discovery rule returns. In the host configuration they are prefixed with an orange link to a discovery rule they come from.
5 Discovery of JMX objects

Overview

It is possible to discover all JMX MBeans or MBean attributes or to specify a pattern for the discovery of these objects.

It is mandatory to understand the difference between an MBean and MBean attributes for discovery rule configuration. An MBean is an object which can represent a device, an application, or any resource that needs to be managed.

For example, there is an MBean which represents a web server. Its attributes are connection count, thread count, request timeout, http file cache, memory usage, etc. Expressing this thought in human comprehensive language we can define a coffee machine as an MBean which has the following attributes to be monitored: water amount per cup, average consumption of water for a certain period of time, number of coffee beans required per cup, coffee beans and water refill time, etc.

Item key

In discovery rule configuration, select JMX agent in the Type field.

Two item keys are supported for JMX object discovery - jmx.discovery[] and jmx.get[]:

<table>
<thead>
<tr>
<th>Item key</th>
<th>Return value</th>
<th>Parameters</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>jmx.discovery[]</td>
<td></td>
<td>&lt;discovery mode&gt;, &lt;object name&gt;, &lt;unique short description&gt;</td>
<td></td>
</tr>
</tbody>
</table>
This item returns a JSON array with LLD macros describing MBean objects or their attributes. Discovery mode - one of the following: attributes (retrieve JMX MBean attributes, default) or beans (retrieve JMX MBeans object name - object name pattern (see documenta- tion) identifying the MBean names to be retrieved (empty limitations to what default, retrieving all registered beans) can unique short description - a unique description that allows multiple JMX items with the same discovery mode can be
discovery Examples:

- \( jmx.discovery \)
- \( jmx.discovery[beans] \)
- \( jmx.discovery[attributes,"*:type=GarbageCollector,name=*"] \)
- \( jmx.discovery[beans,"*:type=GarbageCollector,name=*"] \)

There are some limitations to what MBean properties this item can return based on limited characters that are supported in macro name generation (supported characters can be described by the following regular expression: A-Z0-9_\). So, for example, to discover MBean properties with a hyphenated word or non-ASCII characters, you need to use \( jmx.get[\] \). Supported since Zabbix Java gateway 3.4.
Item key

jmx.get{<discovery mode>,<object name>,<unique short description>}

642
This item returns a JSON array with MBean objects or their attributes. Compared to jmx.discovery[] it does not define LLD macros.

**discovery**

- **mode** - one of the following:
  - attributes
  - beans

**object name** - object name pattern (see documentation) identifying the MBean names to be retrieved (empty by default, retrieving all registered beans)

**unique short description** - a unique description that allows multiple JMX items with the same discovery mode

Supported since Zabbix Java gateway 4.4.
If no parameters are passed, all MBean attributes from JMX are requested. Not specifying parameters for JMX discovery or trying to receive all attributes for a wide range like `*:type=*,name=*` may lead to potential performance problems.

Using jmx.discovery

This item returns a JSON object with low-level discovery macros describing MBean objects or attributes. For example, in the discovery of MBean attributes (reformatted for clarity):

```json
[
  {
    "{#JMXVALUE}" : "0",
    "{#JMXTYPE}" : "java.lang.Long",
    "{#JMXOBJ}" : "java.lang:type=GarbageCollector,name=PS Scavenge",
    "{#JMXDESC}" : "java.lang:type=GarbageCollector,name=PS Scavenge,CollectionCount",
    "{#JMXATTR}" : "CollectionCount"
  },
  {
    "{#JMXVALUE}" : "0",
    "{#JMXTYPE}" : "java.lang.Long",
    "{#JMXOBJ}" : "java.lang:type=GarbageCollector,name=PS Scavenge",
    "{#JMXDESC}" : "java.lang:type=GarbageCollector,name=PS Scavenge,CollectionTime",
    "{#JMXATTR}" : "CollectionTime"
  },
  {
    "{#JMXVALUE}" : "true",
    "{#JMXTYPE}" : "java.lang.Boolean",
    "{#JMXOBJ}" : "java.lang:type=GarbageCollector,name=PS Scavenge",
    "{#JMXDESC}" : "java.lang:type=GarbageCollector,name=PS Scavenge,Valid",
    "{#JMXATTR}" : "Valid"
  },
  {
    "{#JMXVALUE}" : "PS Scavenge",
    "{#JMXTYPE}" : "java.lang.String",
    "{#JMXOBJ}" : "java.lang:type=GarbageCollector,name=PS Scavenge",
    "{#JMXDESC}" : "java.lang:type=GarbageCollector,name=PS Scavenge,Name",
    "{#JMXATTR}" : "Name"
  },
  {
    "{#JMXVALUE}" : "java.lang:type=GarbageCollector,name=PS Scavenge",
    "{#JMXTYPE}" : "javax.management.ObjectName",
    "{#JMXOBJ}" : "java.lang:type=GarbageCollector,name=PS Scavenge",
    "{#JMXDESC}" : "java.lang:type=GarbageCollector,name=PS Scavenge,ObjectName",
    "{#JMXATTR}" : "ObjectName"
  }
]
```

In the discovery of MBeans (reformatted for clarity):

```json
[
  {
    "{#JMXDOMAIN}" : "java.lang",
    "{#JMXTYPE}" : "GarbageCollector",
    "{#JMXOBJ}" : "java.lang:type=GarbageCollector,name=PS Scavenge",
    "{#JMXNAME}" : "PS Scavenge"
  }
]
```

Supported macros

The following macros are supported for use in the discovery rule `filter` and prototypes of items, triggers and graphs:
Macro Description

Discovery of MBean attributes

{#JMXVALUE} Attribute value.
{#JMXTYPE} Attribute type.
{#JMXOBJ} Object name.
{#JMXDESC} Object name including attribute name.
{#JMXATTR} Attribute name.

Discovery of MBeans

{#JMXDOMAIN} MBean domain. (Zabbix reserved name)
{#JMXOBJ} Object name. (Zabbix reserved name)
{#JMX<key property>} MBean properties (like {#JMXTYPE}, {#JMXNAME}) (see Limitations below).

Limitations

There are some limitations associated with the algorithm of creating LLD macro names from MBean property names:

- attribute names are changed to uppercase
- attribute names are ignored (no LLD macros are generated) if they consist of unsupported characters for LLD macro names. Supported characters can be described by the following regular expression: A-Z0-9_.
- if an attribute is called "obj" or "domain" they will be ignored because of the overlap with the values of the reserved Zabbix properties ({#JMXOBJ} and {#JMXDOMAIN}) (supported since Zabbix 3.4.3.)

Please consider this jmx.discovery (with "beans" mode) example. MBean has the following properties defined:

```java
name=test
    =Type
attributes []=1,2,3
Name=NameOfTheTest
domAin=some
```

As a result of JMX discovery, the following LLD macros will be generated:

- {#JMXDOMAIN} - Zabbix internal, describing the domain of MBean
- {#JMXOBJ} - Zabbix internal, describing MBean object
- {#JMXNAME} - created from “name” property

Ignored properties are:

- тип : its name contains unsupported characters (non-ASCII)
- attributes[] : its name contains unsupported characters (square brackets are not supported)
- Name : it's already defined (name=test)
- domAin : it's a Zabbix reserved name

Examples

Let's review two more practical examples of a LLD rule creation with the use of Mbean. To understand the difference between a LLD rule collecting Mbeans and a LLD rule collecting Mbean attributes better please take a look at following table:

<table>
<thead>
<tr>
<th>MBean1</th>
<th>MBean2</th>
<th>MBean3</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBean1Attribute1</td>
<td>MBean2Attribute1</td>
<td>MBean3Attribute1</td>
</tr>
<tr>
<td>MBean1Attribute2</td>
<td>MBean2Attribute2</td>
<td>MBean3Attribute2</td>
</tr>
<tr>
<td>MBean1Attribute3</td>
<td>MBean2Attribute3</td>
<td>MBean3Attribute3</td>
</tr>
</tbody>
</table>

Example 1: Discovering MBeans

This rule will return 3 objects: the top row of the column: MBean1, MBean2, MBean3.

For more information about objects please refer to supported macros table, Discovery of MBeans section.

Discovery rule configuration collecting Mbeans (without the attributes) looks like the following:
The key used here:

```java
jmx.discovery[beans, "*:type=GarbageCollector, name="*"]
```

All the garbage collectors without attributes will be discovered. As Garbage collectors have the same attribute set, we can use desired attributes in item prototypes the following way:

```java
jmx[#{JMXOBJ}, CollectionCount]
jmx[#{JMXOBJ}, CollectionTime]
jmx[#{JMXOBJ}, Valid]
```

LLD discovery rule will result in something close to this (items are discovered for two Garbage collectors):

Example 2: Discovering Mbean attributes

This rule will return 9 objects with the following fields: MBean1Attribute1, MBean2Attribute1, MBean3Attribute1, MBean1Attribute2, MBean2Attribute2, MBean3Attribute2, MBean1Attribute3, MBean2Attribute3, MBean3Attribute3.

For more information about objects please refer to supported macros table, Discovery of MBean attributes section.

Discovery rule configuration collecting MBean attributes looks like the following:

![Image of Discovery rule configuration](image_url)
The key used here:

```
jmx.discovery[attributes,"*:type=GarbageCollector,name=*"]
```

All the garbage collectors with a single item attribute will be discovered.

In this particular case an item will be created from prototype for every MBean attribute. The main drawback of this configuration is that trigger creation from trigger prototypes is impossible as there is only one item prototype for all attributes. So this setup can be used for data collection, but is not recommended for automatic monitoring.

Using `jmx.get`

```
jmx.get[]
```

is similar to the `jmx.discovery[]` item, but is does not turn java object properties into low-level discovery macro names and therefore can return values without limitations that are associated with LLD macro name generation such as hyphens or non-ASCII characters.

When using `jmx.get[]` for discovery, low-level discovery macros can be defined separately in the custom LLD macro tab of the discovery rule configuration, using JSONPath to point to the required values.

Discovered MBeans

Discovery item: `jmx.get[beans,"com.example:type=*,*"]`

Response:

```
[
    {
        "object": "com.example:type=Hello,data-src=data-base, ",
        "domain": "com.example",
        "properties": {
            "data-src": "data-base",
            "": "",
            "type": "Hello"
        }
    },
    {
        "object": "com.example:type=Atomic",
        "domain": "com.example",
        "properties": {
            "type": "Atomic"
        }
    }
]
```
Discovering MBean attributes

Discovery item: `jmx.get[attributes,"com.example:type=*,*"]`

Response:

```json
[
  {
    "object": "com.example:type=*",
    "domain": "com.example",
    "properties": {
      "type": "Simple"
    }
  },
  {
    "object": "com.zabbix:type=yes,domain=zabbix.com,data-source=/dev/rand, = ,obj=true",
    "domain": "com.zabbix",
    "properties": {
      "type": "Hello",
      "domain": "com.example",
      "data-source": "/dev/rand",
      "": "",
      "obj": true
    }
  }
]
```

6 Discovery of IPMI sensors

Overview

It is possible to automatically discover IPMI sensors.

To do that, you may use a combination of:

- the `ipmi.get` IPMI item (supported since Zabbix 5.0.0) as the master item
- dependent low-level discovery rule and item prototypes

Configuration

Master item

Create an IPMI item using the following key:

`ipmi.get`
Set the type of information to "Text" for possibly big JSON data.

Dependent LLD rule

Create a low-level discovery rule as "Dependent item" type:

As master item select the `ipmi.get` item we created.

In the "LLD macros" tab define a custom macro with the corresponding JSONPath:

Dependent item prototype

Create an item prototype with "Dependent item" type in this LLD rule. As master item for this prototype select the `ipmi.get` item we created.
Note the use of the \{#SENSOR_ID\} macro in the item prototype name and key:

- Name: IPMI value for sensor \{#SENSOR_ID\}
- Key: ipmi_sensor[\{#SENSOR_ID\}]

As type of information, Numeric (unsigned).

In the item prototype "Preprocessing" tab select JSONPath and use the following JSONPath expression as parameter:

$.?[(@.id=='{#SENSOR_ID}')]\.value.first()

When discovery starts, one item per each IPMI sensor will be created. This item will return the integer value of the given sensor.

7 Discovery of systemd services

Overview

It is possible to discover systemd units (services, by default) with Zabbix.

Item key

The item to use in the discovery rule is the

systemd.unit.discovery

This item key is only supported in Zabbix agent 2.

This item returns a JSON with information about systemd units, for example:

```json
[
  {
    "{#UNIT.NAME}": "mysqld.service",
    "{#UNIT.DESCRIPTION}": "MySQL Server",
    "{#UNIT.LOADSTATE}": "loaded",
    "{#UNIT.ACTIVESTATE}": "active",
    "{#UNIT.SUBSTATE}": "running",
    "{#UNIT.FOLLOWED}": "",
    "{#UNIT.PATH}": "/org/freedesktop/systemd1/unit/mysqld_2eservice",
    "{#UNIT.JOBID}": 0,
    "{#UNIT.JOBTYPE}": "",
```
"{#UNIT.JOBPATH}": "/",
"{#UNIT.UNITFILESTATE}": "enabled"
],

[ {
"{#UNIT.NAME}": "systemd-journald.socket",
"{#UNIT.DESCRIPTION}": "Journal Socket",
"{#UNIT.LOADSTATE}": "loaded",
"{#UNIT.ACTIVESTATE}": "active",
"{#UNIT.SUBSTATE}": "running",
"{#UNIT.FOLLOWED}": ",
"{#UNIT.PATH}": "/org/freedesktop/systemd1/unit/systemd_2djournald_2esocket",
"{#UNIT.JOBID}": 0,
"{#UNIT.JOBTYPE}": ",
"{#UNIT.JOBPATH}": "/",
"{#UNIT.UNITFILESTATE}": "/",
"{#UNIT.UNITFILESTATE}": "enabled"
}]

Discovery of disabled systemd units

Since Zabbix 6.0.1 it is also possible to discover disabled systemd units. In this case three macros are returned in the resulting JSON:

• {#UNIT.PATH}
• {#UNIT.ACTIVESTATE}
• {#UNIT.UNITFILESTATE}.

To have items and triggers created from prototypes for disabled systemd units, make sure to adjust (or remove) prohibiting LLD filters for {#UNIT.ACTIVESTATE} and {#UNIT.UNITFILESTATE}.

Supported macros

The following macros are supported for use in the discovery rule filter and prototypes of items, triggers and graphs:

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{#UNIT.NAME}</td>
<td>Primary unit name.</td>
</tr>
<tr>
<td>{#UNIT.DESCRIPTION}</td>
<td>Human readable description.</td>
</tr>
<tr>
<td>{#UNIT.LOADSTATE}</td>
<td>Load state (i.e. whether the unit file has been loaded successfully)</td>
</tr>
<tr>
<td>{#UNIT.ACTIVESTATE}</td>
<td>Active state (i.e. whether the unit is currently started or not)</td>
</tr>
<tr>
<td>{#UNIT.SUBSTATE}</td>
<td>Sub state (a more fine-grained version of the active state that is specific to the unit type, which the active state is not)</td>
</tr>
<tr>
<td>{#UNIT.FOLLOWED}</td>
<td>Unit that is being followed in its state by this unit, if there is any; otherwise an empty string.</td>
</tr>
<tr>
<td>{#UNIT.PATH}</td>
<td>Unit object path.</td>
</tr>
<tr>
<td>{#UNIT.JOBID}</td>
<td>Numeric job ID if there is a job queued for the job unit; 0 otherwise.</td>
</tr>
<tr>
<td>{#UNIT.JOBTYPE}</td>
<td>Job type.</td>
</tr>
<tr>
<td>{#UNIT.JOBPATH}</td>
<td>Job object path.</td>
</tr>
<tr>
<td>{#UNIT.UNITFILESTATE}</td>
<td>The install state of the unit file.</td>
</tr>
</tbody>
</table>

Item prototypes

Item prototypes that can be created based on systemd service discovery include, for example:

• Item name: {#UNIT.DESCRIPTION}; item key: systemd.unit.info["{#UNIT.NAME}""]
• Item name: {#UNIT.DESCRIPTION}; item key: systemd.unit.info["{#UNIT.NAME}""],LoadState

systemd.unit.info agent items are supported since Zabbix 4.4.

8 Discovery of Windows services

Overview

In a similar way as file systems are discovered, it is possible to also discover Windows services.

Item key

The item to use in the discovery rule is service.discovery
This item is supported since Zabbix Windows agent 3.0.

Supported macros

The following macros are supported for use in the discovery rule filter and prototypes of items, triggers and graphs:

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{#SERVICE.NAME}</td>
<td>Service name.</td>
</tr>
<tr>
<td>{#SERVICE.DISPLAYNAME}</td>
<td>Displayed service name.</td>
</tr>
<tr>
<td>{#SERVICE.DESCRIPTION}</td>
<td>Service description.</td>
</tr>
</tbody>
</table>
| {#SERVICE.STATE}       | Numerical value of the service state:
                         | 0 - Running
                         | 1 - Paused
                         | 2 - Start pending
                         | 3 - Pause pending
                         | 4 - Continue pending
                         | 5 - Stop pending
                         | 6 - Stopped
                         | 7 - Unknown
| {#SERVICE.STATENAME}   | Name of the service state (Running, Paused, Start pending, Pause pending,
                         | Continue pending, Stop pending, Stopped or Unknown).                        |
| {#SERVICE.PATH}        | Service path.                                                               |
| {#SERVICE.USER}        | Service user.                                                               |
| {#SERVICE.STARTUP}     | Numerical value of the service startup type:
                         | 0 - Automatic
                         | 1 - Automatic delayed
                         | 2 - Manual
                         | 3 - Disabled
                         | 4 - Unknown
| {#SERVICE.STARTUPNAME} | Name of the service startup type (Automatic, Automatic delayed, Manual, Disabled,
                         | Unknown).                                                                   |
| {#SERVICE.STARTUPTRIGGER} | Numerical value to indicate if the service startup type has:
                          | 0 - no startup triggers
                          | 1 - has startup triggers
This macro is supported since Zabbix 3.4.4. It is useful to discover such service startup types as Automatic (trigger start), Automatic delayed (trigger start) and Manual (trigger start).

Based on Windows service discovery you may create an item prototype like

```
service.info[{{#SERVICE.NAME},{<param}>}]
```

where param accepts the following values: state, displayname, path, user, startup or description.

For example, to acquire the display name of a service you may use a “service.info[{{#SERVICE.NAME},displayname}” item. If param value is not specified (“service.info[{{#SERVICE.NAME}}]”), the default state parameter is used.

9 Discovery of Windows performance counter instances

Overview

It is possible to discover object instances of Windows performance counters. This is useful for multi-instance performance counters.

Item key

The item to use in the discovery rule is

```
perf_instance.discovery[object]
```

or, to be able to provide the object name in English only, independently of OS localization:

```
perf_instance_en.discovery[object]
```

For example:

```
perf_instance.discovery[Processador]
perf_instance_en.discovery[Processor]
```
These items are supported since Zabbix Windows agent 5.0.1.

Supported macros

The discovery will return all instances of the specified object in the `{#INSTANCE}` macro, which may be used in the prototypes of `perf_count` and `perf_count_en` items.

```
[
  {
    "{#INSTANCE}" : "0",
    "{#INSTANCE}" : "1",
    "{#INSTANCE}" : "_Total"
  }
]
```

For example, if the item key used in the discovery rule is:

```
perf_instance.discovery[Processor]
```

you may create an item prototype:

```
perf_counter["\Processor({#INSTANCE})\% Processor Time"]
```

Notes:

- If the specified object is not found or does not support variable instances then the discovery item will become NOTSUPPORTED.
- If the specified object supports variable instances, but currently does not have any instances, then an empty JSON array will be returned.
- In case of duplicate instances they will be skipped.

10 Discovery using WMI queries

Overview

WMI is a powerful interface in Windows that can be used for retrieving various information about Windows components, services, state and software installed.

It can be used for physical disk discovery and their performance data collection, network interface discovery, Hyper-V guest discovery, monitoring Windows services and many other things in Windows OS.

This type of low-level discovery is done using WQL queries whose results get automatically transformed into a JSON object suitable for low-level discovery.

Item key

The item to use in the discovery rule is

```
wmigetall[<namespace>,<query>]
```

This item transforms the query result into a JSON array. For example:

```
select * from Win32_DiskDrive where Name like '%PHYSICALDRIVE%'
```

may return something like this:

```
[
  {
    "DeviceID" : "\\.\PHYSICALDRIVE0",
    "BytesPerSector" : 512,
    "Capabilities" : [
      3,
      4
    ],
    "CapabilityDescriptions" : [
      "Random Access",
      "Supports Writing"
    ],
    "Caption" : "VBOX HARDDISK ATA Device",
    "ConfigManagerErrorCode" : "0",
    "ConfigManagerUserConfig" : "false",
    "CreationClassName" : "Win32_DiskDrive",
    "Description" : "Disk drive",
    "FirmwareRevision" : "1.0",
  }
]```
This item is supported since Zabbix Windows agent 4.4.

Low-level discovery macros

Even though no low-level discovery macros are created in the returned JSON, these macros can be defined by the user as an additional step, using the custom LLD macro functionality with JSONPath pointing to the discovered values in the returned JSON.

The macros then can be used to create item, trigger, etc prototypes.

11 Discovery using ODBC SQL queries

Overview

This type of low-level discovery is done using SQL queries, whose results get automatically transformed into a JSON object suitable for low-level discovery.

Item key

SQL queries are performed using a “Database monitor” item type. Therefore, most of the instructions on ODBC monitoring page apply in order to get a working “Database monitor” discovery rule.

Two item keys may be used in “Database monitor” discovery rules:

- **db.odbc.discovery** [unique short description>,<dsn>,<connection string>] - this item transforms the SQL query result into a JSON array, turning the column names from the query result into low-level discovery macro names paired with the discovered field values. These macros can be used in creating item, trigger, etc prototypes. See also: Using db.odbc.discovery.

- **db.odbc.get** [unique short description>,<dsn>,<connection string>] - this item transforms the SQL query result into a JSON array, keeping the original column names from the query result as a field name in JSON paired with the discovered values. Compared to db.odbc.discovery[], this item does not create low-level discovery macros in the returned JSON, therefore there is no need to check if the column names can be valid macro names. The low-level discovery macros can be defined as an additional step as required, using the custom LLD macro functionality with JSONPath pointing to the discovered values in the returned JSON. See also: Using db.odbc.get.

Using db.odbc.discovery

As a practical example to illustrate how the SQL query is transformed into JSON, let us consider low-level discovery of Zabbix proxies by performing an ODBC query on Zabbix database. This is useful for automatic creation of “zabbix[proxy,<name>,lastaccess]” internal items to monitor which proxies are alive.

Let us start with discovery rule configuration:
All mandatory input fields are marked with a red asterisk.

Here, the following direct query on Zabbix database is used to select all Zabbix proxies, together with the number of hosts they are monitoring. The number of hosts can be used, for instance, to filter out empty proxies:

```sql
mysql> SELECT h1.host, COUNT(h2.host) AS count FROM hosts h1 LEFT JOIN hosts h2 ON h1.hostid = h2.proxy_hostid WHERE h1.status IN (5, 6) GROUP BY h1.host;
```

```
+---------+-------+
| host    | count |
|---------+-------|
| Japan 1 | 5     |
| Japan 2 | 12    |
| Latvia  | 3     |
```

3 rows in set (0.01 sec)

By the internal workings of "db.odbc.discovery[{$DSN}]" item, the result of this query gets automatically transformed into the following JSON:

```json
[
    {
        "#HOST": "Japan 1",
        "#COUNT": "5"
    },
    {
        "#HOST": "Japan 2",
        "#COUNT": "12"
    },
    {
        "#HOST": "Latvia",
        "#COUNT": "3"
    }
]
```

It can be seen that column names become macro names and selected rows become the values of these macros.

If it is not obvious how a column name would be transformed into a macro name, it is suggested to use column aliases like "COUNT(h2.host) AS count" in the example above.
In case a column name cannot be converted into a valid macro name, the discovery rule becomes not supported, with the error message detailing the offending column number. If additional help is desired, the obtained column names are provided under DebugLevel=4 in Zabbix server log file:

```
$ grep db.odbc.discovery /tmp/zabbix_server.log
...
23876:20150114:153410.856 In db_odbc_discovery() query:'SELECT h1.host, COUNT(h2.host) FROM hosts h1 LEFT JOIN hosts h2 ON h1.hostid = h2.proxy_hostid WHERE h1.status IN (5, 6) GROUP BY h1.host;'
23876:20150114:153410.860 db_odbc_discovery() column[1]:'host'
23876:20150114:153410.860 db_odbc_discovery() column[2]:'COUNT(h2.host)'
23876:20150114:153410.860 End of db_odbc_discovery():NOTSUPPORTED
23876:20150114:153410.860 Item [Zabbix server:db.odbc.discovery[proxies,{$DSN}]] error: Cannot convert column #2 name to macro.
```

Now that we understand how a SQL query is transformed into a JSON object, we can use {#HOST} macro in item prototypes:

```
Once discovery is performed, an item will be created for each proxy:

```

Using db.odbc.get

Using db.odbc.get[,{$DSN}] and the following SQL example:

```
mysql> SELECT h1.host, COUNT(h2.host) AS count FROM hosts h1 LEFT JOIN hosts h2 ON h1.hostid = h2.proxy_hostid GROUP BY h1.host;
+---------+-------+
| host    | count |
|---------+-------|
| Japan 1 | 5     |
| Japan 2 | 12    |
| Latvia  | 3     |
+---------+-------+
3 rows in set (0.01 sec)
```

this JSON will be returned:

```
[
  {
    "host": "Japan 1",
    "count": "5"
  },
  {
```
As you can see, there are no low-level discovery macros there. However, custom low-level discovery macros can be created in the LLD macros tab of a discovery rule using JSONPath, for example:

```
{#HOST} → $.host
```

Now this `{#HOST}` macro may be used in item prototypes:

---

### 12 Discovery using Prometheus data

**Overview**

Data provided in Prometheus line format can be used for low-level discovery.

See [Prometheus checks](#) for details how Prometheus data querying is implemented in Zabbix.

**Configuration**

The low-level discovery rule should be created as a dependent item to the HTTP master item that collects Prometheus data.

**Prometheus to JSON**

In the discovery rule, go to the Preprocessing tab and select the Prometheus to JSON preprocessing option. Data in JSON format are needed for discovery and the Prometheus to JSON preprocessing option will return exactly that, with the following attributes:

- metric name
- metric value
- help (if present)
- type (if present)
- labels (if present)
- raw line

For example, querying `wmi_logical_disk_free_bytes`:

```json
"host": "Japan 2",
"count": "12"
},
{
"host": "Latvia",
"count": "3"
]
```

---
from these Prometheus lines:

```
# HELP wmi_logical_disk_free_bytes Free space in bytes (LogicalDisk.PercentFreeSpace)
# TYPE wmi_logical_disk_free_bytes gauge
wmi_logical_disk_free_bytes{volume="C:"} 3.5180249088e+11
wmi_logical_disk_free_bytes{volume="D:"} 2.627731456e+09
wmi_logical_disk_free_bytes{volume="HarddiskVolume4"} 4.59276288e+08
```

will return:

```
[
  {
    "name": "wmi_logical_disk_free_bytes",
    "help": "Free space in bytes (LogicalDisk.PercentFreeSpace)",
    "type": "gauge",
    "labels": {
      "volume": "C:"
    },
    "value": "3.5180249088e+11",
    "line_raw": "wmi_logical_disk_free_bytes{volume="C:"} 3.5180249088e+11"
  },
  {
    "name": "wmi_logical_disk_free_bytes",
    "help": "Free space in bytes (LogicalDisk.PercentFreeSpace)",
    "type": "gauge",
    "labels": {
      "volume": "D:"
    },
    "value": "2.627731456e+09",
    "line_raw": "wmi_logical_disk_free_bytes{volume="D:"} 2.627731456e+09"
  },
  {
    "name": "wmi_logical_disk_free_bytes",
    "help": "Free space in bytes (LogicalDisk.PercentFreeSpace)",
    "type": "gauge",
    "labels": {
      "volume": "HarddiskVolume4"
    },
    "value": "4.59276288e+08",
    "line_raw": "wmi_logical_disk_free_bytes{volume="HarddiskVolume4"} 4.59276288e+08"
  }
]
```

Mapping LLD macros

Next you have to go to the LLD macros tab and make the following mappings:

```
{#VOLUME}=${.labels['volume']}
{#METRIC}=${'name'}
{#HELP}=${'help'}
```

**Item prototype**

You may want to create an item prototype like this:
In a similar way as file systems are discovered, it is possible to also discover block devices and their type.

**Item key**

The item key to use in the discovery rule is 

`vfs.dev.discovery`

This item is supported on Linux platforms only, since Zabbix agent 4.4.

You may create discovery rules using this discovery item and:

### 13 Discovery of block devices

**with preprocessing options:**

<table>
<thead>
<tr>
<th>Preprocessing steps</th>
<th>Name</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prometheus pattern</td>
<td><code>{$METRIC}{volume={$#VOLUME}}</code></td>
</tr>
</tbody>
</table>
• filter: {#DEVNAME} matches sd[\d]$ - to discover devices named “sd0”, “sd1”, “sd2”, ...
• filter: {#DEVTYPE} matches disk AND {#DEVNAME} does not match ^loop.* - to discover disk type devices whose name does not start with “loop”

Supported macros
This discovery key returns two macros - {#DEVNAME} and {#DEVTYPE} identifying the block device name and type respectively, e.g.:

```json
[
  {
    "{#DEVNAME}":"loop1",
    "{#DEVTYPE}":"disk"
  },
  {
    "{#DEVNAME}":"dm-0",
    "{#DEVTYPE}":"disk"
  },
  {
    "{#DEVNAME}":"sda",
    "{#DEVTYPE}":"disk"
  },
  {
    "{#DEVNAME}":"sda1",
    "{#DEVTYPE}":"partition"
  }
]
```

Block device discovery allows to use `vfs.dev.read[]` and `vfs.dev.write[]` items to create item prototypes using the `{#DEVNAME}` macro, for example:

- `vfs.dev.read({#DEVNAME},sps]`
- `vfs.dev.write({#DEVNAME},sps]`

{#DEVTYPE} is intended for device filtering.

14 Discovery of host interfaces in Zabbix

Overview
It is possible to discover all interfaces configured in Zabbix frontend for a host.

Item key
The item to use in the discovery rule is the `zabbix[host,discovery,interfaces]` internal item. This item is supported since Zabbix server 3.4.

This item returns a JSON with the description of interfaces, including:

- IP address/DNS hostname (depending on the “Connect to” host setting)
- Port number
- Interface type (Zabbix agent, SNMP, JMX, IPMI)
- If it is the default interface or not
- If the bulk request feature is enabled - for SNMP interfaces only.

For example:

```json
[{"{#IF.CONN}":"192.168.3.1","{#IF.IP}":"192.168.3.1","{#IF.DNS}":""","{#IF.PORT}":"10050","{#IF.TYPE}":"AGENT"}]
```

With multiple interfaces their records in JSON are ordered by:

- Interface type,
- Default - the default interface is put before non-default interfaces,
- Interface ID (in ascending order).

Supported macros
The following macros are supported for use in the discovery rule filter and prototypes of items, triggers and graphs:
16. Distributed monitoring

Overview  Zabbix provides an effective and reliable way of monitoring a distributed IT infrastructure using Zabbix proxies. Proxies can be used to collect data locally on behalf of a centralized Zabbix server and then report the data to the server.

Proxy features

When making a choice of using/not using a proxy, several considerations must be taken into account.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Proxy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lightweight</td>
<td>Yes</td>
</tr>
<tr>
<td>GUI</td>
<td>No</td>
</tr>
<tr>
<td>Works independently</td>
<td>Yes</td>
</tr>
<tr>
<td>Easy maintenance</td>
<td>Yes</td>
</tr>
<tr>
<td>Automatic DB creation[^1]</td>
<td>Yes</td>
</tr>
<tr>
<td>Local administration</td>
<td>No</td>
</tr>
<tr>
<td>Ready for embedded hardware</td>
<td>Yes</td>
</tr>
<tr>
<td>One way TCP connections</td>
<td>Yes</td>
</tr>
<tr>
<td>Centralized configuration</td>
<td>Yes</td>
</tr>
<tr>
<td>Generates notifications</td>
<td>No</td>
</tr>
</tbody>
</table>

[^1] Automatic DB creation feature only works with SQLite. Other databases require a manual setup.

1 Proxies

Overview  A Zabbix proxy can collect performance and availability data on behalf of the Zabbix server. This way, a proxy can take on itself some of the load of collecting data and offload the Zabbix server.

Also, using a proxy is the easiest way of implementing centralized and distributed monitoring, when all agents and proxies report to one Zabbix server and all data is collected centrally.

A Zabbix proxy can be used to:

- Monitor remote locations
- Monitor locations having unreliable communications
- Offload the Zabbix server when monitoring thousands of devices
- Simplify the maintenance of distributed monitoring
The proxy requires only one TCP connection to the Zabbix server. This way it is easier to get around a firewall as you only need to configure one firewall rule.

Zabbix proxy must use a separate database. Pointing it to the Zabbix server database will break the configuration.

All data collected by the proxy is stored locally before transmitting it over to the server. This way no data is lost due to any temporary communication problems with the server. The ProxyLocalBuffer and ProxyOfflineBuffer parameters in the proxy configuration file control for how long the data are kept locally.

It may happen that a proxy, which receives the latest configuration changes directly from Zabbix server database, has a more up-to-date configuration than Zabbix server whose configuration may not be updated as fast due to the value of CacheUpdateFrequency. As a result, proxy may start gathering data and send them to Zabbix server that ignores these data.

Zabbix proxy is a data collector. It does not calculate triggers, process events or send alerts. For an overview of what proxy functionality is, review the following table:

<table>
<thead>
<tr>
<th>Function</th>
<th>Supported by proxy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td></td>
</tr>
<tr>
<td>Zabbix agent checks</td>
<td>Yes</td>
</tr>
<tr>
<td>Zabbix agent checks (active)</td>
<td>Yes 1</td>
</tr>
<tr>
<td>Simple checks</td>
<td>Yes</td>
</tr>
<tr>
<td>Trapper items</td>
<td>Yes</td>
</tr>
<tr>
<td>SNMP checks</td>
<td>Yes</td>
</tr>
<tr>
<td>SNMP traps</td>
<td>Yes</td>
</tr>
<tr>
<td>IPMI checks</td>
<td>Yes</td>
</tr>
<tr>
<td>JMX checks</td>
<td>Yes</td>
</tr>
<tr>
<td>Log file monitoring</td>
<td>Yes</td>
</tr>
<tr>
<td>Internal checks</td>
<td>Yes</td>
</tr>
<tr>
<td>SSH checks</td>
<td>Yes</td>
</tr>
<tr>
<td>Telnet checks</td>
<td>Yes</td>
</tr>
<tr>
<td>External checks</td>
<td>Yes</td>
</tr>
<tr>
<td>Dependent items</td>
<td>Yes</td>
</tr>
<tr>
<td>Script items</td>
<td>Yes</td>
</tr>
<tr>
<td>Built-in web monitoring</td>
<td>Yes</td>
</tr>
<tr>
<td>Item value preprocessing</td>
<td>Yes</td>
</tr>
<tr>
<td>Network discovery</td>
<td>Yes</td>
</tr>
<tr>
<td>Active agent autoregistration</td>
<td>Yes</td>
</tr>
<tr>
<td>Low-level discovery</td>
<td>Yes</td>
</tr>
<tr>
<td>Remote commands</td>
<td>Yes</td>
</tr>
<tr>
<td>Calculating triggers</td>
<td>No</td>
</tr>
<tr>
<td>Processing events</td>
<td>No</td>
</tr>
<tr>
<td>Event correlation</td>
<td>No</td>
</tr>
<tr>
<td>Sending alerts</td>
<td>No</td>
</tr>
</tbody>
</table>

[1] To make sure that an agent asks the proxy (and not the server) for active checks, the proxy must be listed in the ServerActive parameter in the agent configuration file.

Protection from overloading

If Zabbix server was down for some time, and proxies have collected a lot of data, and then the server starts, it may get overloaded (history cache usage stays at 95-100% for some time). This overload could result in a performance hit, where checks are processed.
slower than they should. Protection from this scenario was implemented to avoid problems that arise due to overloading history cache.

When Zabbix server history cache is full the history cache write access is being throttled, stalling server data gathering processes. The most common history cache overload case is after server downtime when proxies are uploading gathered data. To avoid this proxy throttling was added (currently it cannot be disabled).

Zabbix server will stop accepting data from proxies when history cache usage reaches 80%. Instead those proxies will be put on a throttling list. This will continue until the cache usage falls down to 60%. Now server will start accepting data from proxies one by one, defined by the throttling list. This means the first proxy that attempted to upload data during the throttling period will be served first and until it’s done the server will not accept data from other proxies.

This throttling mode will continue until either cache usage hits 80% again or falls down to 20% or the throttling list is empty. In the first case the server will stop accepting proxy data again. In the other two cases the server will start working normally, accepting data from all proxies.

You may use the zabbix[wcache,history,pused] internal item to correlate this behavior of Zabbix server with a metric.

**Configuration** Once you have installed and configured a proxy, it is time to configure it in the Zabbix frontend.

Adding proxies

To configure a proxy in Zabbix frontend:

- Go to: Administration → Proxies
- Click on Create proxy

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxy name</td>
<td>Enter the proxy name. It must be the same name as in the Hostname parameter in the proxy configuration file.</td>
</tr>
<tr>
<td>Proxy mode</td>
<td>Select the proxy mode. <strong>Active</strong> - the proxy will connect to the Zabbix server and request configuration data <strong>Passive</strong> - Zabbix server connects to the proxy Note that without encrypted communications (sensitive) proxy configuration data may become available to parties having access to the Zabbix server trapper port when using an active proxy. This is possible because anyone may pretend to be an active proxy and request configuration data if authentication does not take place or proxy addresses are not limited in the Proxy address field.</td>
</tr>
<tr>
<td>Proxy address</td>
<td>127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,example.example.com</td>
</tr>
</tbody>
</table>

You may use the zabbix[wcache,history,pused] internal item to correlate this behavior of Zabbix server with a metric.
### Parameter Description

**Proxy address**
- If specified then active proxy requests are only accepted from this list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of active Zabbix proxy.
- This field is only available if an active proxy is selected in the Proxy mode field. Macros are not supported.
- This option is supported since Zabbix 4.0.0.

**Interface**
- Enter interface details for the passive proxy.
- This field is only available if a passive proxy is selected in the Proxy mode field.

**IP address**
- IP address of the passive proxy (optional).

**DNS name**
- DNS name of the passive proxy (optional).

**Connect to**
- Clicking the respective button will tell Zabbix server what to use to retrieve data from proxy:
  - **IP** - Connect to the proxy IP address (recommended)
  - **DNS** - Connect to the proxy DNS name

**Port**
- TCP/UDP port number of the passive proxy (10051 by default).

**Description**
- Enter the proxy description.

The **Encryption** tab allows you to require encrypted connections with the proxy.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connections to proxy</td>
<td>How the server connects to the passive proxy: no encryption (default), using PSK (pre-shared key) or certificate.</td>
</tr>
<tr>
<td>Connections from proxy</td>
<td>Select what type of connections are allowed from the active proxy. Several connection types can be selected at the same time (useful for testing and switching to other connection type). Default is &quot;No encryption&quot;.</td>
</tr>
<tr>
<td>Issuer</td>
<td>Allowed issuer of certificate. Certificate is first validated with CA (certificate authority). If it is valid, signed by the CA, then the Issuer field can be used to further restrict allowed CA. This field is optional, intended to use if your Zabbix installation uses certificates from multiple CAs.</td>
</tr>
<tr>
<td>Subject</td>
<td>Allowed subject of certificate. Certificate is first validated with CA. If it is valid, signed by the CA, then the Subject field can be used to allow only one value of Subject string. If this field is empty then any valid certificate signed by the configured CA is accepted.</td>
</tr>
<tr>
<td>PSK identity</td>
<td>Pre-shared key identity string. Do not put sensitive information in the PSK identity, it is transmitted unencrypted over the network to inform a receiver which PSK to use.</td>
</tr>
<tr>
<td>PSK</td>
<td>Pre-shared key (hex-string). Maximum length: 512 hex-digits (256-byte PSK) if Zabbix uses GnuTLS or OpenSSL library, 64 hex-digits (32-byte PSK) if Zabbix uses mbed TLS (PolarSSL) library. Example: 1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952</td>
</tr>
</tbody>
</table>

The editing form of an existing proxy has the following additional buttons:
- Refresh configuration - refresh configuration of the proxy
- Clone - create a new proxy based on the properties of the existing proxy
- Delete - delete the proxy

**Host configuration**

You can specify that an individual host should be monitored by a proxy in the **host configuration** form, using the Monitored by proxy field.

- **Monitored by proxy**
  - (no proxy)
  - Remote proxy

**Host mass update** is another way of specifying that hosts should be monitored by a proxy.

**17. Encryption**
Overview  Zabbix supports encrypted communications between Zabbix components using Transport Layer Security (TLS) protocol v.1.2 and 1.3 (depending on the crypto library). Certificate-based and pre-shared key-based encryption is supported.

Encryption can be configured for connections:

- Between Zabbix server, Zabbix proxy, Zabbix agent, zabbix_sender and zabbix_get utilities
- To Zabbix database from Zabbix frontend and server/proxy

Encryption is optional and configurable for individual components:

- Some proxies and agents can be configured to use certificate-based encryption with the server, while others can use pre-shared key-based encryption, and yet others continue with unencrypted communications (as before)
- Server (proxy) can use different encryption configurations for different hosts

Zabbix daemon programs use one listening port for encrypted and unencrypted incoming connections. Adding an encryption does not require opening new ports on firewalls.

Limitations

- Private keys are stored in plain text in files readable by Zabbix components during startup
- Pre-shared keys are entered in Zabbix frontend and stored in Zabbix database in plain text
- Built-in encryption does not protect communications:
  - Between the web server running Zabbix frontend and user web browser
  - Between Zabbix frontend and Zabbix server
- Currently each encrypted connection opens with a full TLS handshake, no session caching and tickets are implemented
- Adding encryption increases the time for item checks and actions, depending on network latency:
  - For example, if packet delay is 100ms then opening a TCP connection and sending unencrypted request takes around 200ms. With encryption about 1000 ms are added for establishing the TLS connection;
  - Timeouts may need to be increased, otherwise some items and actions running remote scripts on agents may work with unencrypted connections, but fail with timeout with encrypted.
- Encryption is not supported by network discovery. Zabbix agent checks performed by network discovery will be unencrypted and if Zabbix agent is configured to reject unencrypted connections such checks will not succeed.

Compiling Zabbix with encryption support  To support encryption Zabbix must be compiled and linked with one of the supported crypto libraries:

- GnuTLS - from version 3.1.18
- OpenSSL - versions 1.0.1, 1.0.2, 1.1.0, 1.1.1, 3.0.x
- LibreSSL - tested with versions 2.7.4, 2.8.2:
  - LibreSSL 2.6.x is not supported
  - LibreSSL is supported as a compatible replacement of OpenSSL; the new tls_*() LibreSSL-specific API functions are not used. Zabbix components compiled with LibreSSL will not be able to use PSK, only certificates can be used.

The library is selected by specifying the respective option to “configure” script:

- --with-gnutls[=DIR]
- --with-openssl[=DIR] (also used for LibreSSL)

For example, to configure the sources for server and agent with OpenSSL you may use something like:

./configure --enable-server --enable-agent --with-mysql --enable-ipv6 --with-net-snmp --with-libcurl --with-openssl

Different Zabbix components may be compiled with different crypto libraries (e.g. a server with OpenSSL, an agent with GnuTLS).

If you plan to use pre-shared keys (PSK), consider using GnuTLS or OpenSSL 1.1.0 (or newer) libraries in Zabbix components using PSKs. GnuTLS and OpenSSL 1.1.0 libraries support PSK ciphersuites with Perfect Forward Secrecy. Older versions of the OpenSSL library (1.0.1, 1.0.2c) also support PSKs, but available PSK ciphersuites do not provide Perfect Forward Secrecy.

Connection encryption management  Connections in Zabbix can use:

- no encryption (default)
- RSA certificate-based encryption
- PSK-based encryption

There are two important parameters used to specify encryption between Zabbix components:

- TLSConnect - specifies what encryption to use for outgoing connections (unencrypted, PSK or certificate)
- TLSAccept - specifies what types of connections are allowed for incoming connections (unencrypted, PSK or certificate). One or more values can be specified.
TLSConnect is used in the configuration files for Zabbix proxy (in active mode, specifies only connections to server) and Zabbix agent (for active checks). In Zabbix frontend the TLSConnect equivalent is the Connections to host field in Configuration → Hosts → <some host> → Encryption tab and the Connections to proxy field in Administration → Proxies → <some proxy> → Encryption tab. If the configured encryption type for connection fails, no other encryption types will be tried.

TLSAccept is used in the configuration files for Zabbix proxy (in passive mode, specifies only connections from server) and Zabbix agent (for passive checks). In Zabbix frontend the TLSAccept equivalent is the Connections from host field in Configuration → Hosts → <some host> → Encryption tab and the Connections from proxy field in Administration → Proxies → <some proxy> → Encryption tab.

Normally you configure only one type of encryption for incoming encryptions. But you may want to switch the encryption type, e.g. from unencrypted to certificate-based with minimum downtime and rollback possibility. To achieve this:

- Set TLSAccept=unencrypted,cert in the agent configuration file and restart Zabbix agent
- Test connection with zabbix_get to the agent using certificate. If it works, you can reconfigure encryption for that agent in Zabbix frontend in the Configuration → Hosts → <some host> → Encryption tab by setting Connections to host to "Certificate".
- When server configuration cache gets updated (and proxy configuration is updated if the host is monitored by proxy) then connections to that agent will be encrypted
- If everything works as expected you can set TLSAccept=cert in the agent configuration file and restart Zabbix agent. Now the agent will be accepting only encrypted certificate-based connections. Unencrypted and PSK-based connections will be rejected.

In a similar way it works on server and proxy. If in Zabbix frontend in host configuration Connections from host is set to “Certificate” then only certificate-based encrypted connections will be accepted from the agent (active checks) and zabbix_sender (trapper items).

Most likely you will configure incoming and outgoing connections to use the same encryption type or no encryption at all. But technically it is possible to configure it asymmetrically, e.g. certificate-based encryption for incoming and PSK-based for outgoing connections.

Encryption configuration for each host is displayed in the Zabbix frontend, in Configuration → Hosts in the Agent encryption column. For example:

<table>
<thead>
<tr>
<th>Example</th>
<th>Connections to host</th>
<th>Allowed connections from host</th>
<th>Rejected connections from host</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>Unencrypted</td>
<td>Unencrypted</td>
<td>Encrypted, certificate-based</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Encrypted, certificate-based</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Unencrypted and PSK-based</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>encrypted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Unencrypted and certificate-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>based encrypted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Certificate-based encrypted</td>
</tr>
</tbody>
</table>

Connections are unencrypted by default. Encryption must be configured for each host and proxy individually.

zabbix_get and zabbix_sender with encryption See zabbix_get and zabbix_sender manpages for using them with encryption.

Ciphersuites Ciphersuites by default are configured internally during Zabbix startup and, before Zabbix 4.0.19, 4.4.7, are not user-configurable.

Since Zabbix 4.0.19, 4.4.7 also user-configured ciphersuites are supported for GnuTLS and OpenSSL. Users may configure ciphersuites according to their security policies. Using this feature is optional (built-in default ciphersuites still work).

For crypto libraries compiled with default settings Zabbix built-in rules typically result in the following ciphersuites (in order from higher to lower priority):

<table>
<thead>
<tr>
<th>Library</th>
<th>Certificate ciphersuites</th>
<th>PSK ciphersuites</th>
</tr>
</thead>
<tbody>
<tr>
<td>GnuTLS 3.1.18</td>
<td>TLS_ECDHE_RSA_AES_128_GCM_SHA256</td>
<td>TLS_ECDHE_PSK_AES_128_CBC_SHA256</td>
</tr>
<tr>
<td></td>
<td>TLS_ECDHE_RSA_AES_128_CBC_SHA256</td>
<td>TLS_ECDHE_PSK_AES_128_CBC_SHA1</td>
</tr>
<tr>
<td></td>
<td>TLS_ECDHE_RSA_AES_128_CBC_SHA1</td>
<td>TLS_PSK_AES_128_GCM_SHA256</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_AES_128_GCM_SHA256</td>
<td>TLS_PSK_AES_128_CBC_SHA256</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_AES_128_CBC_SHA256</td>
<td>TLS_PSK_AES_128_CBC_SHA1</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_AES_128_CBC_SHA1</td>
<td></td>
</tr>
<tr>
<td>Library</td>
<td>Certificate ciphersuites</td>
<td>PSK ciphersuites</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>OpenSSL 1.0.2c</td>
<td>ECDHE-RSA-AES128-GCM-SHA256</td>
<td>PSK-AES128-CBC-SHA</td>
</tr>
<tr>
<td></td>
<td>ECDHE-RSA-AES128-SHA256</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ECDHE-RSA-AES128-SHA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AES128-GCM-SHA256</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AES128-SHA256</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AES128-SHA</td>
<td></td>
</tr>
<tr>
<td>OpenSSL 1.1.0</td>
<td>ECDHE-RSA-AES128-GCM-SHA256</td>
<td>ECDHE-PSK-AES128-CBC-SHA256</td>
</tr>
<tr>
<td></td>
<td>ECDHE-RSA-AES128-SHA256</td>
<td>ECDHE-PSK-AES128-CBC-SHA</td>
</tr>
<tr>
<td></td>
<td>ECDHE-RSA-AES128-SHA</td>
<td>PSK-AES128-GCM-SHA256</td>
</tr>
<tr>
<td></td>
<td>AES128-GCM-SHA256</td>
<td>PSK-AES128-CCM8</td>
</tr>
<tr>
<td></td>
<td>AES128-CCM8</td>
<td>PSK-AES128-CBC-SHA256</td>
</tr>
<tr>
<td></td>
<td>AES128-CMC</td>
<td>PSK-AES128-CBC-SHA</td>
</tr>
<tr>
<td></td>
<td>AES128-SHA256</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AES128-SHA</td>
<td></td>
</tr>
<tr>
<td>OpenSSL 1.1.1d</td>
<td>TLS_AES_256_GCM_SHA384</td>
<td>TLS_CHACHA20_POLY1305_SHA256</td>
</tr>
<tr>
<td></td>
<td>TLS_CHACHA20_POLY1305_SHA256</td>
<td>TLS_AES_128_GCM_SHA256</td>
</tr>
<tr>
<td></td>
<td>TLS_AES_128_GCM_SHA256</td>
<td>ECDHE-PSK-AES128-CBC-SHA256</td>
</tr>
<tr>
<td></td>
<td>ECDHE-RSA-AES128-GCM-SHA256</td>
<td>ECDHE-PSK-AES128-CBC-SHA</td>
</tr>
<tr>
<td></td>
<td>ECDHE-RSA-AES128-SHA256</td>
<td>PSK-AES128-GCM-SHA256</td>
</tr>
<tr>
<td></td>
<td>ECDHE-RSA-AES128-SHA</td>
<td>PSK-AES128-CCM</td>
</tr>
<tr>
<td></td>
<td>AES128-GCM-SHA256</td>
<td>PSK-AES128-CBC-SHA256</td>
</tr>
<tr>
<td></td>
<td>AES128-CMC</td>
<td>PSK-AES128-CBC-SHA</td>
</tr>
<tr>
<td></td>
<td>AES128-SHA256</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AES128-SHA</td>
<td></td>
</tr>
</tbody>
</table>

**User-configured ciphersuites**  The built-in ciphersuite selection criteria can be overridden with user-configured ciphersuites.

User-configured ciphersuites is a feature intended for advanced users who understand TLS ciphersuites, their security and consequences of mistakes, and who are comfortable with TLS troubleshooting.

The built-in ciphersuite selection criteria can be overridden using the following parameters:

<table>
<thead>
<tr>
<th>Override scope</th>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ciphersuite selection for certificates</td>
<td>TLSCipherCert13</td>
<td>Valid OpenSSL 1.1.1 cipher strings for TLS 1.3 protocol (their values are passed to the OpenSSL function SSL_CTX_set_ciphersuites()).</td>
<td>Certificate-based ciphersuite selection criteria for TLS 1.3</td>
</tr>
<tr>
<td></td>
<td>TLSCipherCert</td>
<td>Valid OpenSSL cipher strings for TLS 1.2 or valid GnuTLS priority strings. Their values are passed to the SSL_CTX_set_cipher_list() or gnutls_priority_init() functions, respectively.</td>
<td>Only OpenSSL 1.1.1 or newer. Certificate-based ciphersuite selection criteria for TLS 1.2/1.3 (GnuTLS), TLS 1.2 (OpenSSL)</td>
</tr>
<tr>
<td>Ciphersuite selection for PSK</td>
<td>TLSCipherPSK13</td>
<td>Valid OpenSSL 1.1.1 cipher strings for TLS 1.3 protocol (their values are passed to the OpenSSL function SSL_CTX_set_ciphersuites()).</td>
<td>PSK-based ciphersuite selection criteria for TLS 1.3</td>
</tr>
<tr>
<td></td>
<td>TLSCipherPSK</td>
<td>Valid OpenSSL cipher strings for TLS 1.2 or valid GnuTLS priority strings. Their values are passed to the SSL_CTX_set_cipher_list() or gnutls_priority_init() functions, respectively.</td>
<td>Only OpenSSL 1.1.1 or newer. PSK-based ciphersuite selection criteria for TLS 1.2/1.3 (GnuTLS), TLS 1.2 (OpenSSL)</td>
</tr>
<tr>
<td>Combined ciphersuite list for certificate and PSK</td>
<td>TLSCipherAll13</td>
<td>Valid OpenSSL 1.1.1 cipher strings for TLS 1.3 protocol (their values are passed to the OpenSSL function SSL_CTX_set_ciphersuites()).</td>
<td>Ciphersuite selection criteria for TLS 1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Only OpenSSL 1.1.1 or newer.</td>
</tr>
<tr>
<td>Override scope</td>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>TLSCipherAll</td>
<td>Valid OpenSSL cipher strings for TLS 1.2 or valid GnuTLS priority strings. Their values are passed to the SSL_CTX_set_cipher_list() or gnutls_priority_init() functions, respectively.</td>
<td>Ciphersuite selection criteria for TLS 1.2/1.3 (GnuTLS), TLS 1.2 (OpenSSL)</td>
</tr>
</tbody>
</table>

To override the ciphersuite selection in `zabbix_get` and `zabbix_sender` utilities - use the command-line parameters:

- `--tls-cipher13`
- `--tls-cipher`

The new parameters are optional. If a parameter is not specified, the internal default value is used. If a parameter is defined it cannot be empty.

If the setting of a TLSCipher* value in the crypto library fails then the server, proxy or agent will not start and an error is logged.

It is important to understand when each parameter is applicable.

Outgoing connections

The simplest case is outgoing connections:

- For outgoing connections with certificate - use TLSCipherCert13 or TLSCipherCert
- For outgoing connections with PSK - use TLSCipherPSK13 and TLSCipherPSK
- In case of `zabbix_get` and `zabbix_sender` utilities the command-line parameters `--tls-cipher13` and `--tls-cipher` can be used (encryption is unambiguously specified with a `--tls-connect` parameter)

Incoming connections

It is a bit more complicated with incoming connections because rules are specific for components and configuration.

For Zabbix **agent**:

<table>
<thead>
<tr>
<th>Agent connection setup</th>
<th>Cipher configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLSCConnect=cert</td>
<td>TLSCipherCert, TLSCipherCert13</td>
</tr>
<tr>
<td>TLSCConnect=psk</td>
<td>TLSCipherPSK, TLSCipherPSK13</td>
</tr>
<tr>
<td>TLSAccept=cert</td>
<td>TLSCipherCert, TLSCipherCert13</td>
</tr>
<tr>
<td>TLSAccept=psk</td>
<td>TLSCipherPSK, TLSCipherPSK13</td>
</tr>
<tr>
<td>TLSAccept=cert,psk</td>
<td>TLSCipherAll, TLSCipherAll13</td>
</tr>
</tbody>
</table>

For Zabbix **server** and **proxy**:

<table>
<thead>
<tr>
<th>Connection setup</th>
<th>Cipher configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outgoing connections using PSK</td>
<td>TLSCipherPSK, TLSCipherPSK13</td>
</tr>
<tr>
<td>Incoming connections using certificates</td>
<td>TLSCipherAll, TLSCipherAll13</td>
</tr>
<tr>
<td>Incoming connections using PSK if server has no certificate</td>
<td>TLSCipherPSK, TLSCipherPSK13</td>
</tr>
<tr>
<td>Incoming connections using PSK if server has certificate</td>
<td>TLSCipherAll, TLSCipherAll13</td>
</tr>
</tbody>
</table>

Some pattern can be seen in the two tables above:

- TLSCipherAll and TLSCipherAll13 can be specified only if a combined list of certificate- and PSK-based ciphersuites is used. There are two cases when it takes place: server (proxy) with a configured certificate (PSK ciphersuites are always configured on server, proxy if crypto library supports PSK), agent configured to accept both certificate- and PSK-based incoming connections
- in other cases TLSCipherCert* and/or TLSCipherPSK* are sufficient

The following tables show the TLSCipher* built-in default values. They could be a good starting point for your own custom values.
### Parameter GnuTLS 3.6.12

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
</table>

### Parameter OpenSSL 1.1.1d

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLSCipherCert13</td>
<td>ECDH+aRSA+AES128:RSA+aRSA+AES128</td>
</tr>
<tr>
<td>TLSCipherPsk13</td>
<td>TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256</td>
</tr>
<tr>
<td>TLSCipherPsk</td>
<td>kECDHEPSK+AES128:kPSK+AES128</td>
</tr>
<tr>
<td>TLSCipherAll13</td>
<td>ECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+AES128</td>
</tr>
</tbody>
</table>

1 Default values are different for older OpenSSL versions (1.0.1, 1.0.2, 1.1.0), for LibreSSL and if OpenSSL is compiled without PSK support.

**Examples of user-configured ciphersuites**

See below the following examples of user-configured ciphersuites:

- Testing cipher strings and allowing only PFS ciphersuites
- Switching from AES128 to AES256

Testing cipher strings and allowing only PFS ciphersuites

To see which ciphersuites have been selected you need to set `DebugLevel=4` in the configuration file, or use the `-vv` option for `zabbix_sender`.

Some experimenting with TLSCipher* parameters might be necessary before you get the desired ciphersuites. It is inconvenient to restart Zabbix server, proxy or agent multiple times just to tweak TLSCipher* parameters. More convenient options are using `zabbix_sender` or the `openssl` command. Let's show both.

1. Using `zabbix_sender`.

Let's make a test configuration file, for example `/home/zabbix/test.conf`, with the syntax of a `zabbix_agentd.conf` file:

```plaintext`
Hostname=nonexisting
ServerActive=nonexisting
TLSConnect=cert
TLSCAFile=/home/zabbix/ca.crt
TLSCertFile=/home/zabbix/agent.crt
TLSKeyFile=/home/zabbix/agent.key
TLSPSKIdentity=nonexisting
TLSPSKFile=/home/zabbix/agent.psk
```

You need valid CA and agent certificates and PSK for this example. Adjust certificate and PSK file paths and names for your environment.

If you are not using certificates, but only PSK, you can make a simpler test file:

```plaintext`
Hostname=nonexisting
ServerActive=nonexisting
TLSConnect=psk
TLSPSKIdentity=nonexisting
TLSPSKFile=/home/zabbix/agentd.psk
```

The selected ciphersuites can be seen by running `zabbix_sender` (example compiled with OpenSSL 1.1.d):

```
$ zabbix_sender -vv -c /home/zabbix/test.conf -k nonexisting_item -o 1 2>&1 | grep ciphersuites
zabbix_sender [41271]: DEBUG: zbx_tls_init_child() certificate ciphersuites: TLS_AES_256_GCM_SHA384 TLS_AES_128_GCM_SHA256 TLS_AES_128_CBC_SHA256 TLS_AES_128_CBC_SHA TLS_AES_128_GCM SHA256 SHA1 CURVE-ALL COMP-NULL SIGN-ALL
```

669
Here you see the ciphersuites selected by default. These default values are chosen to ensure interoperability with Zabbix agents running on systems with older OpenSSL versions (from 1.0.1).

With newer systems you can choose to tighten security by allowing only a few ciphersuites, e.g. only ciphersuites with PFS (Perfect Forward Secrecy). Let's try to allow only ciphersuites with PFS using TLS\text{Cipher}* parameters.

The result will not be interoperable with systems using OpenSSL 1.0.1 and 1.0.2, if PSK is used. Certificate-based encryption should work.

Add two lines to the test.conf configuration file:

\begin{verbatim}
TLSCipherCert=EECDH+aRSA+AES128
TLSCipherPSK=kECDHEPSK+AES128
\end{verbatim}

and test again:

\begin{verbatim}
$ zabbix_sender -vv -c /home/zabbix/test.conf -k nonexisting_item -o 1 2>&1 | grep ciphersuites
\end{verbatim}

The "certificate ciphersuites" and "PSK ciphersuites" lists have changed - they are shorter than before, only containing TLS1.3 ciphersuites and TLS 1.2 ECDHE-* ciphersuites as expected.

2. \text{TLSCipherAll} and \text{TLSCipherAll13} cannot be tested with zabbix\_sender; they do not affect "certificate and PSK ciphersuites" value shown in the example above. To tweak TLS\text{CipherAll} and TLS\text{CipherAll13} you need to experiment with the agent, proxy or server.

So, to allow only PFS ciphersuites you may need to add up to three parameters

\begin{verbatim}
TLSCipherCert=EECDH+aRSA+AES128
TLSCipherPSK=kECDHEPSK+AES128
TLSCipherAll=EECDH+aRSA+AES128:kECDHEPSK+AES128
\end{verbatim}

to zabbix\_agentd.conf, zabbix\_proxy.conf and zabbix\_server\_conf if each of them has a configured certificate and agent has also PSK.

If your Zabbix environment uses only PSK-based encryption and no certificates, then only one:

\begin{verbatim}
TLSCipherPSK=kECDHEPSK+AES128
\end{verbatim}

Now that you understand how it works you can test the ciphersuite selection even outside of Zabbix, with the openssl command. Let's test all three TLS\text{Cipher}* parameter values:

\begin{verbatim}
$ openssl ciphers EECDH+aRSA+AES128 | sed 's/:/ /g'
\end{verbatim}

\begin{verbatim}
0x13,0x02 - TLS_AES_256_GCM_SHA384 TLSv1.3 Kx=any Au=any Enc=AESGCM(256) Mac=AEAD
0x13,0x03 - TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any Au=any Enc=CHACHA20/POLY1305(256) Mac=AEAD
0x13,0x01 - TLS_AES_128_GCM_SHA256 TLSv1.3 Kx=any Au=any Enc=AESGCM(128) Mac=AEAD
0xC0,0x2F - ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH Au=RSA Enc=AESGCM(256) Mac=AEAD
0xC0,0x27 - ECDHE-RSA-AES128-CBC-SHA256 TLSv1.2 Kx=ECDH Au=RSA Enc=AESGCM(128) Mac=SHA256
0xC0,0x13 - ECDHE-RSA-AES128-SHA TLSv1.2 Kx=ECDH Au=RSA Enc=AESGCM(128) Mac=SHA256
0xC0,0x37 - ECDHE-PSK-AES128-CBC-SHA256 TLSv1.2 Kx=ECDHEPSK Au=PSK Enc=AES(128) Mac=SHA256
0xC0,0x35 - ECDHE-PSK-AES128-CBC-SHA TLSv1.2 Kx=ECDHEPSK Au=PSK Enc=AES(128) Mac=SHA256
\end{verbatim}

You may prefer \text{openssl ciphers} with option \text{"-V"} for a more verbose output:

\begin{verbatim}
$ openssl ciphers -V EECDH+aRSA+AES128:kECDHEPSK+AES128
\end{verbatim}

Similarly, you can test the priority strings for GnuTLS:

\begin{verbatim}
\end{verbatim}

\begin{verbatim}
TLS_ECDHE_RSA_AES_128_GCM_SHA256 0xc0, 0x2f TLS1.2
TLS_ECDHE_RSA_AES_128_CBC_SHA256 0xc0, 0x27 TLS1.2
\end{verbatim}

Protocols: VERS-TLS1.2
Ciphers: AES-128-GCM, AES-128-CBC  
MACs: AEAD, SHA256  
Key Exchange Algorithms: ECDHE-RSA  
Groups: GROUP-SECP256R1, GROUP-SECP384R1, GROUP-SECP521R1, GROUP-X25519, GROUP-X448, GROUP-FFDHE2048, GROUP-FFDHE3072, GROUP-FFDHE4096, GROUP-FFDHE6144, GROUP-FFDHE8192  

Switching from AES128 to AES256

Zabbix uses AES128 as the built-in default for data. Let’s assume you are using certificates and want to switch to AES256, on OpenSSL 1.1.1.

This can be achieved by adding the respective parameters in zabbix_server.conf:

```
TLSCAFile=/home/zabbix/ca.crt
TLSCertFile=/home/zabbix/server.crt
TLSKeyFile=/home/zabbix/server.key
TLSCipherCert13=TLS_AES_256_GCM_SHA384
TLSCipherCert=EECDH+aRSA+AES256:-SHA1:-SHA384
TLSCipherPSK13=TLS_CHACHA20_POLY1305_SHA256
TLSCipherPSK=kECDHEPSK+AES256:-SHA1
TLSCipherAll13=TLS_AES_256_GCM_SHA384
TLSCipherAll=EECDH+aRSA+AES256:-SHA1:-SHA384
```

Although only certificate-related ciphersuites will be used, TLSCipherPSK* parameters are defined as well to avoid their default values which include less secure ciphers for wider interoperability. PSK ciphersuites cannot be completely disabled on server/proxy.

And in zabbix_agentd.conf:

```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/ca.crt
TLSCertFile=/home/zabbix/agent.crt
TLSKeyFile=/home/zabbix/agent.key
TLSCipherCert13=TLS_AES_256_GCM_SHA384
TLSCipherCert=EECDH+aRSA+AES256:-SHA1:-SHA384
```

1 Using certificates

Overview

Zabbix can use RSA certificates in PEM format, signed by a public or in-house certificate authority (CA). Certificate verification is done against a pre-configured CA certificate. Optionally certificate revocation lists (CRL) can be used. Each Zabbix component can have only one certificate configured.

For more information how to set up and operate internal CA, how to generate certificate requests and sign them, how to revoke certificates you can find numerous online how-tos, for example, [OpenSSLPKI Tutorial v1.1](#).

Carefully consider and test your certificate extensions - see [Limitations on using X.509 v3 certificate extensions](#).

Certificate configuration parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLSCAFile</td>
<td>yes</td>
<td>Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification. In case of certificate chain with several members they must be ordered: lower level CA certificates first followed by certificates of higher level CA(s). Certificates from multiple CA(s) can be included in a single file.</td>
</tr>
<tr>
<td>TLSCertFile</td>
<td>yes</td>
<td>Full pathname of a file containing certificate (certificate chain). In case of certificate chain with several members they must be ordered: server, proxy, or agent certificate first, followed by lower level CA certificates then certificates of higher level CA(s).</td>
</tr>
<tr>
<td>TLSKeyFile</td>
<td>yes</td>
<td>Full pathname of a file containing private key. Set access rights to this file - it must be readable only by Zabbix user.</td>
</tr>
<tr>
<td>TLS ServerCertIssuer</td>
<td>no</td>
<td>Allowed server certificate issuer.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>TLSServerCertSubject</td>
<td>no</td>
<td>Allowed server certificate subject.</td>
</tr>
</tbody>
</table>

Configuring certificate on Zabbix server

1. In order to verify peer certificates, Zabbix server must have access to file with their top-level self-signed root CA certificates. For example, if we expect certificates from two independent root CAs, we can put their certificates into file `/home/zabbix/zabbix_ca_file` like this:

**Certificate:**
```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
  Signature Algorithm: sha1WithRSAEncryption
  Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
  ... Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    Public-Key: (2048 bit)
    ... X509v3 extensions:
      X509v3 Key Usage: critical
        Certificate Sign, CRL Sign
      X509v3 Basic Constraints: critical
        CA:TRUE
      ...
-----BEGIN CERTIFICATE-----
MIID2jCCAsKgAwIBAgIBATANBgkqhkiG9w0BAQUFADB+MRMwEQYKZImiZPyLGQB
...9wEzdN8uTrqoyU78gi12npljOl8LegRKjb5hFTVmo
-----END CERTIFICATE-----
```

2. Put Zabbix server certificate chain into file, for example, `/home/zabbix/zabbix_server.crt`:

**Certificate:**
```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
  Signature Algorithm: sha1WithRSAEncryption
  Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root2 CA
  ... Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root2 CA
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    Public-Key: (2048 bit)
    ... X509v3 extensions:
      X509v3 Key Usage: critical
        Certificate Sign, CRL Sign
      X509v3 Basic Constraints: critical
        CA:TRUE
      ...
-----BEGIN CERTIFICATE-----
MIID3DCCAsKgAwIBAgIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKZImiZPyLGQB
...vdGNYosfvu4lQA5Vj5FmNJRJrzv5XQO23bB894Y1zY=
-----END CERTIFICATE-----
```

2. Put Zabbix server certificate chain into file, for example, `/home/zabbix/zabbix_server.crt`:

**Certificate:**
```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
  Signature Algorithm: sha1WithRSAEncryption
  Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
```
Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Zabbix server
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
  Public-Key: (2048 bit)

X509v3 extensions:
  X509v3 Key Usage: critical
    Digital Signature, Key Encipherment
  X509v3 Basic Constraints:
    CA:FALSE

-----BEGIN CERTIFICATE-----
MIIECDCCAvCgAwIBAgIBATANBgkqhkiG9w0BAQUFADCBgTETMBEGCmSJomT8ixk
...h02u1GHiy46GI+xfR3LsWFKlkTaaLaL/6aaoQ==
-----END CERTIFICATE-----

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 2 (0x2)
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
    ...Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)

X509v3 extensions:
  X509v3 Key Usage: critical
    Certificate Sign, CRL Sign
  X509v3 Basic Constraints: critical
    CA:TRUE, pathlen:0

-----BEGIN CERTIFICATE-----
MIID4TCCAsmgAwIBAgIBAjANBgkqhkiG9w0BAQUFADB+MRMwEQYK4zImiZPyLGQB
...dyCeWnvL7u5sd6f8o8iRnyQzbbHKmQt/wUtOVvWdMFJMOHw==
-----END CERTIFICATE-----

Here the first is Zabbix server certificate, followed by intermediate CA certificate.

3. Put Zabbix server private key into file, for example, /home/zabbix/zabbix_server.key:

-----BEGIN PRIVATE KEY-----
MIIeAwIBAgIBAkJANBkgqkhkiG9w0BAQQMEDCAEASCBKowggSmAgEAAoIBAQC9tIXIJoVnNXD1
...IJLkhbybBYEf47MLhffWa7xvZTY=
-----END PRIVATE KEY-----

4. Edit TLS parameters in Zabbix server configuration file like this:

TLSCAFile=/home/zabbix/zabbix_ca_file
TLSCertFile=/home/zabbix/zabbix_server.crt
TLSKeyFile=/home/zabbix/zabbix_server.key

Configuring certificate-based encryption for Zabbix proxy

1. Prepare files with top-level CA certificates, proxy certificate (chain) and private key as described in Configuring certificate on Zabbix server. Edit parameters TLSCAFile, TLSCertFile, TLSKeyFile in proxy configuration accordingly.

2. For active proxy edit TLSConnect parameter:

TLSConnect=cert

For passive proxy edit TLSAccept parameter:
3. Now you have a minimal certificate-based proxy configuration. You may prefer to improve proxy security by setting `TLSServerCertIssuer` and `TLSServerCertSubject` parameters (see *Restricting allowed certificate Issuer and Subject*).

4. In final proxy configuration file TLS parameters may look like:

   ```
   TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
   TLSServerCertSubject=CN=Zabbix server,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
   TLSCertFile=/home/zabbix/zabbix_proxy.crt
   TLSKeyFile=/home/zabbix/zabbix_proxy.key
   ```

5. Configure encryption for this proxy in Zabbix frontend:

   - Go to: Administration → Proxies
   - Select proxy and click on **Encryption** tab

   In examples below Issuer and Subject fields are filled in - see *Restricting allowed certificate Issuer and Subject* why and how to use these fields.

   For active proxy

   ![Configuration for Active Proxy](image)

   For passive proxy

   ![Configuration for Passive Proxy](image)

   Configuring certificate-based encryption for Zabbix agent
1. Prepare files with top-level CA certificates, agent certificate (chain) and private key as described in Configuring certificate on Zabbix server. Edit parameters TLSCAFile, TLSCertFile, TLSKeyFile in agent configuration accordingly.

2. For active checks edit TLSConnect parameter:
   
   TLSConnect=cert
   
   For passive checks edit TLSAccept parameter:
   
   TLSAccept=cert

3. Now you have a minimal certificate-based agent configuration. You may prefer to improve agent security by setting TLSServerCertIssuer and TLSServerCertSubject parameters. (see Restricting allowed certificate Issuer and Subject).

4. In final agent configuration file TLS parameters may look like:
   
   TLSConnect=cert
   TLSAccept=cert
   TLSCAFile=/home/zabbix/zabbix_ca_file
   TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
   TLSServerCertSubject=CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
   TLSCertFile=/home/zabbix/zabbix_agentd.crt
   TLSKeyFile=/home/zabbix/zabbix_agentd.key

   (Example assumes that host is monitored via proxy, hence proxy certificate Subject.)

5. Configure encryption for this agent in Zabbix frontend:
   
   • Go to: Configuration → Hosts 
   • Select host and click on Encryption tab 
   
   In example below Issuer and Subject fields are filled in - see Restricting allowed certificate Issuer and Subject why and how to use these fields.

<table>
<thead>
<tr>
<th>Host</th>
<th>Templates</th>
<th>IPMI</th>
<th>Macros</th>
<th>Host inventory</th>
<th>Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Connections to host</td>
<td>No encryption</td>
<td>PSK</td>
<td>Certificate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Connections from host</td>
<td>No encryption</td>
<td>PSK</td>
<td>Certificate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Issuer</td>
<td>CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Subject</td>
<td>CN=www01,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Restricting allowed certificate Issuer and Subject

When two Zabbix components (e.g. server and agent) establish a TLS connection they both check each others certificates. If a peer certificate is signed by a trusted CA (with pre-configured top-level certificate in TLSCAFile), is valid, has not expired and passes some other checks then communication can proceed. Certificate issuer and subject are not checked in this simplest case.

Here is a risk - anybody with a valid certificate can impersonate anybody else (e.g. a host certificate can be used to impersonate server). This may be acceptable in small environments where certificates are signed by a dedicated in-house CA and risk of impersonating is low.

If your top-level CA is used for issuing other certificates which should not be accepted by Zabbix or you want to reduce risk of impersonating you can restrict allowed certificates by specifying their Issuer and Subject strings.

For example, you can write in Zabbix proxy configuration file:

TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix server,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
With these settings, an active proxy will not talk to Zabbix server with different Issuer or Subject string in certificate, a passive proxy will not accept requests from such server.

A few notes about Issuer or Subject string matching:

1. Issuer and Subject strings are checked independently. Both are optional.
2. UTF-8 characters are allowed.
3. Unspecified string means any string is accepted.
4. Strings are compared “as-is”, they must be exactly the same to match.
5. Wildcards and regexp’s are not supported in matching.
6. Only some requirements from RFC 4514 Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names are implemented:
   2. escape characters space (’ ‘ U+0020) or number sign (‘#’ U+0023) at the beginning of string.
   3. escape character space (’ ‘ U+0020) at the end of string.
7. Match fails if a null character (U+0000) is encountered (RFC 4514 allows it).

Order of fields in Issuer and Subject strings and formatting are important! Zabbix follows RFC 4514 recommendation and uses “reverse” order of fields.

The reverse order can be illustrated by example:

TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com

Note that it starts with mid-level (OU, O) and ends with top-level (DC) fields.

OpenSSL by default shows certificate Issuer and Subject fields in “normal” order, depending on additional options used:

```bash
$ openssl x509 -noout -in /home/zabbix/zabbix_proxy.crt -issuer -subject
issuer= /DC=com/DC=zabbix/O=Zabbix SIA/OU=Development group/CN=Signing CA
subject= /DC=com/DC=zabbix/O=Zabbix SIA/OU=Development group/CN=Zabbix proxy
```

```bash
$ openssl x509 -noout -text -in /home/zabbix/zabbix_proxy.crt
Certificate:

...  
  Issuer: DC=com, DC=zabbix, 0=Zabbix SIA, OU=Development group, CN=Signing CA  
  Subject: DC=com, DC=zabbix, 0=Zabbix SIA, OU=Development group, CN=Zabbix proxy
```

Here Issuer and Subject strings start with top-level (DC) and end with low-level (CN) field, spaces and field separators depend on options used. None of these values will match in Zabbix Issuer and Subject fields!

To get proper Issuer and Subject strings usable in Zabbix invoke OpenSSL with special options

```bash
$ openssl x509 -noout -issuer -subject -nameopt esc_2253,esc_ctrl,utf8,dump_nostr,dump_der,sep_comma_plus,dn_rev,sname:\n   -in /home/zabbix/zabbix_proxy.crt
issuer= CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
subject= CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
```

Now string fields are in reverse order, fields are comma-separated, can be used in Zabbix configuration files and frontend.

Limitations on using X.509 v3 certificate extensions

- **Subject Alternative Name (subjectAltName)** extension.
  Alternative subject names from subjectAltName extension (like IP address, e-mail address) are not supported by Zabbix. Only value of “Subject” field can be checked in Zabbix (see Restricting allowed certificate Issuer and Subject).
  If certificate uses the subjectAltName extension then result depends on particular combination of crypto toolkits Zabbix components are compiled with (it may or may not work, Zabbix may refuse to accept such certificates from peers).

- **Extended Key Usage** extension.
  If used then generally both clientAuth (TLS WWW client authentication) and serverAuth (TLS WWW server authentication) are necessary.
  For example, in passive checks Zabbix agent acts in a TLS server role, so serverAuth must be set in agent certificate. For
active checks agent certificate needs clientAuth to be set.
GnuTLS issues a warning in case of key usage violation but allows communication to proceed.

- **Name Constraints** extension.
  Not all crypto toolkits support it. This extension may prevent Zabbix from loading CA certificates where this section is marked as critical (depends on particular crypto toolkit).

Certificate Revocation Lists (CRL)

If a certificate is compromised CA can revoke it by including in CRL. CRLs can be configured in server, proxy and agent configuration file using parameter TLSCRLFile. For example:

```
TLSCRLFile=/home/zabbix/zabbix_crl_file
```
where zabbix_crl_file may contain CRLs from several CAs and look like:

```
-----BEGIN X509 CRL-----
MIIB/DCB5QIBATANBgkqhkiG9w0BAQUFADCBgTETMBEGCgmSJomT8ixkARkWA2Nv...
-----END X509 CRL-----
-----BEGIN X509 CRL-----
MIIB+TCB4gIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLGQBGRYDY29t...
-----END X509 CRL-----
```

CRL file is loaded only on Zabbix start. CRL update requires restart.

If Zabbix component is compiled with OpenSSL and CRLs are used then each top and intermediate level CA in certificate chains must have a corresponding CRL (it can be empty) in TLSCRLFile.

### 2 Using pre-shared keys

**Overview**

Each pre-shared key (PSK) in Zabbix actually is a pair of:

- non-secret PSK identity string,
- secret PSK string value.

PSK identity string is a non-empty UTF-8 string. For example, “PSK ID 001 Zabbix agentd”. It is a unique name by which this specific PSK is referred to by Zabbix components. Do not put sensitive information in PSK identity string - it is transmitted over the network unencrypted.

PSK value is a hard to guess string of hexadecimal digits, for example, “e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9”.

**Size limits**

There are size limits for PSK identity and value in Zabbix, in some cases a crypto library can have lower limit:

<table>
<thead>
<tr>
<th>Component</th>
<th>PSK identity max size</th>
<th>PSK value min size</th>
<th>PSK value max size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zabbix</td>
<td>128 UTF-8 characters</td>
<td>128-bit (16-byte PSK, entered as 32 hexadecimal digits)</td>
<td>2048-bit (256-byte PSK, entered as 512 hexadecimal digits)</td>
</tr>
<tr>
<td>GnuTLS</td>
<td>128 bytes (may include UTF-8 characters)</td>
<td>-</td>
<td>2048-bit (256-byte PSK, entered as 512 hexadecimal digits)</td>
</tr>
<tr>
<td>OpenSSL 1.0.x, 1.1.0</td>
<td>127 bytes (may include UTF-8 characters)</td>
<td>-</td>
<td>2048-bit (256-byte PSK, entered as 512 hexadecimal digits)</td>
</tr>
<tr>
<td>OpenSSL 1.1.1</td>
<td>127 bytes (may include UTF-8 characters)</td>
<td>-</td>
<td>512-bit (64-byte PSK, entered as 128 hexadecimal digits)</td>
</tr>
<tr>
<td>OpenSSL 1.1.1a and later</td>
<td>127 bytes (may include UTF-8 characters)</td>
<td>-</td>
<td>2048-bit (256-byte PSK, entered as 512 hexadecimal digits)</td>
</tr>
</tbody>
</table>
Zabbix frontend allows configuring up to 128-character long PSK identity string and 2048-bit long PSK regardless of crypto libraries used.

If some Zabbix components support lower limits, it is the user’s responsibility to configure PSK identity and value with allowed length for these components.

Exceeding length limits results in communication failures between Zabbix components.

Before Zabbix server connects to agent using PSK, the server looks up the PSK identity and PSK value configured for that agent in database (actually in configuration cache). Upon receiving a connection the agent uses PSK identity and PSK value from its configuration file. If both parties have the same PSK identity string and PSK value the connection may succeed.

Each PSK identity must be paired with only one value. It is the user’s responsibility to ensure that there are no two PSKs with the same identity string but different values. Failing to do so may lead to unpredictable errors or disruptions of communication between Zabbix components using PSKs with this PSK identity string.

Generating PSK

For example, a 256-bit (32 bytes) PSK can be generated using the following commands:

- with OpenSSL:

  ```bash
  $ openssl rand -hex 32
  af8ced32def8714e548694e2d29e1a14ba6fa13f216cb35c19d0feb1084b0429
  ```

- with GnuTLS:

  ```bash
  $ psktool -u psk_identity -p database.psk -s 32
  Generating a random key for user 'psk_identity'
  Key stored to database.psk
  ```

  ```bash
  $ cat database.psk
  psk_identity:9b8eafedfaee00cece62e85d5f4792c79c99bce851b23216a1d300311cc4f7cb
  ```

Note that “psktool” above generates a database file with a PSK identity and its associated PSK. Zabbix expects just a PSK in the PSK file, so the identity string and colon (:) should be removed from the file.

Configuring PSK for server-agent communication (example)

On the agent host, write the PSK value into a file, for example, /home/zabbix/zabbix_agentd.psk. The file must contain PSK in the first text string, for example:

```
1f87b596725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952
```

Set access rights to PSK file - it must be readable only by Zabbix user.

Edit TLS parameters in agent configuration file zabbix_agentd.conf, for example, set:

```bash
TLSConnect=psk
TLSAccept=psk
TLSPSKFile=/home/zabbix/zabbix_agentd.psk
TLSPSKIdentity=PSK 001
```

The agent will connect to server (active checks) and accept from server and zabbix_get only connections using PSK. PSK identity will be “PSK 001”.

Restart the agent. Now you can test the connection using zabbix_get, for example:

```bash
$ zabbix_get -s 127.0.0.1 -k "system.cpu.load[all,avg1]" --tls-connect=psk --tls-psk-identity="PSK 001" --tls-psk-file=/home/zabbix/zabbix_agentd.psk
```

(To minimize downtime see how to change connection type in Connection encryption management).

Configure PSK encryption for this agent in Zabbix frontend:

- Go to: Configuration → Hosts
- Select host and click on Encryption tab

Example:
All mandatory input fields are marked with a red asterisk.

When configuration cache is synchronized with database the new connections will use PSK. Check server and agent logfiles for error messages.

Configuring PSK for server - active proxy communication (example)

On the proxy, write the PSK value into a file, for example, /home/zabbix/zabbix_proxy.psk. The file must contain PSK in the first text string, for example:
e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9

Set access rights to PSK file - it must be readable only by Zabbix user.

Edit TLS parameters in proxy configuration file zabbix_proxy.conf, for example, set:

```
TLSConnect=psk
TLSPSKFile=/home/zabbix/zabbix_proxy.psk
TLSPSKIdentity=PSK 002
```

The proxy will connect to server using PSK. PSK identity will be ”PSK 002”.

(To minimize downtime see how to change connection type in Connection encryption management).

Configure PSK for this proxy in Zabbix frontend. Go to Administration→Proxies, select the proxy, go to ”Encryption” tab. In ”Connections from proxy” mark PSK. Paste into ”PSK identity” field ”PSK 002” and ”e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9” into ”PSK” field. Click ”Update”.

Restart proxy. It will start using PSK-based encrypted connections to server. Check server and proxy logfiles for error messages.

For a passive proxy the procedure is very similar. The only difference - set TLSServerCertificate=psk in proxy configuration file and set “Connections to proxy” in Zabbix frontend to PSK.

3 Troubleshooting

General recommendations

- Start with understanding which component acts as a TLS client and which one acts as a TLS server in problem case. Zabbix server, proxies and agents, depending on interaction between them, all can work as TLS servers and clients. For example, Zabbix server connecting to agent for a passive check, acts as a TLS client. The agent is in role of TLS server. Zabbix agent, requesting a list of active checks from proxy, acts as a TLS client. The proxy is in role of TLS server. zabbix_get and zabbix_sender utilities always act as TLS clients.

- Zabbix uses mutual authentication. Each side verifies its peer and may refuse connection.

For example, Zabbix server connecting to agent can close connection immediately if agent's certificate is invalid. And vice versa - Zabbix agent accepting a connection from server can close connection if server is not trusted by agent.
• Examine logfiles in both sides - in TLS client and TLS server. The side which refuses connection may log a precise reason why it was refused. Other side often reports rather general error (e.g. “Connection closed by peer”, “connection was non-properly terminated”).
• Sometimes misconfigured encryption results in confusing error messages in no way pointing to real cause.

In subsections below we try to provide a (far from exhaustive) collection of messages and possible causes which could help in troubleshooting.

Please note that different crypto toolkits (OpenSSL, GnuTLS) often produce different error messages in same problem situations.

Sometimes error messages depend even on particular combination of crypto toolkits on both sides.

1 Connection type or permission problems

Server is configured to connect with PSK to agent but agent accepts only unencrypted connections

In server or proxy log (with GnuTLS 3.3.16)

Get value from agent failed: zbx_tls_connect(): gnutls_handshake() failed: \\
-110 The TLS connection was non-properly terminated.

In server or proxy log (with OpenSSL 1.0.2c)

Get value from agent failed: TCP connection successful, cannot establish TLS to [[127.0.0.1]:10050]: \\
Connection closed by peer. Check allowed connection types and access rights

One side connects with certificate but other side accepts only PSK or vice versa

In any log (with GnuTLS):

failed to accept an incoming connection: from 127.0.0.1: zbx_tls_accept(): gnutls_handshake() failed: \\
-21 Could not negotiate a supported cipher suite.

In any log (with OpenSSL 1.0.2c):

failed to accept an incoming connection: from 127.0.0.1: TLS handshake returned error code 1:
file .\ssl\s3_srvr.c line 1411: error:1408A0C1:SSL routines:ssl3_get_client_hello:no shared cipher: \\
TLS write fatal alert "handshake failure"

Attempting to use Zabbix sender compiled with TLS support to send data to Zabbix server/proxy compiled without TLS

In connecting-side log:

Linux:

...In zbx_tls_init_child()
...OpenSSL library (version OpenSSL 1.1.1 11 Sep 2018) initialized
...
...In zbx_tls_connect(): psk_identity:"PSK test sender"
...End of zbx_tls_connect():FAIL error:'connection closed by peer'
...send value error: TCP successful, cannot establish TLS to [[localhost]:10051]: connection closed by peer

Windows:

...OpenSSL library (version OpenSSL 1.1.1a 20 Nov 2018) initialized
...
...In zbx_tls_connect(): psk_identity:"PSK test sender"
...zbx_psk_client_cb() requested PSK identity "PSK test sender"
...End of zbx_tls_connect():FAIL error:’SSL_connect() I/O error: [0x00000000] The operation completed successfully.’
...send value error: TCP successful, cannot establish TLS to [[192.168.1.2]:10051]: SSL_connect() I/O error

In accepting-side log:

...failed to accept an incoming connection: from 127.0.0.1: support for TLS was not compiled in

One side connects with PSK but other side uses LibreSSL or has been compiled without encryption support

LibreSSL does not support PSK.

In connecting-side log:

...TCP successful, cannot establish TLS to [[192.168.1.2]:10050]: SSL_connect() I/O error: [0] Success

In accepting-side log:

...failed to accept an incoming connection: from 192.168.1.2: support for PSK was not compiled in
In Zabbix frontend:
Get value from agent failed: TCP successful, cannot establish TLS to \[[192.168.1.2]:10050\]: SSL_connect()
One side connects with PSK but other side uses OpenSSL with PSK support disabled
In connecting-side log:
...TCP successful, cannot establish TLS to \[[192.168.1.2]:10050\]: SSL_connect() set result code to SSL_ERROR_SSL
In accepting-side log:
...failed to accept an incoming connection: from 192.168.1.2: TLS handshake set result code to 1: file ssl

2 Certificate problems

OpenSSL used with CRLs and for some CA in the certificate chain its CRL is not included in TLSCRLFile
In TLS server log in case of OpenSSL peer:
failed to accept an incoming connection: from 127.0.0.1: TLS handshake with 127.0.0.1 returned error code
   file s3_srvr.c line 3251: error:14089086: SSL routines:ssl3_get_client_certificate:certificate verify failed:
   TLS write fatal alert "unknown CA"
In TLS server log in case of GnuTLS peer:
failed to accept an incoming connection: from 127.0.0.1: TLS handshake with 127.0.0.1 returned error code
   file rsa_pk1.c line 103: error:0407006A: rsa routines:RSA_padding_check_PKCS1_type_1:
   block type is not 01 file rsa_eay.c line 705: error:04067072: rsa routines:RSA_EAY_PUBLIC_DECRYPT:
   CRL expired or expires during server operation
OpenSSL, in server log:
   • before expiration:
   cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to \[[127.0.0.1]:20004\]
      SSL_connect() returned SSL_ERROR_SSL: file s3_clnt.c line 1253: error:14090086:
      SSL routines:ssl3_get_server_certificate:certificate verify failed:
      TLS write fatal alert "certificate revoked"
   • after expiration:
   cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to \[[127.0.0.1]:20004\]
      SSL_connect() returned SSL_ERROR_SSL: file s3_clnt.c line 1253: error:14090086:
      SSL routines:ssl3_get_server_certificate:certificate verify failed:
      TLS write fatal alert "certificate expired"

The point here is that with valid CRL a revoked certificate is reported as "certificate revoked". When CRL expires the error message changes to "certificate expired" which is quite misleading.
GnuTLS, in server log:
   • before and after expiration the same:
   cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to \[[127.0.0.1]:20004\]
      invalid peer certificate: The certificate is NOT trusted. The certificate chain is revoked.
Self-signed certificate, unknown CA
OpenSSL, in log:
   error:'self signed certificate: SSL_connect() set result code to SSL_ERROR_SSL: file ..ssl/statem/statem_
      line 1924: error:1416F086:SSL routines:tls_process_server_certificate:certificate verify failed:
      TLS write fatal alert "unknown CA"

This was observed when server certificate by mistake had the same Issuer and Subject string, although it was signed by CA. Issuer and Subject are equal in top-level CA certificate, but they cannot be equal in server certificate. (The same applies to proxy and agent certificates.)
3 PSK problems

PSK contains an odd number of hex-digits
Proxy or agent does not start, message in the proxy or agent log:
invalid PSK in file "/home/zabbix/zabbix_proxy.psk"
PSK identity string longer than 128 bytes is passed to GnuTLS
In TLS client side log:
gnutls_handshake() failed: -110 The TLS connection was non-properly terminated.
In TLS server side log.
gnutls_handshake() failed: -90 The SRP username supplied is illegal.
Too long PSK value used with OpenSSL 1.1.1
In connecting-side log:
...OpenSSL library (version OpenSSL 1.1.1 11 Sep 2018) initialized
...
...In zbx_tls_connect(): psk_identity:"PSK 1"
...zbx_psk_client_cb() requested PSK identity "PSK 1"
...End of zbx_tls_connect():FAIL error:'SSL_connect() set result code to SSL_ERROR_SSL: file ssl\statem\extensions_clnt.c line 801: error:14212044:SSL routines:tls_construct_ctos_early_data:internal error: TLS write fatal alert "internal error'" 
In accepting-side log:
...Message from 123.123.123.123 is missing header. Message ignored.
This problem typically arises when upgrading OpenSSL from 1.0.x or 1.1.0 to 1.1.1 and if the PSK value is longer than 512-bit (64-byte PSK, entered as 128 hexadecimal digits).
See also: Value size limits

18. Web interface

Overview  For an easy access to Zabbix from anywhere and from any platform, the web-based interface is provided.
If using more than one frontend instance make sure that the locales and libraries (LDAP, SAML etc.) are installed and configured identically for all frontends.

Frontend help  A help link is provided in Zabbix frontend forms with direct links to the corresponding parts of the documentation.

1 Menu

Overview
A vertical menu in a sidebar provides access to various Zabbix frontend sections.
The menu is dark blue in the default theme.
Working with the menu

A **global search** box is located below the Zabbix logo.

The menu can be collapsed or hidden completely:

- To collapse, click on next to Zabbix logo
- To hide, click on next to Zabbix logo

Collapsed menu

When the menu is collapsed to icons only, a full menu reappears as soon as the mouse cursor is placed upon it. Note that it reappears over page content; to move page content to the right you have to click on the expand button. If the mouse cursor again is placed outside the full menu, the menu will collapse again after two seconds.

You can also make a collapsed menu reappear fully by hitting the Tab key. Hitting the Tab key repeatedly will allow to focus on the next menu element.
Hidden menu

Even when the menu is hidden completely, a full menu is just one mouse click away, by clicking on the burger icon. Note that it reappears over page content; to move page content to the right you have to unhide the menu by clicking on the show sidebar button.

2 Frontend sections

1 Monitoring

Overview

The Monitoring menu is all about displaying data. Whatever information Zabbix is configured to gather, visualize and act upon, it will be displayed in the various sections of the Monitoring menu.

View mode buttons

The following buttons located in the top right corner are common for every section:

Display page in kiosk mode. In this mode only page content is displayed.

To exit kiosk mode, move the mouse cursor until the exit button appears and click on it. You will be taken back to normal mode.

1 Dashboard

Overview

The Monitoring → Dashboard section is designed to display summaries of all the important information in a dashboard.

While only one dashboard can displayed at one time, it is possible to configure several dashboards. Each dashboard may contain one or several pages that can be rotated in a slideshow.

A dashboard page consists of widgets and each widget is designed to display information of a certain kind and source, which can be a summary, a map, a graph, the clock, etc.

1.36 \[\text{Zabbix server Values per second}\]

Pages and widgets are added to the dashboard and edited in the dashboard editing mode. Pages can be viewed and rotated in the dashboard viewing mode.

The time period that is displayed in graph widgets is controlled by the time period selector located above the widgets. The time period selector label, located to the right, displays the currently selected time period. Clicking the tab label allows expanding and collapsing the time period selector.
Note that when the dashboard is displayed in kiosk mode and widgets only are displayed, it is possible to zoom out the graph period by double-clicking in the graph.

Viewing dashboards

To view all configured dashboards, click on All dashboards just below the section title.

Dashboards are displayed with a sharing tag:

- **My** - indicates a private dashboard
- **Shared** - indicates a public dashboard or a private dashboard shared with any user or user group

The filter located to the right above the list allows to filter dashboards by name and by those created by the current user.

To delete one or several dashboards, mark the checkboxes of the respective dashboards and click on Delete below the list.

Viewing a dashboard

To view a single dashboard, click on its name in the list of dashboards.

When viewing a dashboard, the following options are available:

- **Edit dashboard**: Switch to the dashboard editing mode. The editing mode is also opened when a new dashboard is being created and when you click on the "edit" button of a widget.
- **Open the action menu (see action descriptions below).**
Sharing - edit sharing preferences for the dashboard.
Create new - create a new dashboard.
Clone - create a new dashboard by copying properties of the existing one. First you are prompted to enter dashboard parameters. Then, the new dashboard opens in editing mode with all the widgets of the original dashboard.
Delete - delete the dashboard.
Create new report - open a pop-up window with report configuration form. Disabled if the user does not have permission to manage scheduled reports.
View related reports - open a pop-up window with a list of existing reports based on the current dashboard. Disabled if there are no related reports or the user does not have permission to view scheduled reports.
Display only page content (kiosk mode). Kiosk mode can also be accessed with the following URL parameters: /zabbix.php?action=dashboard.view&kiosk=1.
To exit to normal mode: /zabbix.php?action=dashboard.view&kiosk=0

Sharing

Dashboards can be made public or private.

Public dashboards are visible to all users. Private dashboards are visible only to their owner. Private dashboards can be shared by the owner with other users and user groups.

The sharing status of a dashboard is displayed in the list of all dashboards. To edit the sharing status of a dashboard, click on the Sharing option in the action menu when viewing a single dashboard:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Select dashboard type:</td>
</tr>
<tr>
<td></td>
<td><strong>Private</strong> - dashboard is visible only to selected user groups and users</td>
</tr>
<tr>
<td></td>
<td><strong>Public</strong> - dashboard is visible to all</td>
</tr>
<tr>
<td>List of user group shares</td>
<td>Select user groups that the dashboard is accessible to.</td>
</tr>
<tr>
<td></td>
<td>You may allow read-only or read-write access.</td>
</tr>
<tr>
<td>List of user shares</td>
<td>Select users that the dashboard is accessible to.</td>
</tr>
<tr>
<td></td>
<td>You may allow read-only or read-write access.</td>
</tr>
</tbody>
</table>

Editing a dashboard

When editing a dashboard, the following options are available:

- Edit general dashboard parameters.
- Add a new widget.
- Clicking on the arrow button will open the action menu (see action descriptions below).
Add widget - add a new widget
Add page - add a new page
Paste widget - paste a copied widget. This option is grayed out if no widget has been copied.
Only one entity (widget or page) can be copied at one time.
Paste page - paste a copied page. This option is grayed out if no page has been copied.
Save dashboard changes.
Cancel dashboard changes.

Creating a dashboard

It is possible to create a new dashboard in two ways:

- Click on Create dashboard, when viewing all dashboards
- Select Create new from the action menu, when viewing a single dashboard

You will be first asked to enter general dashboard parameters:

![Dashboard properties](image)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td>Select system user that will be the dashboard owner.</td>
</tr>
<tr>
<td>Name</td>
<td>Enter dashboard name.</td>
</tr>
<tr>
<td>Default page display period</td>
<td>Select period for how long a dashboard page is displayed before rotating to the next page in a slideshow.</td>
</tr>
<tr>
<td>Start slideshow automatically</td>
<td>Mark this checkbox to run a slideshow automatically one more than one dashboard page exists.</td>
</tr>
</tbody>
</table>

When you click on Apply, an empty dashboard is opened:
To populate the dashboard, you can add widgets and pages.

Click on the Save changes button to save the dashboard. If you click on Cancel, the dashboard will not be created.

Adding widgets

To add a widget to a dashboard:

- Click on the `Add` button or the Add widget option in the action menu that can be opened by clicking on the arrow. Fill the widget configuration form. The widget will be created in its default size and placed after the existing widgets (if any);

Or

- Move your mouse to the desired empty spot for the new widget. Notice how a placeholder appears, onmouseover, on any empty slot on the dashboard. Then click to open the widget configuration form. After filling the form the widget will be created in its default size or, if its default size is bigger than is available, take up the available space. Alternatively, you may click and drag the placeholder to the desired widget size, then release, and then fill the widget configuration form. (Note that when there is a widget copied onto the clipboard, you will be first prompted to select between Add widget and Paste widget options to create a widget.)

In the widget configuration form:

- Select the Type of widget
- Enter widget parameters
- Click on Add
Widgets

The following widgets can be added to a dashboard:

- Action log
- Clock
- Data overview
- Discovery status
- Favorite graphs
- Favorite maps
- Geomap
- Graph
- Graph (classic)
- Graph prototype
- Host availability
- Item value
- Map
- Map navigation tree
- Plain text
- Problem hosts
- Problems
- System information
- Problems by severity
- Top hosts
- Trigger overview
- URL
- Web monitoring

In dashboard editing mode widgets can be resized and moved around the dashboard by clicking on the widget title bar and dragging it to a new location. Also, you can click on the following buttons in the top-right corner of the widget to:

- ⚙️ - edit a widget;
- ⋯⋯⋯ - access the widget menu

Click on Save changes for the dashboard to make any changes to the widgets permanent.

Copying/pasting widgets

Dashboard widgets can be copied and pasted, allowing to create a new widget with the properties of an existing one. They can be copy-pasted within the same dashboard, or between dashboards opened in different tabs.
A widget can be copied using the widget menu. To paste the widget:

- click on the arrow next to the Add button and selecting the Paste widget option, when editing the dashboard
- use the Paste widget option when adding a new widget by selecting some area in the dashboard (a widget must be copied first for the paste option to become available)

A copied widget can be used to paste over an existing widget using the Paste option in the widget menu.

Creating a slideshow

A slideshow will run automatically if the dashboard contains two or more pages (see Adding pages) and if one of the following is true:

- The Start slideshow automatically option is marked in dashboard properties
- The dashboard URL contains a slideshow=1 parameter

The pages rotate according to the intervals given in the properties of the dashboard and individual pages. Click on:

- Stop slideshow - to stop the slideshow
- Start slideshow - to start the slideshow

![Slideshow controls](image)

Slideshow-related controls are also available in kiosk mode (where only the page content is shown):

- stop slideshow
- start slideshow
- go back one page
- go to the next page

Adding pages

To add a new page to a dashboard:

- Make sure the dashboard is in the editing mode
- Click on the arrow next to the Add button and select the Add page option

![Add page options](image)

- Fill the general page parameters and click on Apply. If you leave the name empty, the page will be added with a Page \( N \) name where 'N' is the incremental number of the page. The page display period allows to customize how long a page is displayed in a slideshow.
A new page will be added, indicated by a new tab (Page 2).

The pages can be reordered by dragging-and-dropping the page tabs. Reordering maintains the original page naming. It is always possible to go to each page by clicking on its tab.

When a new page is added, it is empty. You can add widgets to it as described above.

Copying/pasting pages

Dashboard pages can be copied and pasted, allowing to create a new page with the properties of an existing one. They can be pasted from the same dashboard or a different dashboard.

To paste an existing page to the dashboard, first copy it, using the page menu:

To paste the copied page:

- Make sure the dashboard is in the editing mode
- Click on the arrow next to the Add button and select the Paste page option

Page menu

The page menu can be opened by clicking on the three dots next to the page name:

It contains the following options:

- Copy - copy the page
- Delete - delete the page (pages can only be deleted in the dashboard editing mode)
- Properties - customize the page parameters (the name and the page display period in a slideshow)

Widget menu

The widget menu contains different options based on whether the dashboard is in the edit or view mode:

<table>
<thead>
<tr>
<th>Widget menu</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>In dashboard edit mode:</td>
<td>Copy - copy the widget</td>
</tr>
<tr>
<td></td>
<td>Paste - paste a copied widget over this widget</td>
</tr>
<tr>
<td></td>
<td>This option is grayed out if no widget has been copied.</td>
</tr>
<tr>
<td></td>
<td>Delete - delete the widget</td>
</tr>
</tbody>
</table>

| In dashboard view mode: | Copy - copy the widget                      |
|                        | Download image - download the widget as a PNG image (only available for graph/classic graph widgets) |
|                        | Refresh interval - select the frequency of refreshing the widget contents |

Dynamic widgets

When configuring some of the widgets:

- Graphs (simple and custom)
- Item value
- Plain text
- URL

there is an extra option called Dynamic item. You can check this box to make the widget dynamic - i.e. capable of displaying different content based on the selected host.

Now, when saving the dashboard, you will notice that a new host selection field has appeared atop the dashboard for selecting the host (while the Select button allows selecting the host group in a popup):
Thus you have a widget, which can display content that is based on the data from the host that is selected. The benefit of this is that you do not need to create extra widgets just because, for example, you want to see the same graphs containing data from various hosts.

Permissions to dashboards

Permissions to dashboards for regular users and users of ‘Admin’ type are limited in the following way:

- They can see and clone a dashboard if they have at least READ rights to it;
- They can edit and delete dashboard only if they have READ/WRITE rights to it;
- They cannot change the dashboard owner.

Host menu

Clicking on a host in the Problems widget brings up the host menu. It includes links to host inventory, latest data, problems, graphs, dashboards, web scenarios and configuration. Note that host configuration is available for Admin and Superadmin users only.

Global scripts can also be run from the host menu. These scripts need to have their scope defined as ‘Manual host action’ to be available in the host menu.

The host menu is accessible by clicking on a host in several other frontend sections:

- Monitoring → Problems
- Monitoring → Problems → Event details
- Monitoring → Hosts
- Monitoring → Hosts → Web Monitoring
- Monitoring → Latest data
- Monitoring → Maps
- Reports → Triggers top 100

Problem event popup

The problem event popup includes the list of problem events for this trigger and, if defined, the trigger description and a clickable URL.
To bring up the problem event popup:

- roll a mouse over the problem duration in the Duration column of the Problems widget. The popup disappears once you remove the mouse from the duration.
- click on the duration in the Duration column of the Problems widget. The popup disappears only if you click on the duration again.

Resolved values of \{ITEM.VALUE\} and \{ITEM.LASTVALUE\} macros in trigger descriptions are truncated to 20 characters. To see the entire values you may use macro functions with these macros, e.g. \{{ITEM.VALUE}.regsub("(.*)", \1)\}, \{{ITEM.LASTVALUE}.regsub("(.*)", \1)\} as a workaround.

Dashboard widgets

Overview

This section provides the details of parameters that are common for **dashboard** widgets.

Common parameters

The following parameters are common for every single widget:

<table>
<thead>
<tr>
<th>Name</th>
<th>Enter a widget name.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh interval</td>
<td>Configure default refresh interval. Default refresh intervals for widgets range from <strong>No refresh</strong> to 15 minutes depending on the type of widget. For example: <strong>No refresh</strong> for URL widget, 1 minute for action log widget, 15 minutes for clock widget.</td>
</tr>
<tr>
<td>Show header</td>
<td>Mark the checkbox to show the header permanently. When unchecked the header is hidden to save space and only slides up and becomes visible again when the mouse is positioned over the widget, both in view and edit modes. It is also semi-visible when dragging a widget to a new place.</td>
</tr>
</tbody>
</table>

Refresh intervals for a widget can be set to a default value for all the corresponding users and also each user can set his own refresh interval value:

- To set a default value for all the corresponding users switch to editing mode (click the Edit dashboard button, find the right widget, click the Edit button opening the editing form of a widget), and choose the required refresh interval from the dropdown list.
- Setting a unique refresh interval for each user separately is possible in view mode by clicking the ***** button for a certain widget.

Unique refresh interval set by a user has priority over the widget setting and once it’s set it’s always preserved when the widget’s setting is modified.

To see **specific parameters** for each widget, go to individual widget pages for:

- Action log
- Clock
- Discovery status
- Favorite graphs
- Favorite maps
- Geomap
- Graph
- Graph (classic)
- Graph prototype
- Host availability
- Item value
- Map
- Map navigation tree
- Plain text
- Problem hosts
- Problems
- SLA report
- System information
- Problems by severity
- Top hosts
- Trigger overview
• URL
• Web monitoring

Deprecated widgets:
• Data overview

Deprecated widgets will be removed in upcoming major release.

1 Action log

Overview

In the action log widget, you can display details of action operations (notifications, remote commands). It replicates information from Administration → Audit.

Configuration

To configure, select Action log as type:

![Add widget](image)

In addition to the parameters that are common for all widgets, you may set the following specific options:

<table>
<thead>
<tr>
<th>Sort entries by</th>
<th>Sort entries by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (descending or ascending)</td>
<td></td>
</tr>
<tr>
<td>Type (descending or ascending)</td>
<td></td>
</tr>
<tr>
<td>Status (descending or ascending)</td>
<td></td>
</tr>
<tr>
<td>Recipient (descending or ascending)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Show lines</th>
<th>Set how many action log lines will be displayed in the widget.</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

2 Clock

Overview

In the clock widget, you may display local, server, or specified host time.

Both analog and digital clocks can be displayed:
Configuration

To configure, select Clock as type:

Edit widget

Type: Clock
Name: Local time
Refresh interval: Default (15 minutes)
Time type: Local time
Clock type: Analog

Show: Date, Time, Time zone

Advanced configuration

Apply  Cancel
In addition to the parameters that are common for all widgets, you may set the following specific options:

<table>
<thead>
<tr>
<th>Time type</th>
<th>Select local, server, or specified host time. Server time will be identical to the time zone set globally or for the Zabbix user.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Select the item for displaying time. To display host time, use the <code>system localtime[local]</code> item. This item must exist on the host. This field is available only when Host time is selected.</td>
</tr>
</tbody>
</table>
| Clock type| Select clock type: **Analog** - analog clock  
|           | **Digital** - digital clock |
| Show      | Select information units to display in the digital clock (date, time, time zone). This field is available only if “Digital” is selected in the Clock type field. |
| Advanced configuration | Mark the checkbox to display advanced configuration options for the digital clock. This field is available only if “Digital” is selected in the Clock type field. |

Advanced configuration options become available if the Advanced configuration checkbox is marked (see screenshot) and only for those elements that are selected in the Show field (see above).

Additionally, advanced configuration allows to change the background color for the whole widget.

![Advanced configuration settings](image)

**Background color** Select the background color from the color picker.  
D stands for default color (depends on the frontend theme). To return to the default value, click the Use default button in the color picker.

**Date**
- **Size** Enter font size height for the date (in percent relative to total widget height).
- **Bold** Mark the checkbox to display date in bold type.
- **Color** Select the date color from the color picker.  
D stands for default color (depends on the frontend theme). To return to the default value, click the Use default button in the color picker.

**Time**
- **Size** Enter font size height for the time (in percent relative to total widget height).
- **Bold** Mark the checkbox to display time in bold type.
- **Color** Select the time color from the color picker.  
D stands for default color (depends on the frontend theme). To return to the default value, click the Use default button in the color picker.
- **Seconds** Mark the checkbox to display seconds. Otherwise only hours and minutes will be displayed.
- **Format** Select to display a 24-hour or 12-hour time.

**Time zone**
- **Size** Enter font size height for the time zone (in percent relative to total widget height).
- **Bold** Mark the checkbox to display time zone in bold type.
3 Data overview

This widget is deprecated and will be removed in the upcoming major release.

Overview

In the data overview widget, you can display the latest data for a group of hosts.

The color of problem items is based on the problem severity color, which can be adjusted in the problem update screen.

By default, only values that fall within the last 24 hours are displayed. This limit has been introduced with the aim of improving initial loading times for large pages of latest data. This limit is configurable in Administration → General → GUI, using the Max history display period option.

Clicking on a piece of data offers links to some predefined graphs or latest values.

Note that 50 records are displayed by default (configurable in Administration → General → GUI, using the Max number of columns and rows in overview tables option). If more records exist than are configured to display, a message is displayed at the bottom of the table, asking to provide more specific filtering criteria. There is no pagination.

Configuration

To configure, select Data overview as type:

In addition to the parameters that are common for all widgets, you may set the following specific options:
Host groups Select host groups. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Scroll down to select. Click on 'x' to remove the selected.

Hosts Select hosts. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. Scroll down to select. Click on 'x' to remove the selected.

Tags Specify tags to limit the number of item data displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive.

There are several operators available for each condition:
- **Exists** - include the specified tag names
- **Equals** - include the specified tag names and values (case-sensitive)
- **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)
- **Does not exist** - exclude the specified tag names
- **Does not equal** - exclude the specified tag names and values (case-sensitive)
- **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)

There are two calculation types for conditions:
- **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition
- **Or** - enough if one condition is met

Show suppressed problems Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance.

Hosts location Select host location - left or top.

---

4 Discovery status

Overview

This widget displays a status summary of the active network discovery rules.

![Add widget](image)

All configuration parameters are **common** for all widgets.

5 Favorite graphs

Overview

This widget contains shortcuts to the most needed graphs, sorted alphabetically.

The list of shortcuts is populated when you view a graph and then click on its [Add to favorites button].

All configuration parameters are **common** for all widgets.

6 Favorite maps
Overview

This widget contains shortcuts to the most needed maps, sorted alphabetically.

The list of shortcuts is populated when you view a map and then click on its Add to favorites button.

All configuration parameters are common for all widgets.

7 Geomap

Overview

Geomap widget displays hosts as markers on a geographical map using open-source JavaScript interactive maps library Leaflet. Zabbix offers multiple predefined map tile service providers and an option to add a custom tile service provider or even host tiles themselves (configurable in the Administration → General → Geographical maps menu section).

By default, the widget displays all enabled hosts with valid geographical coordinates defined in the host configuration. It is possible to configure host filtering in the widget parameters.

The valid host coordinates are:

- Latitude: from -90 to 90 (can be integer or float number)
- Longitude: from -180 to 180 (can be integer or float number)

Configuration

To add the widget, select Geomap as type.

In addition to the parameters that are common for all widgets, you may set the following specific options:

**Host groups**

Select host groups to be displayed on the map. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Scroll down to select. Click on ‘x’ to remove selected groups.

If nothing is selected in both Host groups and Hosts fields, all hosts with valid coordinates will be displayed.
Hosts

Select hosts to be displayed all the map. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. Scroll down to select. Click on ‘x’ to remove selected hosts.

If nothing is selected in both Host groups and Hosts fields, all hosts with valid coordinates will be displayed.

Tags

Specify tags to limit the number of hosts displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive.

There are several operators available for each condition:

Exists - include the specified tag names

Equals - include the specified tag names and values (case-sensitive)

Contains - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)

Does not exist - exclude the specified tag names

Does not equal - exclude the specified tag names and values (case-sensitive)

Does not contain - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)

There are two calculation types for conditions:

And/Or - all conditions must be met, conditions having the same tag name will be grouped by the Or condition

Or - enough if one condition is met

Initial view

Comma-separated center coordinates and an optional zoom level to display when the widget is initially loaded in the format <latitude>,<longitude>,<zoom>

If initial zoom is specified, the Geomap widget is loaded at the given zoom level. Otherwise, initial zoom is calculated as half of the max zoom for the particular tile provider.

The initial view is ignored if the default view is set (see below).

Examples:

=> 40.6892494,-74.0466891,14
=> 40.6892494,-122.0466891

Host markers displayed on the map have the color of the host’s most serious problem and green color if a host has no problems. Clicking on a host marker allows viewing the host’s visible name and the number of unresolved problems grouped by severity. Clicking on the visible name will open host menu.

Hosts displayed on the map can be filtered by problem severity. Press on the filter icon in the widget’s upper right corner and mark the required severities.

It is possible to zoom in and out the map by using the plus and minus buttons in the widget’s upper left corner or by using the mouse scroll wheel or touchpad. To set the current view as default, right-click anywhere on the map and select Set this view as default. This setting will override Initial view widget parameter for the current user. To undo this action, right-click anywhere on the map again and select Reset the initial view.

When Initial view or Default view is set, you can return to this view at any time by pressing on the home icon on the left.
Overview

The graph widget provides a modern and versatile way of visualizing data collected by Zabbix using a vector image drawing technique. This graph widget is supported since Zabbix 4.0. Note that the graph widget supported before Zabbix 4.0 can still be used as Graph (classic).

Configuration

To configure, select Graph as type:

The Data set tab allows to add data sets and define their visual representation:
Data set
Enter the host and item patterns; data of items that match the entered patterns will be
displayed on the graph. Wildcard patterns may be used (for example, * will return results
that match zero or more characters). To specify a wildcard pattern, just enter the string
manually and press Enter. While you are typing, note how all matching hosts are displayed in
the dropdown.
Up to 50 items may be displayed in the graph.
Host pattern and item pattern fields are mandatory.
The wildcard symbol is always interpreted, therefore it is not possible to add, for example, an
item named “item**” individually, if there are other matching items (e.g. item2, item3).
Alternatively to specifying item patterns, you may select a list of items, if the data set has
been added with the Item list option (see description of the Add new data set button).

Draw
Choose the draw type of the metric. Possible draw types are Line (set by default), Points,
Staircase and Bar.
Note that if there’s only one data point in the line/staircase graph it is drawn as a point
regardless of the draw type. The point size is calculated from the line width, but it cannot be
smaller than 3 pixels, even if the line width is less.

Stacked
Mark the checkbox to display data as stacked (filled areas displayed). This option is disabled
when Points draw type is selected.

Width
Set the line width. This option is available when Line or Staircase draw type is selected.

Point size
Set the point size. This option is available when Points draw type is selected.

Transparency
Set the transparency level.

Fill
Set the fill level. This option is available when Line or Staircase draw type is selected.

Missing data
Select the option for displaying missing data:
None - the gap is left empty
Connected - two border values are connected
Treat as 0 - the missing data is displayed as 0 values
Last known - the missing data is displayed with the same value as the last known value
Not applicable for the Points and Bar draw type.

Y-axis
Select the side of the graph where the Y-axis will be displayed.

Time shift
Specify time shift if required. You may use time suffixes in this field. Negative values are
allowed.

Aggregation function
Specify which aggregation function to use:
min - display the smallest value
max - display the largest value
avg - display the average value
sum - display the sum of values
count - display the count of values
first - display the first value
last - display the last value
none - display all values (no aggregation)
Aggregation allows to display an aggregated value for the chosen interval (5 minutes, an
hour, a day), instead of all values. See also: Aggregation in graphs.

Aggregation interval
Specify the interval for aggregating values. You may use time suffixes in this field. A numeric
value without a suffix will be regarded as seconds.

Aggregate
Specify whether to aggregate:
Each item - each item in the dataset will be aggregated and displayed separately.
Dataset - all dataset items will be aggregated and displayed as one value.

Approximation
Specify the approximation option:
all - display all values (min/max/avg)
min - display the smallest values
max - display the largest values
avg - display the average values

Existing data sets are displayed in a list. You may:

- click on the move icon and drag a data set to a new place in the list
- click on the expand icon to expand data set details. When expanded, this icon turns into a collapse icon.
- click on the color icon to change the base color, either from the color picker or manually. The base color is used to
calculate different colors for each item of the data set.
- Click on this button to add an empty data set allowing to select the host/item pattern.

- If you click on the downward pointing icon next to the Add new data set button, a drop-down menu appears allowing to add a new data set with item pattern/item list or by cloning the currently open data set. If all data sets are collapsed, the Clone option is not available.

The **Displaying options** tab allows to define history data selection:

<table>
<thead>
<tr>
<th>Data set 2</th>
<th>Displaying options</th>
<th>Time period</th>
</tr>
</thead>
<tbody>
<tr>
<td>History data selection</td>
<td>Auto</td>
<td>History</td>
</tr>
<tr>
<td>Simple triggers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working time</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**History data selection**
Set the source of graph data:
- **Auto** - data are sourced according to the classic graph algorithm (default)
- **History** - data from history
- **Trends** - data from trends

**Simple triggers**
Mark the checkbox to show simple triggers as lines with black dashes over the trigger severity color.

**Working time**
Mark the checkbox to show working time on the graph. Working time (working days) is displayed in graphs as a white background, while non-working time is displayed in gray (with the Original blue default frontend theme).

**Percentile line (left)**
Mark the checkbox and enter the percentile value to show the specified percentile as a line on the left Y-axis of the graph.
If, for example, a 95% percentile is set, then the percentile line will be at the level where 95 percent of the values fall under.

**Percentile line (right)**
Mark the checkbox and enter the percentile value to show the specified percentile as a line on the right Y-axis of the graph.
If, for example, a 95% percentile is set, then the percentile line will be at the level where 95 percent of the values fall under.

The **Time period** tab allows to set a custom time period:

<table>
<thead>
<tr>
<th>Data set 2</th>
<th>Displaying options</th>
<th>Time period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set custom time period</td>
<td></td>
<td></td>
</tr>
<tr>
<td>From</td>
<td>now-1h</td>
<td>To</td>
</tr>
</tbody>
</table>
Set custom time period

Mark this checkbox to set the custom time period for the graph (unmarked by default).

From

Set the start time of the custom time period for the graph.

To

Set the end time of the custom time period for the graph.

The **Axes** tab allows to customize how axes are displayed:

<table>
<thead>
<tr>
<th>Data set 2</th>
<th>Displaying options</th>
<th>Time period</th>
<th>Axes</th>
<th>Legend</th>
<th>Problems</th>
<th>Overrides</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Left Y</strong></td>
<td>Show</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>calculated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>calculated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Units</td>
<td>Auto value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Right Y</strong></td>
<td>Show</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Min</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Units</td>
<td>Auto value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>X-Axis</strong></td>
<td>Show</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Left Y**  
  Mark this checkbox to make left Y-axis visible. The checkbox may be disabled if unselected either in Data set or in Overrides tab.

- **Right Y**  
  Mark this checkbox to make right Y-axis visible. The checkbox may be disabled if unselected either in Data set or in Overrides tab.

- **X-Axis**  
  Unmark this checkbox to hide X-axis (marked by default).

- **Min**  
  Set the minimum value of the corresponding axis. Visible range minimum value of Y-axis is specified.

- **Max**  
  Set the maximum value of the corresponding axis. Visible range maximum value of Y-axis is specified.

- **Units**  
  Choose the unit for the graph axis values from the dropdown. If the Auto option is chosen axis values are displayed using units of the first item of the corresponding axis. Static option allows you to assign the corresponding axis’ custom name. If the Static option is chosen and the value input field left blank the corresponding axis’ name will only consist of a numeric value.

The **Legend** tab allows to customize the graph legend:

<table>
<thead>
<tr>
<th>Data set 2</th>
<th>Displaying options</th>
<th>Time period</th>
<th>Axes</th>
<th>Legend</th>
<th>Problems</th>
<th>Overrides</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Show legend</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display min/max/avg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Number of rows</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Number of columns</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Show legend**  
  Unmark this checkbox to hide the legend on the graph (marked by default).

- **Display min/max/avg**  
  Mark this checkbox to display the minimum, maximum and average values of the item in the legend.

- **Number of rows**  
  Set the number of legend rows to be displayed.

- **Number of columns**  
  Set the number of legend columns to be displayed.

The **Problems** tab allows to customize the problem display:
Show problems

Mark this checkbox to enable problem displaying on the graph (unmarked, i.e. disabled by default).

Selected items only

Mark this checkbox to include problems for the selected items only to be displayed on the graph.

Problem hosts

Select the problem hosts to be displayed on the graph. Wildcard patterns may be used (for example, * will return results that match zero or more characters). To specify a wildcard pattern, just enter the string manually and press Enter. While you are typing, note how all matching hosts are displayed in the dropdown.

Severity

Mark the problem severities to be displayed on the graph.

Problem

Specify the problem’s name to be displayed on the graph.

Tags

Specify problem tags to limit the number of problems displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive.

There are several operators available for each condition:

- **Exists** - include the specified tag names
- **Equals** - include the specified tag names and values (case-sensitive)
- **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)
- **Does not exist** - exclude the specified tag names
- **Does not equal** - exclude the specified tag names and values (case-sensitive)
- **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)

There are two calculation types for conditions:

- **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition
- **Or** - enough if one condition is met

The **Overrides** tab allows to add custom overrides for data sets:
Overrides are useful when several items are selected for a data set using the * wildcard and you want to change how the items are displayed by default (e.g. default base color or any other property).

Existing overrides (if any) are displayed in a list. To add a new override:

- Click on the **Add new override** button
- Select hosts and items for the override. Alternatively, you may enter host and item patterns. Wildcard patterns may be used (for example, * will return results that match zero or more characters). To specify a wildcard pattern, just enter the string manually and press Enter. While you are typing, note how all matching hosts are displayed in the dropdown. The wildcard symbol is always interpreted, therefore it is not possible to add, for example, an item named "item*" individually if there are other matching items (e.g. item2, item3). Host pattern and item pattern fields are mandatory.
- Click on **+** to select override parameters. At least one override parameter should be selected. For parameter descriptions, see the Data set tab above.

Information displayed by the graph widget can be downloaded as a .png image using the **widget menu**:

A screenshot of the widget will be saved to the Downloads folder.

9 Graph (classic)
Overview

In the classic graph widget, you can display a single custom graph or simple graph.

Configuration

To configure, select Graph (classic) as type:

In addition to the parameters that are common for all widgets, you may set the following specific options:

<table>
<thead>
<tr>
<th>Source</th>
<th>Select graph type;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph</strong></td>
<td>custom graph</td>
</tr>
<tr>
<td><strong>Simple graph</strong></td>
<td>simple graph</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Graph</th>
<th>Select the custom graph to display.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Select the item to display in a simple graph.</td>
</tr>
<tr>
<td>Show legend</td>
<td>Unmark this checkbox to hide the legend on the graph (marked by default).</td>
</tr>
<tr>
<td>Dynamic item</td>
<td>Set graph to display different data depending on the selected host.</td>
</tr>
</tbody>
</table>

Information displayed by the classic graph widget can be downloaded as .png image using the widget menu:
A screenshot of the widget will be saved to the Downloads folder.

10 Graph prototype

Overview

In the graph prototype widget, you can display a grid of graphs created from either a graph prototype or an item prototype by low-level discovery.

Configuration

To configure, select Graph prototype as widget type:
In addition to the parameters that are common for all widgets, you may set the following specific options:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Select source: either a <strong>Graph prototype</strong> or a <strong>Simple graph prototype</strong>.</td>
</tr>
<tr>
<td><strong>Graph prototype</strong></td>
<td>Select a graph prototype to display discovered graphs of the graph prototype.</td>
</tr>
<tr>
<td></td>
<td>This option is available if ‘Graph prototype’ is selected as Source.</td>
</tr>
<tr>
<td><strong>Item prototype</strong></td>
<td>Select an item prototype to display simple graphs based on discovered items of an item prototype.</td>
</tr>
<tr>
<td></td>
<td>This option is available if ‘Simple graph prototype’ is selected as Source.</td>
</tr>
<tr>
<td><strong>Show legend</strong></td>
<td>Mark this checkbox to show the legend on the graphs (marked by default).</td>
</tr>
<tr>
<td><strong>Dynamic item</strong></td>
<td>Set graphs to display different data depending on the selected host.</td>
</tr>
<tr>
<td><strong>Columns</strong></td>
<td>Enter the number of columns of graphs to display within a graph prototype widget.</td>
</tr>
<tr>
<td><strong>Rows</strong></td>
<td>Enter the number of rows of graphs to display within a graph prototype widget.</td>
</tr>
</tbody>
</table>

While the Columns and Rows settings allow fitting more than one graph in the widget, there still may be more discovered graphs than there are columns/rows in the widget. In this case paging becomes available in the widget and a slide-up header allows to switch between pages using the left and right arrows.

11 Host availability

Overview

In the host availability widget, high-level statistics about host availability are displayed in four colored columns/lines.
Host availability in each column/line is counted as follows:

- **Available** - hosts with all interfaces available
- **Not available** - hosts with at least one interface unavailable
- **Unknown** - hosts with at least one interface unknown (none unavailable)
- **Total** - total of all hosts

**Configuration**

To configure, select Host availability as type:

![Add widget](image)

In addition to the parameters that are **common** for all widgets, you may set the following specific options:
Host groups
Select host group(s). This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Scroll down to select. Click on ‘x’ to remove the selected.

Interface type
Select which host interfaces you want to see availability data for.
Availability of all interfaces is displayed by default if nothing is selected.

Layout
Select horizontal display (columns) or vertical display (lines).

Show hosts in maintenance
Include hosts that are in maintenance in the statistics.

12 Item value

Overview
This widget is useful for displaying the value of a single item prominently.

```
2022-01-05 22:43:15

2.38

Load average (5m avg)
```

Besides the value itself, additional elements can be displayed, if desired:

- time of the metric
- item description
- change indicator for the value
- item unit

The widget can display numeric and string values. String values are displayed on a single line and truncated, if needed. “No data” is displayed, if there is no value for the item.

Clicking on the value leads to an ad-hoc graph for numeric items or latest data for string items.

The widget and all elements in it can be visually fine-tuned using advanced configuration options, allowing to create a wide variety of visual styles:
Configuration

To configure, select item value as the widget type:
In addition to the parameters that are common for all widgets, you may set the following specific options:

<table>
<thead>
<tr>
<th><strong>Item</strong></th>
<th>Select the item.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Show</strong></td>
<td>Mark the checkbox to display the respective element (description, value, time, change indicator). Unmark to hide. At least one element must be selected.</td>
</tr>
<tr>
<td><strong>Advanced configuration</strong></td>
<td>Mark the checkbox to display advanced configuration options.</td>
</tr>
<tr>
<td><strong>Dynamic item</strong></td>
<td>Mark the checkbox to display a different value depending on the selected host.</td>
</tr>
</tbody>
</table>

**Advanced configuration**

Advanced configuration options become available if the Advanced configuration checkbox is marked (see screenshot) and only for those elements that are selected in the Show field (see above).

Additionally, advanced configuration allows to change the background color for the whole widget.
### Description

Enter the item description. This description may override the default item name. Multiline descriptions are supported. A combination of text and supported macros is possible. \{(HOST.*), \{ITEM.*\}, \{INVENTORY.*\} and user macros are supported.

- **Horizontal position**: Select horizontal position of the item description - left, right or center.
- **Vertical position**: Select vertical position of the item description - top, bottom or middle.
- **Size**: Enter font size height for the item description (in percent relative to total widget height).
- **Bold**: Mark the checkbox to display item description in bold type.
- **Color**: Select the item description color from the color picker. *D* stands for default color (depends on the frontend theme). To return to the default value, click the Use default button in the color picker.

### Value

- **Decimal places**: Select how many decimal places will be displayed with the value. This value will affect only float items.
- **Size**: Enter font size height for the decimal places (in percent relative to total widget height).
- **Horizontal position**: Select horizontal position of the item value - left, right or center.
- **Vertical position**: Select vertical position of the item value - top, bottom or middle.

---

715
Size
Enter font size height for the item value (in percent relative to total widget height).
Note that the size of item value is prioritised; other elements have to concede space for the value. With the change indicator though, if the value is too large, it will be truncated to show the change indicator.

Bold
Mark the checkbox to display item value in bold type.

Color
Select the item value color from the color picker.
D stands for default color (depends on the frontend theme). To return to the default value, click the Use default button in the color picker.

Units
Mark the checkbox to display units with the item value. If you enter a unit name, it will override the unit from item configuration.

Position
Select the item unit position - above, below, before or after the value.

Size
Enter font size height for the item unit (in percent relative to total widget height).

Bold
Mark the checkbox to display item unit in bold type.

Color
Select the item unit color from the color picker.
D stands for default color (depends on the frontend theme). To return to the default value, click the Use default button in the color picker.

Time
Time is the clock value from item history.

Horizontal position
Select horizontal position of the time - left, right or center.

Vertical position
Select vertical position of the time - top, bottom or middle.

Size
Enter font size height for the time (in percent relative to total widget height).

Bold
Mark the checkbox to display time in bold type.

Color
Select the time color from the color picker.
D stands for default color (depends on the frontend theme). To return to the default value, click the Use default button in the color picker.

Change indicator
Select the color of change indicators from the color picker. The change indicators are as follows:
↑ - item value is up (for numeric items)
↓ - item value is down (for numeric items)
↕ - item value has changed (for string items and items with value mapping)
D stands for default color (depends on the frontend theme). To return to the default value, click the Use default button in the color picker.

Vertical size of the change indicator is equal to the size of the value (integer part of the value for numeric items).
Note that up and down indicators are not shown with just one value.

Background color
Select the background color for the whole widget from the color picker.
D stands for default color (depends on the frontend theme). To return to the default value, click the Use default button in the color picker.

Note that multiple elements cannot occupy the same space; if they are placed in the same space, an error message will be displayed.

13 Map

Overview
In the map widget you can display either:

• a single configured network map
• one of the configured network maps in the map navigation tree (when clicking on the map name in the tree).

Configuration
To configure, select Map as type:
In addition to the parameters that are common for all widgets, you may set the following specific options:

<table>
<thead>
<tr>
<th>Source type</th>
<th>Select to display:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>- network map</td>
</tr>
<tr>
<td>Map navigation tree</td>
<td>- one of the maps in the selected map navigation tree</td>
</tr>
</tbody>
</table>

Map
Select the map to display.
This option is available if ‘Map’ is selected as Source type.

Filter
Select the map navigation tree to display the maps of.
This option is available if ‘Map navigation tree’ is selected as Source type.

See also: [known issue with IE11](#)

14 Map navigation tree

Overview
This widget allows building a hierarchy of existing maps while also displaying problem statistics with each included map and map group.

It becomes even more powerful if you link the Map widget to the navigation tree. In this case, clicking on a map name in the navigation tree displays the map in full in the Map widget.

Statistics with the top-level map in the hierarchy display a sum of problems of all sub-maps and their own problems.

Configuration
To configure the navigation tree widget, select Map navigation tree as type:
In addition to the parameters that are common for all widgets, you may set the following specific options:

| Show unavailable maps | Mark this checkbox to display maps that the user does not have read permission to. Available maps in the navigation tree will be displayed with a grayed-out icon. Note that if this checkbox is marked, available sub-maps are displayed even if the parent level map is unavailable. If unmarked, available sub-maps to an unavailable parent map will not be displayed at all. Problem count is calculated based on available maps and available map elements. |

15 Plain text

Overview

In the plain text widget, you can display the latest item data in plain text.

Configuration

To configure, select Plain text as type:
In addition to the parameters that are common for all widgets, you may set the following specific options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Select the items.</td>
</tr>
<tr>
<td>Items location</td>
<td>Choose the location of selected items to be displayed in the widget.</td>
</tr>
<tr>
<td>Show lines</td>
<td>Set how many latest data lines will be displayed in the widget.</td>
</tr>
<tr>
<td>Show text as HTML</td>
<td>Set to display text as HTML.</td>
</tr>
<tr>
<td>Dynamic item</td>
<td>Set to display different data depending on the selected host.</td>
</tr>
</tbody>
</table>

16 Problem hosts

Overview

In the problem host widget, you can display high-level information about host availability.

Configuration

To configure, select Problem hosts as type:
In addition to the parameters that are common for all widgets, you may set the following specific options:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host groups</td>
<td>Enter host groups to display in the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specify a parent host group implicitly selects all nested host groups. Host data from these host groups will be displayed in the widget. If no host groups are entered, all host groups will be displayed.</td>
</tr>
<tr>
<td>Exclude host groups</td>
<td>Enter host groups to hide from the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specify a parent host group implicitly selects all nested host groups. Host data from these host groups will not be displayed in the widget. For example, hosts 001, 002, 003 may be in Group A and hosts 002, 003 in Group B as well. If we select to show Group A and exclude Group B at the same time, only data from host 001 will be displayed in the Dashboard.</td>
</tr>
<tr>
<td>Hosts</td>
<td>Enter hosts to display in the widget. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. If no hosts are entered, all hosts will be displayed.</td>
</tr>
<tr>
<td>Problem</td>
<td>You can limit the number of problem hosts displayed by the problem name. If you enter a string here, only those hosts with problems whose name contains the entered string will be displayed. Macros are not expanded.</td>
</tr>
<tr>
<td>Severity</td>
<td>Mark the problem severities to be displayed in the widget.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| **Tags**                    | Specify problem tags to limit the number of problems displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition:  
  **Exists** - include the specified tag names  
  **Equals** - include the specified tag names and values (case-sensitive)  
  **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)  
  **Does not exist** - exclude the specified tag names  
  **Does not equal** - exclude the specified tag names and values (case-sensitive)  
  **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)  
  There are two calculation types for conditions:  
  **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition  
  **Or** - enough if one condition is met |
| **Show suppressed problems**| Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance.                                 |
| **Hide groups without problems** | Mark the Hide groups without problems option to hide data from host groups without problems in the widget.                                      |
| **Problem display**         | Display problem count as:  
  **All** - full problem count will be displayed  
  **Separated** - unacknowledged problem count will be displayed separated as a number of the total problem count  
  **Unacknowledged only** - only the unacknowledged problem count will be displayed. |

17 Problems

**Overview**

In this widget you can display current problems. The information in this widget is similar to Monitoring → Problems.

**Configuration**

To configure, select Problems as type:
You can limit how many problems are displayed in the widget in various ways - by problem status, problem name, severity, host group, host, event tag, acknowledgment status, etc.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show</td>
<td>Filter by problem status:</td>
</tr>
<tr>
<td></td>
<td><strong>Recent problems</strong> - unresolved and recently resolved problems are displayed (default)</td>
</tr>
<tr>
<td></td>
<td><strong>Problems</strong> - unresolved problems are displayed</td>
</tr>
<tr>
<td></td>
<td><strong>History</strong> - history of all events is displayed</td>
</tr>
<tr>
<td>Host groups</td>
<td>Enter host groups to display problems of in the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Problems from these host groups will be displayed in the widget. If no host groups are entered, problems from all host groups will be displayed.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Exclude host groups</td>
<td>Enter host groups to hide problems of from the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Problems from these host groups will not be displayed in the widget. For example, hosts 001, 002, 003 may be in Group A and hosts 002, 003 in Group B as well. If we select to show Group A and exclude Group B at the same time, only problems from host 001 will be displayed in the widget.</td>
</tr>
<tr>
<td>Hosts</td>
<td>Enter hosts to display problems of in the widget. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. If no hosts are entered, problems of all hosts will be displayed.</td>
</tr>
<tr>
<td>Problem</td>
<td>You can limit the number of problems displayed by their name. If you enter a string here, only those problems whose name contains the entered string will be displayed. Macros are not expanded.</td>
</tr>
<tr>
<td>Severity</td>
<td>Mark the problem severities to be displayed in the widget.</td>
</tr>
</tbody>
</table>
| Tags                  | Specify problem tags to limit the number of problems displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition:  
  **Exists** - include the specified tag names  
  **Equals** - include the specified tag names and values (case-sensitive)  
  **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)  
  **Does not exist** - exclude the specified tag names  
  **Does not equal** - exclude the specified tag names and values (case-sensitive)  
  **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)  
  There are two calculation types for conditions:  
  **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition  
  **Or** - enough if one condition is met  
  When filtered, the tags specified here will be displayed first with the problem, unless overridden by the Tag display priority (see below) list.  
  Select the number of displayed tags:  
  **None** - no Tags column in Monitoring → Problems  
  **1** - Tags column contains one tag  
  **2** - Tags column contains two tags  
  **3** - Tags column contains three tags  
  To see all tags for the problem roll your mouse over the three dots icon.  
  **Tag name display mode**:  
  **Full** - tag names and values are displayed in full  
  **Shortened** - tag names are shortened to 3 symbols; tag values are displayed in full  
  **None** - only tag values are displayed; no names  
  Enter tag display priority for a problem, as a comma-separated list of tags (for example: Services, Applications, Application). Tag names only should be used, no values. The tags of this list will always be displayed first, overriding the natural ordering by alphabet.  
  **Show operational data**:  
  **None** - no operational data is displayed  
  **Separately** - operational data is displayed in a separate column  
  **With problem name** - append operational data to the problem name, using parentheses for the operational data  
  **Show suppressed problems** - Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance or single problem suppression.  
  **Show unacknowledged only** - Mark the checkbox to display unacknowledged problems only.  
  **Sort entries by**:  
  **Time** (descending or ascending)  
  **Severity** (descending or ascending)  
  **Problem name** (descending or ascending)  
  **Host** (descending or ascending).  
  **Show timeline** - Mark the checkbox to display a visual timeline.  
  **Show lines** - Specify the number of problem lines to display. |
Overview
In this widget, you can display problems by severity. You can limit what hosts and triggers are displayed in the widget and define how the problem count is displayed.

Configuration
To configure, select Problems by severity as type:

In addition to the parameters that are common for all widgets, you may set the following specific options:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host groups</td>
<td>Enter host groups to display in the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will be displayed in the widget. If no host groups are entered, all host groups will be displayed.</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exclude host groups</strong></td>
<td>Enter host groups to hide from the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will not be displayed in the widget. For example, hosts 001, 002, 003 may be in Group A and hosts 002, 003 in Group B as well. If we select to show Group A and exclude Group B at the same time, only data from host 001 will be displayed in the Dashboard.</td>
</tr>
<tr>
<td><strong>Hosts</strong></td>
<td>Enter hosts to display in the widget. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. If no hosts are entered, all hosts will be displayed.</td>
</tr>
<tr>
<td><strong>Problem</strong></td>
<td>You can limit the number of problem hosts displayed by the problem name. If you enter a string here, only those hosts with problems whose name contains the entered string will be displayed. Macros are not expanded.</td>
</tr>
<tr>
<td><strong>Severity</strong></td>
<td>Mark the problem severities to be displayed in the widget.</td>
</tr>
<tr>
<td><strong>Tags</strong></td>
<td>Specify problem tags to limit the number of problems displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Exists</strong> - include the specified tag names</td>
</tr>
<tr>
<td></td>
<td>- <strong>Equals</strong> - include the specified tag names and values (case-sensitive)</td>
</tr>
<tr>
<td></td>
<td>- <strong>Contains</strong> - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)</td>
</tr>
<tr>
<td></td>
<td>- <strong>Does not exist</strong> - exclude the specified tag names</td>
</tr>
<tr>
<td></td>
<td>- <strong>Does not equal</strong> - exclude the specified tag names and values (case-sensitive)</td>
</tr>
<tr>
<td></td>
<td>- <strong>Does not contain</strong> - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)</td>
</tr>
<tr>
<td></td>
<td>There are two calculation types for conditions:</td>
</tr>
<tr>
<td></td>
<td>- <strong>And/Or</strong> - all conditions must be met, conditions having the same tag name will be grouped by the Or condition</td>
</tr>
<tr>
<td></td>
<td>- <strong>Or</strong> - enough if one condition is met</td>
</tr>
<tr>
<td><strong>Show</strong></td>
<td>Select the show option:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Host groups</strong> - display problems per host group</td>
</tr>
<tr>
<td></td>
<td>- <strong>Totals</strong> - display a problem total for all selected host groups in colored blocks corresponding to the problem severity.</td>
</tr>
<tr>
<td><strong>Layout</strong></td>
<td>Select the layout option:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Horizontal</strong> - colored blocks of totals will be displayed horizontally</td>
</tr>
<tr>
<td></td>
<td>- <strong>Vertical</strong> - colored blocks of totals will be displayed vertically</td>
</tr>
<tr>
<td></td>
<td>This field is available for editing if &quot;Totals&quot; is selected as the Show option.</td>
</tr>
<tr>
<td><strong>Show suppressed problems</strong></td>
<td>Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance.</td>
</tr>
<tr>
<td><strong>Hide groups without problems</strong></td>
<td>Mark the Hide groups without problems option to hide data from host groups without problems in the widget.</td>
</tr>
<tr>
<td><strong>Show operational data</strong></td>
<td>Mark the checkbox to display operational data (see description of Operational data in Monitoring → Problems).</td>
</tr>
<tr>
<td><strong>Problem display</strong></td>
<td>Display problem count as:</td>
</tr>
<tr>
<td></td>
<td>- <strong>All</strong> - full problem count will be displayed</td>
</tr>
<tr>
<td></td>
<td>- <strong>Separated</strong> - unacknowledged problem count will be displayed separated as a number of the total problem count</td>
</tr>
<tr>
<td></td>
<td>- <strong>Unacknowledged only</strong> - only the unacknowledged problem count will be displayed.</td>
</tr>
<tr>
<td><strong>Show timeline</strong></td>
<td>Mark the checkbox to display a visual timeline.</td>
</tr>
</tbody>
</table>

### Overview

This widget is useful for displaying SLA reports. Functionally it is similar to the Services -> SLA report section.

### Configuration

To configure, select SLA report as type:
In addition to the parameters that are common for all widgets, you may set the following specific options:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLA</td>
<td>Select the SLA for the report.</td>
</tr>
<tr>
<td>Service</td>
<td>Select the service for the report.</td>
</tr>
<tr>
<td>Show periods</td>
<td>Set how many periods will be displayed in the widget (20 by default, 100 maximum).</td>
</tr>
<tr>
<td>From</td>
<td>Select the beginning date for the report.</td>
</tr>
<tr>
<td>To</td>
<td>Select the end date for the report.</td>
</tr>
</tbody>
</table>

Relative dates are supported: `now`, `now/d`, `now/w−1w` etc; supported date modifiers: `d, w, M, y`.

20 System information

Overview

This widget displays the same information as in Reports → System information, however, a single dashboard widget can only display either the system stats or the high availability nodes at a time (not both).

Configuration

To configure, select System information as type:
All configuration parameters are common for all widgets.

21 Top hosts

Overview

This widget provides a way to create custom tables for displaying the data situation, allowing to display Top N-like reports and progress-bar reports useful for capacity planning.

<table>
<thead>
<tr>
<th>Top 3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>CPU</td>
</tr>
<tr>
<td>Server3</td>
<td>3.16</td>
</tr>
<tr>
<td>Zabbix server</td>
<td>3.11</td>
</tr>
<tr>
<td>New host</td>
<td>3.1</td>
</tr>
</tbody>
</table>

**Key statistics**

<table>
<thead>
<tr>
<th>Name</th>
<th>Space utilization</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zabbix server</td>
<td>95.8489 %</td>
<td>1.77</td>
</tr>
</tbody>
</table>

Configuration

To configure, select Top hosts as type:
In addition to the parameters that are common for all widgets, you may set the following specific options:

**Host groups**
Host groups to display data for.

**Hosts**
Hosts to display data for.

**Host tags**
Specify tags to limit the number of hosts displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive.

There are several operators available for each condition:
- **Exists** - include the specified tag names
- **Equals** - include the specified tag names and values (case-sensitive)
- **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)
- **Does not exist** - exclude the specified tag names
- **Does not equal** - exclude the specified tag names and values (case-sensitive)
- **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)

There are two calculation types for conditions:
- **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition
- **Or** - enough if one condition is met

**Columns**
Add data columns to display.
The column order determines their display from left to right.
Columns can be reordered by dragging up and down by the handle before the column name.

**Order**
Specify the ordering of rows:
- **Top N** - in descending order by the Order column aggregated value
- **Bottom N** - in ascending order by the Order column aggregated value

**Order column**
Specify the column from the defined Columns list to use for Top N or Bottom N ordering.

**Host count**
Count of host rows to be shown.

Column configuration
Common column parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Name of the column.</td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td>Data type to display in the column:</td>
</tr>
<tr>
<td>* Item</td>
<td>Select the item.</td>
</tr>
<tr>
<td><strong>Time shift</strong></td>
<td>Specify time shift if required. You may use time suffixes in this field.</td>
</tr>
<tr>
<td><strong>Aggregation function</strong></td>
<td>Specify which aggregation function to use:</td>
</tr>
<tr>
<td><strong>Display</strong></td>
<td>Define how the value should be displayed:</td>
</tr>
<tr>
<td><strong>History data</strong></td>
<td>Specify the interval for aggregating values. You may use time suffixes in this field. A numeric value without a suffix will be regarded as seconds. This field will not be displayed if Aggregation function is &quot;none&quot;.</td>
</tr>
<tr>
<td><strong>Base color</strong></td>
<td>Background color of the column; fill color if Item value data is displayed as bar/indicators. For Item value data the default color can be overridden by custom color, if the item value is over one of the specified &quot;Thresholds&quot;.</td>
</tr>
<tr>
<td><strong>Thresholds</strong></td>
<td></td>
</tr>
</tbody>
</table>

Specific parameters for item value columns:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item</strong></td>
<td>Select the item.</td>
</tr>
<tr>
<td><strong>Time shift</strong></td>
<td>Specify time shift if required. You may use time suffixes in this field.</td>
</tr>
<tr>
<td><strong>Aggregation function</strong></td>
<td>Specify which aggregation function to use:</td>
</tr>
<tr>
<td><strong>Display</strong></td>
<td>Define how the value should be displayed:</td>
</tr>
<tr>
<td><strong>History data</strong></td>
<td>Specify the interval for aggregating values. You may use time suffixes in this field. A numeric value without a suffix will be regarded as seconds. This field will not be displayed if Aggregation function is &quot;none&quot;.</td>
</tr>
<tr>
<td><strong>Base color</strong></td>
<td>Background color of the column; fill color if Item value data is displayed as bar/indicators. For Item value data the default color can be overridden by custom color, if the item value is over one of the specified &quot;Thresholds&quot;.</td>
</tr>
</tbody>
</table>

**Item value** - value of the specified item  
**Host name** - host name of the item specified in the Item value column  
**Text** - static text string
History settings:

- **History** - take history data
- **Trends** - take trend data

This setting applies only to numeric data. Non-numeric data will always be taken from history.

**Min**
- Minimum value for bar/indicators.

**Max**
- Maximum value for bar/indicators.

**Thresholds**
- Specify threshold values when the background/fill color should change. The list will be sorted in ascending order when saved.

Note that only numeric items can be displayed in this column if thresholds are used.

---

Specific parameters for text columns:

<table>
<thead>
<tr>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enter the string to display. May contain host and inventory macros.</strong></td>
</tr>
</tbody>
</table>

---

**22 Trigger overview**

**Overview**

In the trigger overview widget, you can display the trigger states for a group of hosts.

- The trigger states are displayed as colored blocks (the color of problem triggers depends on the problem severity color, which can be adjusted in the problem update screen). Note that recent trigger changes (within the last 2 minutes) will be displayed as blinking blocks.
- Blue up and down arrows indicate triggers that have dependencies. On mouseover, dependency details are revealed.
- A checkbox icon indicates acknowledged problems. All problems or resolved problems of the trigger must be acknowledged for this icon to be displayed.

Clicking on a trigger block provides context-dependent links to problem events of the trigger, the problem acknowledgment screen, trigger configuration, trigger URL or a simple graph/latest values list.

Note that 50 records are displayed by default (configurable in Administration → General → GUI, using the Max number of columns and rows in overview tables option). If more records exist than are configured to display, a message is displayed at the bottom of the table, asking to provide more specific filtering criteria. There is no pagination.

**Configuration**

To configure, select Trigger overview as type:
In addition to the parameters that are common for all widgets, you may set the following specific options:

**Show**
- Filter by problem status:
  - **Recent problems** - unresolved and recently resolved problems are displayed (default)
  - **Problems** - unresolved problems are displayed
  - **Any** - history of all events is displayed

**Host groups**
- Select the host group(s). This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups.

**Hosts**
- Select hosts. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. Scroll down to select. Click on ‘x’ to remove the selected.

**Tags**
- Specify tags to limit the number of item and trigger data displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive.
- There are several operators available for each condition:
  - **Exists** - include the specified tag names
  - **Equals** - include the specified tag names and values (case-sensitive)
  - **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)
  - **Does not exist** - exclude the specified tag names
  - **Does not equal** - exclude the specified tag names and values (case-sensitive)
  - **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)
- There are two calculation types for conditions:
  - **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition
  - **Or** - enough if one condition is met

**Show suppressed problems**
- Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance.

**Hosts location**
- Select host location - left or top.
Overview

This widget displays the content retrieved from the specified URL.

Configuration

To configure, select URL as type:

In addition to the parameters that are common for all widgets, you may set the following specific options:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>Enter the URL to display. Relative paths are allowed since Zabbix 4.4.8.</td>
</tr>
<tr>
<td></td>
<td>(HOST.*) macros are supported.</td>
</tr>
<tr>
<td>Dynamic item</td>
<td>Set to display different URL content depending on the selected host.</td>
</tr>
<tr>
<td></td>
<td>This can work if {HOST.*) macros are used in the URL.</td>
</tr>
</tbody>
</table>

Browsers might not load an HTTP page included in the widget if Zabbix frontend is accessed over HTTPS.

24 Web monitoring

Overview

This widget displays a status summary of the active web monitoring scenarios.

Configuration
In cases when a user does not have permission to access certain widget elements, that element’s name will appear as Inaccessible during the widget’s configuration. This results in Inaccessible Item, Inaccessible Host, Inaccessible Group, Inaccessible Map, and Inaccessible Graph appearing instead of the "real" name of the element.

In addition to the parameters that are common for all widgets, you may set the following specific options:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host groups</td>
<td>Enter host groups to display in the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will be displayed in the widget. If no host groups are entered, all host groups will be displayed.</td>
</tr>
<tr>
<td>Exclude host groups</td>
<td>Enter host groups to hide from the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will not be displayed in the widget. For example, hosts 001, 002, 003 may be in Group A and hosts 002, 003 in Group B as well. If we select to show Group A and exclude Group B at the same time, only data from host 001 will be displayed in the Dashboard.</td>
</tr>
<tr>
<td>Hosts</td>
<td>Enter hosts to display in the widget. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. If no hosts are entered, all hosts will be displayed.</td>
</tr>
</tbody>
</table>
Parameter | Description
---|---
Tags | Specify tags to limit the number of web scenarios displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition:
- **Exists** - include the specified tag names
- **Equals** - include the specified tag names and values (case-sensitive)
- **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)
- **Does not exist** - exclude the specified tag names
- **Does not equal** - exclude the specified tag names and values (case-sensitive)
- **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)
There are two calculation types for conditions:
- **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition
- **Or** - enough if one condition is met

Show hosts in maintenance | Include hosts that are in maintenance in the statistics.

## 2 Problems

### Overview

In Monitoring → Problems you can see what problems you currently have. Problems are those triggers that are in the “Problem” state.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Problem start time is displayed.</td>
</tr>
<tr>
<td>Severity</td>
<td>Problem severity is displayed. Problem severity is originally based on the severity of the underlying problem trigger, however, after the event has happened it can be updated using the Update problem screen. Color of the problem severity is used as cell background during problem time.</td>
</tr>
<tr>
<td>Recovery time</td>
<td>Problem resolution time is displayed.</td>
</tr>
</tbody>
</table>
| Status | Problem status is displayed:  
- **Problem** - unresolved problem  
- **Resolved** - recently resolved problem. You can hide recently resolved problems using the filter. New and recently resolved problems blink for 2 minutes. Resolved problems are displayed for 5 minutes in total. Both of these values are configurable in Administration → General → Trigger displaying options. |
<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info</td>
<td>A green information icon is displayed if a problem is closed by global correlation or manually when updating the problem. Rolling a mouse over the icon will display more details:</td>
</tr>
<tr>
<td></td>
<td><img src="Image1" alt="Zabbix server" /> Resolved by user &quot;Admin (Zabbix Administrator)&quot;. The following icon is displayed if a suppressed problem is being shown (see Show suppressed problems option in the filter). Rolling a mouse over the icon will display more details:</td>
</tr>
<tr>
<td>Host</td>
<td>Problem host is displayed.</td>
</tr>
<tr>
<td>Problem</td>
<td>Problem name is displayed. Problem name is based on the name of the underlying problem trigger. Macros in the trigger name are resolved at the time of the problem happening and the resolved values do not update any more. Note that it is possible to append the problem name with operational data showing some latest item values. Clicking on the problem name brings up the event menu. Hovering on the icon after the problem name will bring up the trigger description (for those problems that have it). (Note that resolved values of {ITEM.VALUE} and {ITEM.LASTVALUE} macros in trigger descriptions are truncated to 20 characters. To see the entire values you may use macro functions with these macros, e.g. {{ITEM.VALUE}.regsub(&quot;(.<em>)&quot;, \1)}, {{ITEM.LASTVALUE}.regsub(&quot;(.</em>)&quot;, \1)} as a workaround.)</td>
</tr>
<tr>
<td>Operational data</td>
<td>Operational data are displayed containing latest item values. If no operational data is configured on a trigger level, the latest values of all items from the expression are displayed. This column is only displayed if Separately is selected for Show operational data in the filter.</td>
</tr>
<tr>
<td>Duration</td>
<td>Problem duration is displayed. See also: Negative problem duration</td>
</tr>
<tr>
<td>Ack</td>
<td>The acknowledgment status of the problem is displayed: Yes - green text indicating that the problem is acknowledged. A problem is considered to be acknowledged if all events for it are acknowledged. No - a red link indicating unacknowledged events. If you click on the link you will be taken to the problem update screen where various actions can be taken on the problem, including commenting and acknowledging the problem.</td>
</tr>
</tbody>
</table>
Column Description

**Actions**

History of activities about the problem is displayed using symbolic icons:

- comments have been made. The number of comments is also displayed.
- problem severity has been increased (e.g. Information → Warning)
- problem severity has been decreased (e.g. Warning → Information)
- problem severity has been changed, but returned to the original level (e.g. Warning → Information → Warning)
- actions have been taken. The number of actions is also displayed.
- actions have been taken, at least one is in progress. The number of actions is also displayed.
- actions have been taken, at least one has failed. The number of actions is also displayed.

When rolling the mouse over the icons, popups with details about the activity are displayed. See viewing details to learn more about icons used in the popup for actions taken.

**Tags**

Tags are displayed (if any).

In addition, tags from an external ticketing system may also be displayed (see the Process tags option when configuring webhooks).

Operational data of problems

It is possible to display operational data for current problems, i.e. the latest item values as opposed to the item values at the time of the problem.

Operational data display can be configured in the filter of Monitoring → Problems or in the configuration of the respective dashboard widget, by selecting one of the three options:

- None - no operational data is displayed
- Separately - operational data is displayed in a separate column
- With problem name - operational data is appended to the problem name and in parentheses. Operational data are appended to the problem name only if the Operational data field is non-empty in the trigger configuration.

The content of operational data can be configured with each trigger, in the Operational data field. This field accepts an arbitrary string with macros, most importantly, the \{ITEM.LASTVALUE<1–9>\} macro.

\{ITEM.LASTVALUE<1–9>\} in this field will always resolve to the latest values of items in the trigger expression. \{ITEM.VALUE<1–9>\} in this field will resolve to the item values at the moment of trigger status change (i.e. change into problem, change into OK, being closed manually by a user or being closed by correlation).

Negative problem duration

It is actually possible in some common situations to have negative problem duration i.e. when the problem resolution time is earlier than problem creation time, e. g.:

- If some host is monitored by proxy and a network error happens, leading to no data received from the proxy for a while, the nodata/(host/key) trigger will be fired by the server. When the connection is restored, the server will receive item data from the proxy having a time from the past. Then, the nodata/(host/key) problem will be resolved and it will have a negative problem duration;
- When item data that resolve the problem event are sent by Zabbix sender and contain a timestamp earlier than the problem creation time, a negative problem duration will also be displayed.
Negative problem duration is not affecting SLA calculation or Availability report of a particular trigger in any way; it neither reduces nor expands problem time.

Mass editing options

Buttons below the list offer some mass-editing options:

- Mass update - update the selected problems by navigating to the problem update screen

To use this option, mark the checkboxes before the respective problems, then click on the Mass update button.

Buttons

The button to the right offers the following option:

Export to CSV

Export content from all pages to a CSV file.

View mode buttons, being common for all sections, are described on the Monitoring page.

Using filter

You can use the filter to display only the problems you are interested in. For better search performance, data is searched with macros unresolved.

The filter is located above the table. Favorite filter settings can be saved as tabs and then quickly accessed by clicking on the tabs above the filter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show</td>
<td>Filter by problem status:</td>
</tr>
<tr>
<td></td>
<td>Recent problems - unresolved and recently resolved problems are displayed (default)</td>
</tr>
<tr>
<td></td>
<td>Problems - unresolved problems are displayed</td>
</tr>
<tr>
<td></td>
<td>History - history of all events is displayed</td>
</tr>
<tr>
<td>Host groups</td>
<td>Filter by one or more host groups.</td>
</tr>
<tr>
<td></td>
<td>Specifying a parent host group implicitly selects all nested host groups.</td>
</tr>
<tr>
<td>Hosts</td>
<td>Filter by one or more hosts.</td>
</tr>
<tr>
<td>Triggers</td>
<td>Filter by one or more triggers.</td>
</tr>
<tr>
<td>Problem</td>
<td>Filter by problem name.</td>
</tr>
<tr>
<td>Severity</td>
<td>Filter by trigger (problem) severity.</td>
</tr>
<tr>
<td>Age less than</td>
<td>Filter by how old the problem is.</td>
</tr>
<tr>
<td>Host inventory</td>
<td>Filter by inventory type and value.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Tags               | Filter by event tag name and value. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition:  
  - **Exists** - include the specified tag names  
  - **Equals** - include the specified tag names and values (case-sensitive)  
  - **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)  
  - **Does not exist** - exclude the specified tag names  
  - **Does not equal** - exclude the specified tag names and values (case-sensitive)  
  - **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)  
There are two calculation types for conditions:  
  - **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition  
  - **Or** - enough if one condition is met  
When filtered, the tags specified here will be displayed first with the problem, unless overridden by the Tag display priority (see below) list. |
| Show tags          | Select the number of displayed tags:  
  - **None** - no Tags column in Monitoring → Problems  
  - **1** - Tags column contains one tag  
  - **2** - Tags column contains two tags  
  - **3** - Tags column contains three tags  
To see all tags for the problem roll your mouse over the three dots icon. |
| Tag name           | Select tag name display mode:  
  - **Full** - tag names and values are displayed in full  
  - **Shortened** - tag names are shortened to 3 symbols; tag values are displayed in full  
  - **None** - only tag values are displayed; no names |
| Tag display priority| Enter tag display priority for a problem, as a comma-separated list of tags (for example: Services,Applications,Application). Tag names only should be used, no values. The tags of this list will always be displayed first, overriding the natural ordering by alphabet. |
| Show operational data| Select the mode for displaying operational data:  
  - **None** - no operational data is displayed  
  - **Separately** - operational data is displayed in a separate column  
  - **With problem name** - append operational data to the problem name, using parentheses for the operational data |
| Show suppressed problems| Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance or single problem suppression. |
| Compact view       | Mark the checkbox to enable compact view. |
| Show details       | Mark the checkbox to display underlying trigger expressions of the problems. Disabled if Compact view checkbox is marked. |
| Show unacknowledged only| Mark the checkbox to display unacknowledged problems only. |
| Show timeline      | Mark the checkbox to display the visual timeline and grouping. Disabled if Compact view checkbox is marked. |
| Highlight whole row| Mark the checkbox to highlight the full line for unresolved problems. The problem severity color is used for highlighting.  
  Enabled only if the Compact view checkbox is marked in the standard blue and dark themes. Highlight whole row is not available in the high-contrast themes. |

**Tabs for favorite filters**

Frequently used sets of filter parameters can be saved in tabs.

To save a new set of filter parameters, open the main tab, and configure the filter settings, then press the Save as button. In a new popup window, define Filter properties.
**Filter properties**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the filter to display in the tab list.</td>
</tr>
<tr>
<td>Show number of records</td>
<td>Check, if you want the number of problems to be displayed next to the tab name.</td>
</tr>
<tr>
<td>Set custom time period</td>
<td>Check to set specific default time period for this filter set. If set, you will only be able to</td>
</tr>
<tr>
<td></td>
<td>change the time period for this tab by updating filter settings. For tabs without a custom time</td>
</tr>
<tr>
<td></td>
<td>period, the time range can be changed by pressing the time selector button in the top right</td>
</tr>
<tr>
<td></td>
<td>corner (button name depends on selected time interval: This week, Last 30 minutes, Yesterday,</td>
</tr>
<tr>
<td></td>
<td>etc.). This option is available only for filters in Monitoring→Problems.</td>
</tr>
<tr>
<td>From/To</td>
<td>Time period start and end in absolute (Y-m-d   H:i:s) or relative time syntax (now-1d). Available,</td>
</tr>
<tr>
<td></td>
<td>if Set custom time period is checked.</td>
</tr>
</tbody>
</table>

When saved, the filter is created as a named filter tab and immediately activated.

To edit the filter properties of an existing filter, press the gear symbol next to the active tab name.

![Filter properties](image)

Notes:

- To hide the filter area, click on the name of the current tab. Click on the active tab name again to open the filter area again.
- Keyboard navigation is supported: use arrows to switch between tabs, press Enter to open.
- The left/right buttons above the filter may be used to switch between saved filters. Alternatively, the downward pointing button opens a drop-down menu with all saved filters and you can click on the one you need.
- Filter tabs can be re-arranged by dragging and dropping.
- If the settings of a saved filter have been changed (but not saved), a green dot is displayed after the filter name. To update the filter according to the new settings, click on the Update button, which is displayed instead of the Save as button.
- Current filter settings are remembered in the user profile. When the user opens the page again, the filter settings will have stayed the same.

To share filters, copy and send to others a URL of an active filter. After opening this URL, other users will be able to save this set of parameters as a permanent filter in their Zabbix account.

See also: [Page parameters](#).

**Filter buttons**

- **Apply**: Apply specified filtering criteria (without saving).
- **Reset**: Reset current filter and return to saved parameters of the current tab. On the main tab, this will clear the filter.
- **Save as**: Save current filter parameters in a new tab. Only available on the main tab.
Event menu

Clicking on the problem name brings up the event menu:

The event menu allows to:

- filter the problems of the trigger
- access the trigger configuration
- access a simple graph/item history of the underlying item(s)
- access an external ticket of the problem (if configured, see the Include event menu entry option when configuring webhooks)
- execute global scripts (these scripts need to have their scope defined as ‘Manual event action’). This feature may be handy for running scripts used for managing problem tickets in external systems.

Viewing details

The times for problem start and recovery in Monitoring → Problems are links. Clicking on them opens more details of the event.
Note how the problem severity differs for the trigger and the problem event - for the problem event it has been updated using the Update problem screen.

In the action list, the following icons are used to denote the activity type:

- ![event](image) - problem event generated
- ![message](image) - message has been sent
- ![acknowledged](image) - problem event acknowledged
- ![unacknowledged](image) - problem event unacknowledged
- ![comment](image) - a comment has been added
- ![severity](image) - problem severity has been increased (e.g. Information → Warning)
- ![severity](image) - problem severity has been decreased (e.g. Warning → Information)
- ![severity](image) - problem severity has been changed, but returned to the original level (e.g. Warning → Information → Warning)
- ![command](image) - a remote command has been executed
- ![recovered](image) - problem event has recovered
- ![closed](image) - the problem has been closed manually
- ![suppressed](image) - the problem has been suppressed
- ![unsuppressed](image) - the problem has been unsuppressed

3 Hosts

Overview

The Monitoring → Hosts section displays a full list of monitored hosts with detailed information about host interface, availability, tags, current problems, status (enabled/disabled), and links to easily navigate to the host’s latest data, problem history, graphs, dashboards and web scenarios.
<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The visible host name. Clicking on the name brings up the host menu. An orange wrench icon after the name indicates that this host is in maintenance. Click on the column header to sort hosts by name in ascending or descending order.</td>
</tr>
<tr>
<td>Interface</td>
<td>The main interface of the host is displayed.</td>
</tr>
<tr>
<td>Availability</td>
<td>Host availability per configured interface is displayed. Icons represent only those interface types (Zabbix agent, SNMP, IPMI, JMX) that are configured. If you position the mouse on the icon, a popup list appears listing all interfaces of this type with details, status and errors (for the agent interface, availability of active checks is also listed). The column is empty for hosts with no interfaces. The current status of all interfaces of one type is displayed by the respective icon color: <strong>Green</strong> - all interfaces available <strong>Yellow</strong> - at least one interface available and at least one unavailable; others can have any value including ‘unknown’ <strong>Red</strong> - no interfaces available <strong>Gray</strong> - at least one interface unknown (none unavailable)</td>
</tr>
<tr>
<td>Tags</td>
<td>Tags of the host and all linked templates, with macros unresolved. Click on the column header to sort hosts by status in ascending or descending order.</td>
</tr>
<tr>
<td>Status</td>
<td>Host status - Enabled or Disabled. Click on the column header to sort hosts by status in ascending or descending order. The number of items with latest data is displayed in gray.</td>
</tr>
<tr>
<td>Latest data</td>
<td>Clicking on the link will open the Monitoring - Latest data page with all the latest data collected from the host. The number of open host problems sorted by severity. The color of the square indicates problem severity. The number on the square means the number of problems for the given severity. Clicking on the icon will open Monitoring - Problems page for the current host.</td>
</tr>
<tr>
<td>Problems</td>
<td>Clicking on the link will display graphs configured for the host. The number of graphs is displayed in gray. If a host has no graphs, the link is disabled (gray text) and no number is displayed.</td>
</tr>
<tr>
<td>Graphs</td>
<td>Clicking on the link will display graphs configured for the host. The number of graphs is displayed in gray. If a host has no graphs, the link is disabled (gray text) and no number is displayed.</td>
</tr>
<tr>
<td>Dashboards</td>
<td>If a host has no dashboards, the link is disabled (gray text) and no number is displayed. If a host has no dashboards, the link is disabled (gray text) and no number is displayed.</td>
</tr>
<tr>
<td>Web</td>
<td>Clicking on the link will display web scenarios configured for the host. The number of web scenarios is displayed in gray. If a host has no web scenarios, the link is disabled (gray text) and no number is displayed.</td>
</tr>
</tbody>
</table>

**Active check availability**

Since Zabbix 6.2 active checks also affect host availability, if there is at least one enabled active check on the host. To determine active check availability heartbeat messages are sent in the agent active check thread. The frequency of the heartbeat messages is set by the **HeartbeatFrequency** parameter in Zabbix agent and agent 2 configurations (60 seconds by default, 0-3600 range). Active checks are considered unavailable when the active check heartbeat is older than 2 x HeartbeatFrequency seconds. Note that if Zabbix agents older than 6.2.x are used, they are not sending any active check heartbeats, so the availability of their hosts will remain unknown. Active agent availability is counted towards the total Zabbix agent availability in the same way as a passive interface is (for example, if a passive interface is available, while the active checks are unknown, the total agent availability is set to gray(unknown)).

**Buttons**

Create host allows to create a new host. This button is available for Admin and Super Admin users only.

View mode buttons being common for all sections are described on the Monitoring page.

Using filter

You can use the filter to display only the hosts you are interested in. For better search performance, data is searched with macros unresolved.

The filter is located above the table. It is possible to filter hosts by name, host group, IP or DNS, interface port, tags, problem severity, status (enabled/disabled/any); you can also select whether to display suppressed problems and hosts that are currently
in maintenance.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Filter by visible host name.</td>
</tr>
<tr>
<td>Host groups</td>
<td>Filter by one or more host groups.</td>
</tr>
<tr>
<td>Specification</td>
<td>Specifying a parent host group implicitly selects all nested host groups.</td>
</tr>
<tr>
<td>IP</td>
<td>Filter by IP address.</td>
</tr>
<tr>
<td>DNS</td>
<td>Filter by DNS name.</td>
</tr>
<tr>
<td>Port</td>
<td>Filter by port number.</td>
</tr>
<tr>
<td>Severity</td>
<td>Filter by problem severity. By default problems of all severities are displayed. Problems are displayed if not suppressed.</td>
</tr>
<tr>
<td>Status</td>
<td>Filter by host status.</td>
</tr>
</tbody>
</table>
| Tags            | Filter by host tag name and value. Hosts can be filtered by host-level tags as well as tags from all linked templates, including parent templates. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition:

- **Exists** - include the specified tag names
- **Equals** - include the specified tag names and values (case-sensitive)
- **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)
- **Does not exist** - exclude the specified tag names
- **Does not equal** - exclude the specified tag names and values (case-sensitive)
- **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)

There are two calculation types for conditions:

- **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition
- **Or** - enough if one condition is met

- **Show hosts in maintenance**
  - Mark the checkbox to display hosts that are in maintenance (displayed by default).

- **Show suppressed problems**
  - Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance or single problem suppression.

**Saving filter**

Favorite filter settings can be saved as tabs and then quickly accessed by clicking on the respective tab above the filter.

See more details about saving filters.

**1 Graphs**

**Overview**

Host graphs can be accessed from Monitoring → Hosts by clicking on Graphs for the respective host.

Any custom graph that has been configured for the host can be displayed, as well as any simple graph.
Graphs are sorted by:

- graph name (custom graphs)
- item name (simple graphs)

Graphs for disabled hosts are also accessible.

Time period selector

Take note of the time period selector above the graph. It allows selecting often required periods with one mouse click.

See also: Time period selector

Using filter

To view a specific graph, select it in the filter. The filter allows to specify the host, the graph name and the Show option (all/host graphs/simple graphs).

If no host is selected in the filter, no graphs are displayed.

Using subfilter

The subfilter is useful for a quick one-click access to related graphs. The subfilter operates autonomously from the main filter - results are filtered immediately, no need to click on Apply in the main filter.

Note that the subfilter only allows to further modify the filtering from the main filter.

Unlike the main filter, the subfilter is updated together with each table refresh request to always get up-to-date information of available filtering options and their counter numbers.

The subfilter shows clickable links allowing to filter graphs based on a common entity - the tag name or tag value. As soon as the entity is clicked, graphs are immediately filtered; the selected entity is highlighted with gray background. To remove the filtering, click on the entity again. To add another entity to the filtered results, click on another entity.

The number of entities displayed is limited to 100 horizontally. If there are more, a three-dot icon is displayed at the end; it is not clickable. Vertical lists (such as tags with their values) are limited to 20 entries. If there are more, a three-dot icon is displayed; it is not clickable.
A number next to each clickable entity indicates the number of graphs it has in the results of the main filter.

Once one entity is selected, the numbers with other available entities are displayed with a plus sign indicating how many graphs may be added to the current selection.

Buttons

View mode buttons, being common for all sections, are described on the Monitoring page.

2 Web scenarios

Overview

Host web scenario information can be accessed from Monitoring → Hosts by clicking on Web for the respective host.

Data of disabled hosts is also accessible. The name of a disabled host is listed in red.

The maximum number of scenarios displayed per page depends on the Rows per page user profile setting.

By default, only values that fall within the last 24 hours are displayed. This limit has been introduced with the aim of improving initial loading times for large pages of latest data. You can extend this time period by changing the value of Max history display period parameter in the Administration→General menu section.

The scenario name is link to more detailed statistics about it:
Using filter

The page shows a list of all web scenarios of the selected host. To view web scenarios for another host or host group without returning to the Monitoring → Hosts page, select that host or group in the filter. You may also filter scenarios based on tags.

Buttons

View mode buttons being common for all sections are described on the Monitoring page.

4 Latest data

Overview

In this section you can view the latest values gathered by items.

Graphs are also available for the item values.
This section contains:

- the **filter** (collapsed by default)
- the **subfilter** (never collapsed)
- the **item list**

Items are displayed with their name, time since the last check, **last value**, change amount, tags, and a link to a simple graph/history of item values.

Values in the Last value column are displayed with unit conversion and value mapping applied. To view raw data, hover over the value.

Tags in the item list are clickable. If you click on a tag, this tag becomes enabled in the **subfilter**. The item list now displays the items corresponding to this tag and any other previously selected tags in the subfilter. Note that once the items have been filtered in this way, tags in the list are no longer clickable. Further modification based on tags (e.g. remove, add another filter) must be done in the subfilter.

If an item has errors, for example, has become unsupported, an information icon will be displayed in the Info column. Hover over the icon for details.

An icon with a question mark is displayed next to the item name for all items that have a description. Hover over this icon to see a tooltip with the item description.

If a host to which the item belongs is in maintenance, an orange wrench icon is displayed after the host’s name.

Note: The name of a disabled host is displayed in red. Data of disabled hosts, including graphs and item value lists, is also accessible in **Latest data**.

By default, only values that fall within the last 24 hours are displayed. This limit has been introduced with the aim of improving initial loading times for large pages of the latest data. This time period can be extended by changing the value of the Max history display period parameter in Administration → **General** (/manual/web_interface/frontend_sections/administration/general#gui).

For items with an update frequency of 1 day or more the change amount will never be displayed (with the default setting). Also in this case the last value will not be displayed at all if it was received more than 24 hours ago.

**Item menu**

Clicking on the item name opens the item menu with links to:

- simple graphs
- list of latest values
- list of 500 latest values
- item configuration
- the option to **execute a check** for new item value immediately (see also **mass actions**)

**Buttons**

View mode buttons being common for all sections are described on the **Monitoring** page.

**Mass actions**

Buttons below the list offer mass actions with one or several selected items:

- Display stacked graph - display a stacked **ad-hoc graph**
- Display graph - display a simple **ad-hoc graph**
- Execute now - execute a check for new item values immediately. Supported for **passive** checks only (see more details). This option is available only for hosts with read-write access. Accessing this option for hosts with read-only permissions depends on the **user role** option called Invoke “Execute now” on read-only hosts.
To use these options, mark the checkboxes before the respective items, then click on the required button.

Using filter

You can use the filter to display only the items you are interested in. For better search performance, data is searched with macros unresolved.

The filter icon is located above the table and the subfilter. Click on it to expand the filter.

The filter allows to narrow the list by host group, host, item name, tag and other settings. Specifying a parent host group in the filter implicitly selects all nested host groups. See Monitoring -> Problems for details on filtering by tags.

Show details allows to extend the information displayed for the items. Such details as the refresh interval, history and trends settings, item type, and item errors (fine/unsupported) are displayed.

Saving filter

Favorite filter settings can be saved as tabs and then quickly accessed by clicking on the respective tab above the filter.

See more details about saving filters.

Using subfilter

The subfilter is useful for a quick one-click access to groups of related items. The subfilter operates autonomously from the main filter - results are filtered immediately, no need to click on Apply in the main filter.

Note that the subfilter only allows to further modify the filtering from the main filter.

Unlike the main filter, the subfilter is updated together with each table refresh request to always get up-to-date information of available filtering options and their counter numbers.

The subfilter shows clickable links allowing to filter items based on a common entity group - the host, tag name or tag value. As soon as the entity is clicked, items are immediately filtered; the selected entity is highlighted with gray background. To remove the filtering, click on the entity again. To add another entity to the filtered results, click on another entity.

For each entity group (e.g. tags, hosts) up to 10 rows of entities are displayed. If there are more entities, this list can be expanded to a maximum of 1000 entries (the value of SUBFILTER_VALUES_PER_GROUP in frontend definitions) by clicking on a three-dot icon displayed at the end. Once expanded to the maximum, the list cannot be collapsed.

In the list of Tag values up to 10 rows of tag names are displayed. If there are more tag names with values, this list can be expanded to a maximum of 200 tag names by clicking on a three-dot icon displayed at the bottom. Once expanded to the maximum, the list cannot be collapsed.

For each tag name up to 10 rows of values are displayed (expandable to 1000 entries (the value of SUBFILTER_VALUES_PER_GROUP in frontend definitions)).

The host options in the subfilter are available only if no hosts or more than one host is selected in the main filter.

By default, items with and without data are displayed in the item list. If only one host is selected in the main filter, the subfilter offers the option to filter only items with data, only without data, or both for this host.

A number next to each clickable entity indicates the number of items it has in the results of the main filter. Entities without items are not displayed, unless they were selected in the subfilter before.

Once one entity is selected, the numbers with other available entities are displayed with a plus sign indicating how many items may be added to the current selection.
Graphs

Links to value history/simple graph

The last column in the latest value list offers:

- a **History** link (for all textual items) - leading to listings (Values/500 latest values) displaying the history of previous item values.
- a **Graph** link (for all numeric items) - leading to a **simple graph**. However, once the graph is displayed, a dropdown on the upper right offers a possibility to switch to Values/500 latest values as well.

The values displayed in this list are “raw”, that is, no postprocessing is applied.

The total amount of values displayed is defined by the value of Limit for search and filter results parameter, set in Administration → General.

5 Maps

Overview

In the Monitoring → Maps section you can configure, manage and view network maps.

When you open this section, you will either see the last map you accessed or a listing of all maps you have access to.

All maps can be either public or private. Public maps are available to all users, while private maps are accessible only to their owner and the users the map is shared with.

Map listing

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the map. Click on the name to view the map.</td>
</tr>
<tr>
<td>Width</td>
<td>Map width is displayed.</td>
</tr>
<tr>
<td>Height</td>
<td>Map height is displayed.</td>
</tr>
<tr>
<td>Actions</td>
<td>Two actions are available:</td>
</tr>
<tr>
<td></td>
<td><strong>Properties</strong> - edit general map properties</td>
</tr>
<tr>
<td></td>
<td><strong>Constructor</strong> - access the grid for adding map elements</td>
</tr>
</tbody>
</table>

To configure a new map, click on the Create map button in the top right-hand corner. To import a map from a YAML, XML, or JSON file, click on the Import button in the top right-hand corner. The user who imports the map will be set as its owner.
Two buttons below the list offer some mass-editing options:

- Export - export the maps to a YAML, XML, or JSON file
- Delete - delete the maps

To use these options, mark the checkboxes before the respective maps, then click on the required button.

Using filter

You can use the filter to display only the maps you are interested in. For better search performance, data is searched with macros unresolved.

Viewing maps

To view a map, click on its name in the list of all maps.

You can use the drop-down in the map title bar to select the lowest severity level of the problem triggers to display. The severity marked as default is the level set in the map configuration. If the map contains a sub-map, navigating to the sub-map will retain the higher-level map severity (except if it is Not classified, in this case, it will not be passed to the sub-map).

Icon highlighting

If a map element is in problem status, it is highlighted with a round circle. The fill color of the circle corresponds to the severity color of the problem. Only problems on or above the selected severity level will be displayed with the element. If all problems are acknowledged, a thick green border around the circle is displayed.

Additionally:

- a host in maintenance is highlighted with an orange, filled square. Note that maintenance highlighting has priority over the problem severity highlighting, if the map element is host.
- a disabled (not-monitored) host is highlighted with a gray, filled square.

Highlighting is displayed if the Icon highlighting check-box is marked in map configuration.

Recent change markers

Inward pointing red triangles around an element indicate a recent trigger status change - one that’s happened within the last 30 minutes. These triangles are shown if the Mark elements on trigger status change check-box is marked in map configuration.

Links

Clicking on a map element opens a menu with some available links.

Buttons

Buttons to the right offer the following options:
Go to map constructor to edit the map content.

Add map to the favorites widget in the Dashboard.

The map is in the favorites widget in the Dashboard. Click to remove map from the favorites widget.

View mode buttons being common for all sections are described on the Monitoring page.

Readable summary in maps

A hidden "aria-label" property is available allowing map information to be read with a screen reader. Both general map description and individual element description is available, in the following format:

- for map description: `<Map name>, <*> of <*> items in problem state>, <*> problems in total>.`
- for describing one element with one problem: `<Element type>, Status <Element status>, <Element name>, <Problem description>.`
- for describing one element with multiple problems: `<Element type>, Status <Element status>, <Element name>, <*> problems>.`
- for describing one element without problems: `<Element type>, Status <Element status>, <Element name>.`

For example, this description is available:

'Local network, 1 of 6 elements in problem state, 1 problem in total. Host, Status problem, My host, Free disk space is less than 20% on volume / server. Host, Status ok, Server 1. Host, Status ok, Server 2. Host, Status ok, Server 3. Host, Status ok, Server 4. ' for the following map:

Referencing a network map

Network maps can be referenced by both sysmapid and mapname GET parameters. For example, http://zabbix/zabbix/zabbix.php?action=map.view&mapname=Local%20network will open the map with that name (Local network).

If both sysmapid (map ID) and mapname (map name) are specified, mapname has higher priority.
6 Discovery

Overview

In the Monitoring → Discovery section results of network discovery are shown. Discovered devices are sorted by the discovery rule.

If a device is already monitored, the host name will be listed in the Monitored host column, and the duration of the device being discovered or lost after previous discovery is shown in the Uptime/Downtime column.

After that follow the columns showing the state of individual services for each discovered device (red cells show services that are down). Service uptime or downtime is included within the cell.

Only those services that have been found on at least one device will have a column showing their state.

Buttons

View mode buttons being common for all sections are described on the Monitoring page.

Using filter

You can use the filter to display only the discovery rules you are interested in. For better search performance, data is searched with macros unresolved.

With nothing selected in the filter, all enabled discovery rules are displayed. To select a specific discovery rule for display, start typing its name in the filter. All matching enabled discovery rules will be listed for selection. More than one discovery rule can be selected.

2 Services

Overview

The Services menu is for the service monitoring functions of Zabbix.

1 Services

Overview

In this section you can see a high-level status of whole services that have been configured in Zabbix, based on your infrastructure.

A service may be a hierarchy consisting of several levels of other services, called “child” services, which are attributed to the overall status of the service (see also an overview of the service monitoring functionality.)

The main categories of service status are OK or Problem, where the Problem status is expressed by the corresponding problem severity name and color.

While the view mode allows to monitor services with their status and other details, you can also configure the service hierarchy in this section (add/edit services, child services) by switching to the edit mode.

To switch from the view to the edit mode (and back) click on the respective button in the upper right corner:

- view services
- add/edit services, and child services

Note that access to editing depends on user role settings.

Viewing services

A list of the existing services is displayed.

Displayed data:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Service name. The service name is a link to service details. The number after the name indicates how many child services the service has.</td>
</tr>
<tr>
<td>Status</td>
<td>Service status: OK - no problems (trigger color and severity) - indicates a problem and its severity. If there are multiple problems, the color and severity of the problem with highest severity is displayed.</td>
</tr>
<tr>
<td>Root cause</td>
<td>Underlying problems that directly or indirectly affect the service status are listed. The same problems are listed as returned by the (SERVICE.ROOTCAUSE) macro. Click on the problem name to see more details about it in Monitoring → Problems. Problems that do not affect the service status are not in the list.</td>
</tr>
<tr>
<td>Created at</td>
<td>The time when the service was created is displayed.</td>
</tr>
<tr>
<td>Tags</td>
<td>Tags of the service are displayed. Tags are used to identify a service in service actions and SLAs.</td>
</tr>
</tbody>
</table>

Buttons

View mode buttons being common for all sections are described on the Monitoring page.

Using filter

You can use the filter to display only the services you are interested in.

Editing services

Click on the Edit button to access the edit mode. When in edit mode, the listing is complemented with checkboxes before the entries and also these additional options:

- add a child service to this service
- edit this service
- delete this service

To configure a new service, click on the Create service button in the top right-hand corner.

Service details

To access service details, click on the service name. To return to the list of all services, click on All services.

Service details include the info box and the list of child services.
To access the info box, click on the Info tab. The info box contains the following entries:

- Names of parent services (if any)
- Current status of this service
- Current SLA(s) of this service, in the format **SLA name:service level indicator**. ‘SLA name’ is also a link to the SLA report for this service. If you position the mouse on the info box next to the service-level indicator (SLI), a pop-up info list is displayed with SLI details. The service-level indicator displays the current service level, in percentage.
- Service tags

The info box also contains a link to the service configuration.

To use the filter for child services, click on the Filter tab.

When in edit mode, the child service listing is complemented with additional editing options:

- add a child service to this service
- edit this service
- delete this service

## 2 Service actions

### Overview

In the Services → Service actions section users can configure and maintain service actions. Configured actions are displayed in the list with respect to the user role permissions. Users will only see actions for services their user role grants access to.

Displayed data, filter and available mass editing options are the same as for other types of actions.

## 3 SLA

### Overview

This section allows to view and configure SLAs.

### SLAs

A list of the configured SLAs is displayed. Note that only the SLAs related to services accessible to the user will be displayed (as read-only, unless Manage SLA is enabled for the user role).

Displayed data:
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The SLA name is displayed. The name is a link to SLA configuration.</td>
</tr>
<tr>
<td>SLO</td>
<td>The service level objective (SLO) is displayed.</td>
</tr>
<tr>
<td>Effective date</td>
<td>The date of starting SLA calculation is displayed.</td>
</tr>
<tr>
<td>Reporting period</td>
<td>The period used in the SLA report is displayed - daily, weekly, monthly, quarterly, or annually.</td>
</tr>
<tr>
<td>Time zone</td>
<td>The SLA time zone is displayed.</td>
</tr>
<tr>
<td>Schedule</td>
<td>The SLA schedule is displayed - 24x7 or custom.</td>
</tr>
<tr>
<td>SLA report</td>
<td>Click on the link to see the SLA report for this SLA.</td>
</tr>
<tr>
<td>Status</td>
<td>The SLA status is displayed - enabled or disabled.</td>
</tr>
</tbody>
</table>

4 SLA report

Overview

This section allows to view SLA reports, based on the criteria selected in the filter. SLA reports can also be displayed as a dashboard widget.

Report

The filter allows to select the report based on the SLA name as well as the service name. It is also possible to limit the displayed period.

Each column (period) displays the SLI for that period. SLIs that are in breach of the set SLO are highlighted in red.

20 periods are displayed in the report. A maximum of 100 periods can be displayed, if both the From date and To date are specified.

Report details

If you click on the service name in the report, you can access another report that displays a more detailed view.
3 Inventory

Overview
The Inventory menu features sections providing an overview of host inventory data by a chosen parameter as well as the ability to view host inventory details.

1 Overview

Overview
The Inventory → Overview section provides ways of having an overview of host inventory data.
For an overview to be displayed, choose host groups (or none) and the inventory field by which to display data. The number of hosts corresponding to each entry of the chosen field will be displayed.

The completeness of an overview depends on how much inventory information is maintained with the hosts.
Numbers in the Host count column are links; they lead to these hosts being filtered out in the Host Inventories table.

2 Hosts

Overview
In the Inventory → Hosts section inventory data of hosts are displayed.
You can filter the hosts by host group(s) and by any inventory field to display only the hosts you are interested in.

To display all host inventories, select no host group in the filter, clear the comparison field in the filter and press "Filter".
While only some key inventory fields are displayed in the table, you can also view all available inventory information for that host. To do that, click on the hostname in the first column.
Inventory details

The **Overview** tab contains some general information about the host along with links to predefined scripts, latest monitoring data and host configuration options:

The **Details** tab contains all available inventory details for the host:

The completeness of inventory data depends on how much inventory information is maintained with the host. If no information is maintained, the Details tab is disabled.

**4 Reports**

**Overview**

The Reports menu features several sections that contain a variety of predefined and user-customizable reports focused on displaying an overview of such parameters as system information, triggers and gathered data.

**1 System information**

**Overview**

In Reports → System information a summary of key Zabbix server and system data is displayed.

Note that in a high availability setup, it is possible to redirect the system information source (server instance) by editing the ui/conf/zabbix.conf.php file - uncomment and set $ZBX_SERVER, $ZBX_SERVER_PORT to a server other than the one shown active.

With the high availability setup enabled, a separate block is displayed below the system stats with details of high availability nodes. This block is visible to Zabbix Super Admin users only.

System information is also available as a dashboard **widget**.

System stats
Displayed data:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zabbix server is running</td>
<td>Yes, 192.168.1.100:10051</td>
<td>Location and port of Zabbix server.</td>
</tr>
<tr>
<td>Number of hosts</td>
<td>3</td>
<td>Number of monitored hosts/not monitored hosts.</td>
</tr>
<tr>
<td>Number of templates</td>
<td>140</td>
<td>Number of monitored templates.</td>
</tr>
<tr>
<td>Number of items</td>
<td>136</td>
<td>Number of monitored/items/disabled/unsupported items.</td>
</tr>
<tr>
<td>Number of triggers</td>
<td>89</td>
<td>Number of enabled/disabled triggers.</td>
</tr>
<tr>
<td>Number of users</td>
<td>3</td>
<td>Number of users online.</td>
</tr>
<tr>
<td>Required server performance, new values per second</td>
<td>1.94</td>
<td>Required server performance is an estimate and can be useful as a guideline. For precise numbers of values processed, use the <code>zabbix[wcach*, values, all]</code> internal item.</td>
</tr>
<tr>
<td>Database history tables upgraded</td>
<td>No - database history tables have not been upgraded</td>
<td>Enabled items from monitored hosts are included in the calculation. Log items are counted as one value per item update interval. Regular interval values are counted; flexible and scheduling interval values are not. The calculation is not adjusted during a &quot;nodata&quot; maintenance period. Trapper items are not counted. This field is displayed if database upgrade to extended range for numeric (float) values has not been completed. See instructions for enabling an extended range of numeric (float) values. If enabled, the failover delay is displayed.</td>
</tr>
<tr>
<td>High availability cluster</td>
<td>enabled - at least one high availability node exists</td>
<td></td>
</tr>
</tbody>
</table>

System information will also display an error message if:

- the database used does not have the required character set or collation (UTF-8);
- the version of the database is below or above supported range (available only to users with the Super admin role type).

High availability nodes
If **high availability cluster** is enabled, then another block of data is displayed with the status of each high availability node.

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Last access</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>node-1</td>
<td>192.168.1.13:10051</td>
<td>12h</td>
<td>Active</td>
</tr>
<tr>
<td>node-2</td>
<td>192.168.1.10:10053</td>
<td>1h 2m 40s</td>
<td>Unavailable</td>
</tr>
<tr>
<td>node-3</td>
<td>192.168.1.11:10053</td>
<td>3h 40s</td>
<td>Unavailable</td>
</tr>
<tr>
<td>node-4</td>
<td>192.168.1.8:16052</td>
<td>1h 3m 20s</td>
<td>Stopped</td>
</tr>
<tr>
<td>node-5</td>
<td>192.168.1.9:16053</td>
<td>1h 3m 51s</td>
<td>Stopped</td>
</tr>
<tr>
<td>node-6</td>
<td>192.168.1.12:10051</td>
<td>21h 16s</td>
<td>Stopped</td>
</tr>
<tr>
<td>node-1</td>
<td>192.168.1.5:16051</td>
<td>17h</td>
<td>Standby</td>
</tr>
<tr>
<td>node-2</td>
<td>192.168.1.6:16051</td>
<td>18h</td>
<td>Standby</td>
</tr>
<tr>
<td>node-3</td>
<td>192.168.1.7:16052</td>
<td>10/20/2021 11:38:47</td>
<td>Standby</td>
</tr>
</tbody>
</table>

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Node name, as defined in server configuration.</td>
</tr>
<tr>
<td>Address</td>
<td>Node IP address and port.</td>
</tr>
<tr>
<td>Last access</td>
<td>Time of node last access. Hovering over the cell shows the timestamp of last access in long format.</td>
</tr>
</tbody>
</table>
| Status    | Node status:  
  - **Active** - node is up and working  
  - **Unavailable** - node hasn’t been seen for more than failover delay (you may want to find out why)  
  - **Stopped** - node has been stopped or couldn’t start (you may want to start it or delete it)  
  - **Standby** - node is up and waiting |

2 **Scheduled reports**

Overview

In the Reports → Scheduled reports users with sufficient permissions can configure scheduled generation of PDF versions of the dashboards, which will be sent by email to specified recipients.

The opening screen displays information about scheduled reports, which can be filtered out for easy navigation - see Using filter section below.

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the report</td>
</tr>
<tr>
<td>Owner</td>
<td>User that created the report</td>
</tr>
<tr>
<td>Repeats</td>
<td>Report generation frequency (daily/weekly/monthly/yearly)</td>
</tr>
<tr>
<td>Period</td>
<td>Period for which the report is prepared</td>
</tr>
<tr>
<td>Last sent</td>
<td>The date and time when the latest report has been sent</td>
</tr>
<tr>
<td>Status</td>
<td>Current status of the report (enabled/disabled/expired). Users with sufficient permissions can change the status by clicking on it - from Enabled to Disabled (and back); from Expired to Disabled (and back). Displayed as a text for users with insufficient rights.</td>
</tr>
</tbody>
</table>
### Using filter

You may use the filter to narrow down the list of reports. For better search performance, data is searched with macros unresolved.

The following filtering options are available:

- Name - partial name match is allowed;
- Report owner - created by current user or all reports;
- Status - select between any (show all reports), enabled, disabled, or expired.

The filter is located above the Scheduled reports bar. It can be opened and collapsed by clicking on the Filter tab in the upper right corner.

### Mass update

Sometimes you may want to change status or delete a number of reports at once. Instead of opening each individual report for editing, you may use the mass update function for that.

To mass-update some reports, do the following:

- Mark the checkboxes of the reports to update in the list
- Click on the required button below the list to make changes (Enable, Disable or Delete).

### 3 Availability report

#### Overview

In Reports → Availability report you can see what proportion of time each trigger has been in problem/ok state. The percentage of time for each state is displayed.

Thus it is easy to determine the availability situation of various elements on your system.

From the drop-down in the upper right corner, you can choose the selection mode - whether to display triggers by hosts or by triggers belonging to a template.
The name of the trigger is a link to the latest events of that trigger.

Using filter

The filter can help narrow down the number of hosts and/or triggers displayed. For better search performance, data is searched with macros unresolved.

The filter is located below the Availability report bar. It can be opened and collapsed by clicking on the Filter tab on the left.

Filtering by trigger template

In the by trigger template mode results can be filtered by one or several parameters listed below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template group</td>
<td>Select all hosts with triggers from templates belonging to that group.</td>
</tr>
<tr>
<td>Template</td>
<td>Select hosts with triggers from the chosen template and all nested templates. Only triggers inherited from the selected template will be displayed. If a nested template has additional own triggers, those triggers will not be displayed.</td>
</tr>
<tr>
<td>Template trigger</td>
<td>Select hosts with chosen trigger. Other triggers of the selected hosts will not be displayed.</td>
</tr>
<tr>
<td>Host group</td>
<td>Select hosts belonging to the group.</td>
</tr>
</tbody>
</table>

Filtering by host

In the by host mode results can be filtered by a host or by the host group. Specifying a parent host group implicitly selects all nested host groups.

Time period selector

The time period selector allows to select often required periods with one mouse click. The time period selector can be opened by clicking on the time period tab next to the filter.

Clicking on Show in the Graph column displays a bar graph where availability information is displayed in bar format each bar representing a past week of the current year.
The green part of a bar stands for OK time and red for problem time.

4 Triggers top 100

Overview

In Reports → Triggers top 100 you can see the triggers that have changed their state most often within the period of evaluation, sorted by the number of status changes.

Both host and trigger column entries are links that offer some useful options:

- for host - links to user-defined scripts, latest data, inventory, graphs, and dashboards for the host
- for trigger - links to latest events, the trigger configuration form, and a simple graph

Using filter

You may use the filter to display triggers by host group, host, or trigger severity. Specifying a parent host group implicitly selects all nested host groups. For better search performance, data is searched with macros unresolved.

The filter is located below the 100 busiest triggers bar. It can be opened and collapsed by clicking on the Filter tab on the left.

Time period selector

The time period selector allows to select often required periods with one mouse click. The time period selector can be opened by clicking on the time period tab next to the filter.
5 Audit

Overview

In the Reports → Audit section users can view records of changes made in the frontend.

Audit logging should be enabled in the Administration settings to display audit records. If logging is disabled, history of frontend changes does not get recorded to the database and audit records cannot be viewed.

Audit log displays the following data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Timestamp of the audit record.</td>
</tr>
<tr>
<td>User</td>
<td>User who performed the activity.</td>
</tr>
<tr>
<td>IP</td>
<td>IP from which the activity was initiated.</td>
</tr>
<tr>
<td>Resource</td>
<td>Type of the affected resource (host, host group, etc.).</td>
</tr>
<tr>
<td>Action</td>
<td>Activity type: Login, Logout, Added, Updated, Deleted, Enabled, or Disabled.</td>
</tr>
<tr>
<td>ID</td>
<td>ID of the affected resource. Clicking on the hyperlink will result in filtering audit log records by this resource ID.</td>
</tr>
<tr>
<td>Recordset ID</td>
<td>Shared ID for all audit log records created as a result of the same frontend operation. For example, when linking a template to a host, a separate audit log record is created for each inherited template entity (item, trigger, etc.) - all these records will have the same Recordset ID. Clicking on the hyperlink will result in filtering audit log records by this Recordset ID.</td>
</tr>
<tr>
<td>Details</td>
<td>Description of the resource and detailed information about the performed activity. If a record contains more than two rows, an additional link Details will be displayed. Click on this link to view the full list of changes.</td>
</tr>
</tbody>
</table>

Using filter

The filter is located below the Audit log bar. It can be opened and collapsed by clicking on the Filter tab in the upper right corner.

You may use the filter to narrow down the records by user, affected resource, resource ID and frontend operation (Recordset ID). One or more actions (e.g., add, update, delete, etc) for the resource can be selected in the filter.

For better search performance, all data is searched with macros unresolved.

Time period selector

The time period selector allows to select often required periods with one mouse click. The time period selector can be opened by clicking on the time period tab next to the filter.

6 Action log

Overview

In the Reports → Action log section users can view details of operations (notifications, remote commands) executed within an action.
Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Timestamp of the operation.</td>
</tr>
<tr>
<td>Action</td>
<td>Name of the action causing operations is displayed.</td>
</tr>
<tr>
<td>Type</td>
<td>Operation type is displayed - Email or Command.</td>
</tr>
<tr>
<td>Recipient(s)</td>
<td>Username, name, surname (in parentheses) and e-mail address of the notification recipient is displayed.</td>
</tr>
<tr>
<td>Message</td>
<td>The content of the message/remote command is displayed.</td>
</tr>
<tr>
<td>Status</td>
<td>Operation status is displayed:</td>
</tr>
<tr>
<td></td>
<td>In progress - action is in progress</td>
</tr>
<tr>
<td></td>
<td>For actions in progress the number of retries left is displayed - the remaining number of times the server will try to send the notification.</td>
</tr>
<tr>
<td></td>
<td>Sent - notification has been sent</td>
</tr>
<tr>
<td></td>
<td>Executed - command has been executed</td>
</tr>
<tr>
<td></td>
<td>Not sent - action has not been completed.</td>
</tr>
<tr>
<td>Info</td>
<td>Error information (if any) regarding the action execution is displayed.</td>
</tr>
</tbody>
</table>

Using filter

You may use the filter to narrow down the records by the message recipient(s). For better search performance, data is searched with macros unresolved.

The filter is located below the Action log bar. It can be opened and collapsed by clicking on the Filter tab on the left.

Time period selector

The time period selector allows to select often required periods with one mouse click. The time period selector can be opened by clicking on the time period tab next to the filter.

7 Notifications

Overview

In the Reports → Notifications section a report on the number of notifications sent to each user is displayed.

From the dropdowns in the top right-hand corner you can choose the media type (or all), period (data for each day/week/month/year) and year for the notifications sent.
Each column displays totals per one system user.

5 Configuration

Overview

The Configuration menu contains sections for setting up major Zabbix functions, such as hosts and host groups, data gathering, data thresholds, sending problem notifications, creating data visualization and others.

1 Items

Overview

The item list for a template can be accessed from Configuration → Templates by clicking on Items for the respective template. A list of existing items is displayed.

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item menu</td>
<td>Click on the three-dot icon to open the menu for this specific item with these options:</td>
</tr>
<tr>
<td></td>
<td>Create trigger - create a trigger based on this item</td>
</tr>
<tr>
<td></td>
<td>Triggers - click to see a list with links to already-configured trigger of this item</td>
</tr>
<tr>
<td></td>
<td>Create dependent item - create a dependent item for this item</td>
</tr>
<tr>
<td></td>
<td>Create dependent discovery rule - create a dependent discovery rule for this item</td>
</tr>
<tr>
<td>Template</td>
<td>Template the item belongs to.</td>
</tr>
<tr>
<td></td>
<td>This column is displayed only if multiple templates are selected in the filter.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the item displayed as a blue link to item details.</td>
</tr>
<tr>
<td></td>
<td>Clicking on the item name link opens the item configuration form.</td>
</tr>
<tr>
<td></td>
<td>If the item is inherited from another template, the template name is displayed before the item name, as a gray link. Clicking on the template link will open the item list on that template level.</td>
</tr>
<tr>
<td>Triggers</td>
<td>Moving the mouse over Triggers will display an infobox displaying the triggers associated with the item.</td>
</tr>
<tr>
<td></td>
<td>The number of the triggers is displayed in gray.</td>
</tr>
<tr>
<td>Key</td>
<td>Item key is displayed.</td>
</tr>
<tr>
<td>Column</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Interval</td>
<td>Frequency of the check is displayed.</td>
</tr>
<tr>
<td>History</td>
<td>How many days item data history will be kept is displayed.</td>
</tr>
<tr>
<td>Trends</td>
<td>How many days item trends history will be kept is displayed.</td>
</tr>
<tr>
<td>Type</td>
<td>Item type is displayed (Zabbix agent, SNMP agent, simple check, etc).</td>
</tr>
<tr>
<td>Status</td>
<td>Item status is displayed - Enabled or Disabled. By clicking on the status you can change it - from Enabled to Disabled (and back).</td>
</tr>
<tr>
<td>Tags</td>
<td>Item tags are displayed. Up to three tags (name:value pairs) can be displayed. If there are more tags, a &quot;...&quot; link is displayed that allows to see all tags on mouseover.</td>
</tr>
</tbody>
</table>

To configure a new item, click on the Create item button at the top right corner.

**Mass editing options**

Buttons below the list offer some mass-editing options:

- Enable - change item status to Enabled.
- Disable - change item status to Disabled.
- Copy - copy the items to other hosts or templates.
- Mass update - update several properties for a number of items at once.
- Delete - delete the items.

To use these options, mark the checkboxes before the respective items, then click on the required button.

**Using filter**

The item list may contain a lot of items. By using the filter, you can filter out some of them to quickly locate the items you’re looking for. For better search performance, data is searched with macros unresolved.

The Filter icon is available at the top right corner. Clicking on it will open a filter where you can specify the desired filtering criteria.

---

**Parameter Description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template groups</td>
<td>Filter by one or more template groups. Specifying a parent template group implicitly selects all nested groups.</td>
</tr>
<tr>
<td>Templates</td>
<td>Filter by one or more templates.</td>
</tr>
<tr>
<td>Name</td>
<td>Filter by item name.</td>
</tr>
<tr>
<td>Key</td>
<td>Filter by item key.</td>
</tr>
<tr>
<td>Value mapping</td>
<td>Filter by the value map used.</td>
</tr>
<tr>
<td>Type</td>
<td>Filter by item type (Zabbix agent, SNMP agent, etc.).</td>
</tr>
<tr>
<td>Type of information</td>
<td>Filter by type of information (Numeric unsigned, float, etc.).</td>
</tr>
<tr>
<td>History</td>
<td>Filter by how long item history is kept.</td>
</tr>
<tr>
<td>Trends</td>
<td>Filter by how long item trends are kept.</td>
</tr>
</tbody>
</table>

---

766
The **Subfilter** below the filter offers further filtering options (for the data already filtered). You can select groups of items with a common parameter value. Upon clicking on a group, it gets highlighted and only the items with this parameter value remain in the list.

### 2 Triggers

#### Overview

The trigger list for a template can be accessed from Configuration → Templates by clicking on Triggers for the respective template.

<table>
<thead>
<tr>
<th>Severity</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information</td>
<td>Template Module Linux generic by Zabbix agent: Webserver has a screen changed</td>
<td>(test)Linux OS agent/file/directory/changed or=testLinux OS agent/file/directory/changed=A2</td>
</tr>
<tr>
<td>Information</td>
<td>Template Module Linux generic by Zabbix agent: Configuration a number of open file descriptions is too low (&lt; 5000) [KERNEL_MAXFILES_MIN]</td>
<td>lastLinux OS agent/kennel/filesystem/[KERNEL_MAXFILES_MIN]</td>
</tr>
<tr>
<td>Information</td>
<td>Template Module Linux generic by Zabbix agent: Configuration a number of processes is too low (&lt; 5000) [KERNEL_MAXPROC_MIN]</td>
<td>lastLinux OS agent/kennel/processes/[KERNEL_MAXPROC_MIN]</td>
</tr>
<tr>
<td>Warning</td>
<td>Template Module Linux generic by Zabbix agent: Getting closer to process limit (over 60%)</td>
<td>(TESTLASTVALUES) active. (TESTLASTVALUES) less. &lt;=50%</td>
</tr>
<tr>
<td>Warning</td>
<td>Template Module Linux CPU by Zabbix agent: CPU utilization (top) [CPU_UTIL_CRIT]% to 95%</td>
<td>Current utilization (TESTLASTVALUES)</td>
</tr>
<tr>
<td>Column</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Operational data</td>
<td>Operational data definition of the trigger, containing arbitrary strings and macros that will resolve dynamically in Monitoring → Problems.</td>
<td></td>
</tr>
<tr>
<td>Expression</td>
<td>Trigger expression is displayed. The template-item part of the expression is displayed as a link, leading to the item configuration form.</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Trigger status is displayed - Enabled or Disabled. By clicking on the status you can change it - from Enabled to Disabled (and back).</td>
<td></td>
</tr>
<tr>
<td>Tags</td>
<td>If a trigger contains tags, tag name and value are displayed in this column.</td>
<td></td>
</tr>
</tbody>
</table>

To configure a new trigger, click on the Create trigger button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:
- Enable - change trigger status to Enabled
- Disable - change trigger status to Disabled
- Copy - copy the triggers to other hosts or templates
- Mass update - update several properties for a number of triggers at once
- Delete - delete the triggers

To use these options, mark the checkboxes before the respective triggers, then click on the required button.

Using filter

You can use the filter to display only the triggers you are interested in. For better search performance, data is searched with macros unresolved.

The Filter icon is available at the top right corner. Clicking on it will open a filter where you can specify the desired filtering criteria.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template groups</td>
<td>Filter by one or more template groups. Specifying a parent template group implicitly selects all nested groups.</td>
</tr>
<tr>
<td>Templates</td>
<td>Filter by one or more templates. If template groups are already selected above, template selection is limited to those groups.</td>
</tr>
<tr>
<td>Name</td>
<td>Filter by trigger name.</td>
</tr>
<tr>
<td>Severity</td>
<td>Select to filter by one or several trigger severities.</td>
</tr>
<tr>
<td>Status</td>
<td>Filter by trigger status.</td>
</tr>
<tr>
<td>Tags</td>
<td>Filter by trigger tag name and value. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive.</td>
</tr>
</tbody>
</table>

There are several operators available for each condition:
- **Exists** - include the specified tag names
- **Equals** - include the specified tag names and values (case-sensitive)
- **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)
- **Does not exist** - exclude the specified tag names
- **Does not equal** - exclude the specified tag names and values (case-sensitive)
- **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)

There are two calculation types for conditions:
- **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition
- **Or** - enough if one condition is met

Macros and **macro functions** are supported in tag name and tag value fields.
### Parameter Description

<table>
<thead>
<tr>
<th>Inherited</th>
<th>Filter triggers inherited (or not inherited) from linked templates.</th>
</tr>
</thead>
<tbody>
<tr>
<td>With dependencies</td>
<td>Filter triggers with (or without) dependencies.</td>
</tr>
</tbody>
</table>

#### 3 Graphs

#### Overview

The custom graph list for a template can be accessed from Configuration → Templates by clicking on Graphs for the respective template.

A list of existing graphs is displayed.

#### Graphs

<table>
<thead>
<tr>
<th>Displayed data:</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Template</strong></td>
<td>Template the graph belongs to. This column is displayed only if multiple templates are selected in the filter.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Name of the custom graph, displayed as a blue link to graph details. Clicking on the graph name link opens the graph configuration form. If the graph is inherited from another template, the template name is displayed before the graph name, as a gray link. Clicking on the template link will open the graph list on that template level.</td>
</tr>
<tr>
<td><strong>Width</strong></td>
<td>Graph width is displayed.</td>
</tr>
<tr>
<td><strong>Height</strong></td>
<td>Graph height is displayed.</td>
</tr>
<tr>
<td><strong>Graph type</strong></td>
<td>Graph type is displayed - Normal, Stacked, Pie or Exploded.</td>
</tr>
</tbody>
</table>

To configure a new graph, click on the Create graph button at the top right corner.

#### Mass editing options

Buttons below the list offer some mass-editing options:

- Copy - copy the graphs to other hosts or templates
- Delete - delete the graphs

To use these options, mark the checkboxes before the respective graphs, then click on the required button.

#### Using filter

You can filter graphs by template group and template. For better search performance, data is searched with macros unresolved.

#### 4 Discovery rules

#### Overview

The list of low-level discovery rules for a template can be accessed from Configuration → Templates by clicking on Discovery for the respective template.

A list of existing low-level discovery rules is displayed. It is also possible to see all discovery rules independently of the template, or all discovery rules of a specific template group by changing the filter settings.
Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template</td>
<td>The template discovery rule belongs to.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the rule, displayed as a blue link.</td>
</tr>
<tr>
<td>Items</td>
<td>A link to the list of item prototypes is displayed.</td>
</tr>
<tr>
<td>Triggers</td>
<td>A link to the list of trigger prototypes is displayed.</td>
</tr>
<tr>
<td>Graphs</td>
<td>A link to the list of graph prototypes displayed.</td>
</tr>
<tr>
<td>Hosts</td>
<td>A link to the list of host prototypes displayed.</td>
</tr>
<tr>
<td>Key</td>
<td>The item key used for discovery is displayed.</td>
</tr>
<tr>
<td>Interval</td>
<td>The frequency of performing discovery is displayed.</td>
</tr>
<tr>
<td>Type</td>
<td>The item type used for discovery is displayed (Zabbix agent, SNMP agent, etc).</td>
</tr>
<tr>
<td>Status</td>
<td>Discovery rule status is displayed - Enabled or Disabled. By clicking on the status you can change it - from Enabled to Disabled (and back).</td>
</tr>
</tbody>
</table>

To configure a new low-level discovery rule, click on the Create discovery rule button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- Enable - change the low-level discovery rule status to Enabled
- Disable - change the low-level discovery rule status to Disabled
- Delete - delete the low-level discovery rules

To use these options, mark the checkboxes before the respective discovery rules, then click on the required button.

Using filter

You can use the filter to display only the discovery rules you are interested in. For better search performance, data is searched with macros unresolved.

The Filter icon is available at the top right corner. Clicking on it will open a filter where you can specify the desired filtering criteria such as template, discovery rule name, item key, item type, etc.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template groups</td>
<td>Filter by one or more template groups. Specifying a parent template group implicitly selects all nested groups.</td>
</tr>
<tr>
<td>Templates</td>
<td>Filter by one or more templates.</td>
</tr>
<tr>
<td>Name</td>
<td>Filter by discovery rule name.</td>
</tr>
<tr>
<td>Key</td>
<td>Filter by discovery item key.</td>
</tr>
<tr>
<td>Type</td>
<td>Filter by discovery item type.</td>
</tr>
<tr>
<td>Update interval</td>
<td>Filter by update interval. Not available for Zabbix trapper and dependent items.</td>
</tr>
<tr>
<td>Keep lost resources period</td>
<td>Filter by Keep lost resources period.</td>
</tr>
<tr>
<td>Status</td>
<td>Filter by discovery rule status (All/Enabled/Disabled).</td>
</tr>
</tbody>
</table>

1. Item prototypes

Overview

In this section the configured item prototypes of a low-level discovery rule on the template are displayed.

If the template is linked to the host, item prototypes will become the basis of creating real host items during low-level discovery.

### Item prototypes

<table>
<thead>
<tr>
<th>Name</th>
<th>Key</th>
<th>Interval</th>
<th>History</th>
<th>Trends</th>
<th>Type</th>
<th>Create enabled</th>
<th>Discover</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>(FS/HARD) Free space on %</td>
<td>vhs.fs.hadoop(FS/HARD).free</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Yes</td>
<td>Yes</td>
<td>discovery enabled</td>
</tr>
<tr>
<td>(FS/HARD) Space utilization</td>
<td>vhs.fs.hadoop(FS/HARD).usage</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Yes</td>
<td>Yes</td>
<td>discovery enabled</td>
</tr>
<tr>
<td>(FS/HARD) Total space</td>
<td>vhs.fs.hadoop(FS/HARD).total</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Yes</td>
<td>Yes</td>
<td>discovery enabled</td>
</tr>
<tr>
<td>(FS/HARD) Used space</td>
<td>vhs.fs.hadoop(FS/HARD).used</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Yes</td>
<td>Yes</td>
<td>discovery enabled</td>
</tr>
</tbody>
</table>

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the item prototype, displayed as a blue link. Clicking on the name opens the item prototype configuration form. If the item prototype belongs to a linked template, the template name is displayed before the item name, as a gray link. Clicking on the template link will open the item prototype list on the linked template level.</td>
</tr>
<tr>
<td>Key</td>
<td>Key of the item prototype is displayed.</td>
</tr>
<tr>
<td>Interval</td>
<td>Frequency of the check is displayed.</td>
</tr>
<tr>
<td>History</td>
<td>How many days to keep item data history is displayed.</td>
</tr>
<tr>
<td>Trends</td>
<td>How many days to keep item trends history is displayed.</td>
</tr>
<tr>
<td>Type</td>
<td>Type of the item prototype is displayed (Zabbix agent, SNMP agent, simple check, etc).</td>
</tr>
</tbody>
</table>
| Create enabled  | Create the item based on this prototype as:  
  **Yes** - enabled  
  **No** - disabled. You can switch between ‘Yes’ and ‘No’ by clicking on them.                                                                                                                                   |
| Discover        | Discover the item based on this prototype:  
  **Yes** - discover  
  **No** - do not discover. You can switch between ‘Yes’ and ‘No’ by clicking on them.                                                                                                                      |
| Tags            | Tags of the item prototype is displayed.                                                                                                                                                                      |

To configure a new item prototype, click on the Create item prototype button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- Create enabled - create these items as Enabled
- Create disabled - create these items as Disabled
• Mass update - mass update these item prototypes
• Delete - delete these item prototypes

To use these options, mark the checkboxes before the respective item prototypes, then click on the required button.

2 Trigger prototypes

Overview

In this section the configured trigger prototypes of a low-level discovery rule on the template are displayed.

If the template is linked to the host, trigger prototypes will become the basis of creating real host triggers during low-level discovery.

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the trigger prototype, displayed as a blue link. Clicking on the name opens the trigger prototype configuration form. If the trigger prototype belongs to a linked template, the template name is displayed before the trigger name, as a gray link. Clicking on the template link will open the trigger prototype list on the linked template level.</td>
</tr>
<tr>
<td>Operational data</td>
<td>Format of the operational data of the trigger is displayed, containing arbitrary strings and macros that will resolve dynamically in Monitoring → Problems.</td>
</tr>
<tr>
<td>Create enabled</td>
<td>Create the trigger based on this prototype as: <strong>Yes</strong> - enabled <strong>No</strong> - disabled. You can switch between ‘Yes’ and ‘No’ by clicking on them.</td>
</tr>
<tr>
<td>Discover</td>
<td>Discover the trigger based on this prototype: <strong>Yes</strong> - discover <strong>No</strong> - do not discover. You can switch between ‘Yes’ and ‘No’ by clicking on them.</td>
</tr>
<tr>
<td>Tags</td>
<td>Tags of the trigger prototype are displayed.</td>
</tr>
</tbody>
</table>

To configure a new trigger prototype, click on the Create trigger prototype button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

• Create enabled - create these triggers as Enabled
• Create disabled - create these triggers as Disabled
• Mass update - mass update these trigger prototypes
• Delete - delete these trigger prototypes

To use these options, mark the checkboxes before the respective trigger prototypes, then click on the required button.

3 Graph prototypes
Overview

In this section the configured graph prototypes of a low-level discovery rule on the template are displayed.

If the template is linked to the host, graph prototypes will become the basis of creating real host graphs during low-level discovery.

---

**Graph prototypes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Width</th>
<th>Height</th>
<th>Graph type</th>
<th>Discover</th>
</tr>
</thead>
<tbody>
<tr>
<td>[VMWARE] Disk space usage</td>
<td>600</td>
<td>340</td>
<td>Pie</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Displaying 1 of 1 found

**Displayed data:**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the graph prototype, displayed as a blue link. Clicking on the name opens the graph prototype configuration form. If the graph prototype belongs to a linked template, the template name is displayed before the graph name, as a gray link. Clicking on the template link will open the graph prototype list on the linked template level.</td>
</tr>
<tr>
<td>Width</td>
<td>Width of the graph prototype is displayed.</td>
</tr>
<tr>
<td>Height</td>
<td>Height of the graph prototype is displayed.</td>
</tr>
<tr>
<td>Type</td>
<td>Type of the graph prototype is displayed - Normal, Stacked, Pie or Exploded.</td>
</tr>
<tr>
<td>Discover</td>
<td>Discover the graph based on this prototype: <strong>Yes</strong> - discover <strong>No</strong> - do not discover. You can switch between 'Yes' and 'No' by clicking on them.</td>
</tr>
</tbody>
</table>

To configure a new graph prototype, click on the Create graph prototype button at the top right corner.

**Mass editing options**

Buttons below the list offer some mass-editing options:

- **Delete** - delete these graph prototypes

To use these options, mark the checkboxes before the respective graph prototypes, then click on the required button.

---

**Host prototypes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Template</th>
<th>Discover</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>[VMWARE]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Displaying 1 of 1 found

**Displayed data:**

**Overview**

In this section the configured host prototypes of a low-level discovery rule on the template are displayed.

If the template is linked to the host, host prototypes will become the basis of creating real hosts during low-level discovery.

---

**Host prototypes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Template</th>
<th>Discover</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>[VMWARE] VM</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Displaying 1 of 1 found

**Displayed data:**
**Column** | **Description**
---|---
Name | Name of the host prototype, displayed as a blue link. Clicking on the name opens the host prototype configuration form. If the host prototype belongs to a linked template, the template name is displayed before the host name, as a gray link. Clicking on the template link will open the host prototype list on the linked template level.
Templates | Templates of the host prototype are displayed.
Create enabled | Create the host based on this prototype as:  
*Yes* - enabled  
*No* - disabled. You can switch between ‘Yes’ and ‘No’ by clicking on them.
Discover | Discover the host based on this prototype:  
*Yes* - discover  
*No* - do not discover. You can switch between ‘Yes’ and ‘No’ by clicking on them.
Tags | Tags of the host prototype are displayed.

To configure a new host prototype, click on the Create host prototype button at the top right corner.

**Mass editing options**

Buttons below the list offer some mass-editing options:

- Create enabled - create these hosts as Enabled
- Create disabled - create these hosts as Disabled
- Delete - delete these host prototypes

To use these options, mark the checkboxes before the respective host prototypes, then click on the required button.

**5 Web scenarios**

**Overview**

The web scenario list for a template can be accessed from Configuration → Templates by clicking on Web for the respective template.

A list of existing web scenarios is displayed.

![Web Monitoring Table](image)

**Displayed data:**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the web scenario. Clicking on the web scenario name opens the web scenario configuration form. If the web scenario is inherited from another template, the template name is displayed before the web scenario name, as a gray link. Clicking on the template link will open the web scenarios list on that template level.</td>
</tr>
<tr>
<td>Number of steps</td>
<td>The number of steps the scenario contains.</td>
</tr>
<tr>
<td>Update interval</td>
<td>How often the scenario is performed.</td>
</tr>
<tr>
<td>Attempts</td>
<td>How many attempts for executing web scenario steps are performed.</td>
</tr>
<tr>
<td>Authentication</td>
<td>Authentication method is displayed - Basic, NTLM on None.</td>
</tr>
<tr>
<td>HTTP proxy</td>
<td>Displays HTTP proxy or ‘No’ if not used.</td>
</tr>
<tr>
<td>Status</td>
<td>Web scenario status is displayed - Enabled or Disabled. By clicking on the status you can change it.</td>
</tr>
<tr>
<td>Tags</td>
<td>Web scenario tags are displayed. Up to three tags (name:value pairs) can be displayed. If there are more tags, a “…” link is displayed that allows to see all tags on mouseover.</td>
</tr>
</tbody>
</table>

To configure a new web scenario, click on the Create web scenario button at the top right corner.
Mass editing options

Buttons below the list offer some mass-editing options:

- Enable - change the scenario status to Enabled
- Disable - change the scenario status to Disabled
- Delete - delete the web scenarios

To use these options, mark the checkboxes before the respective web scenarios, then click on the required button.

Using filter

You can use the filter to display only the scenarios you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available above the list of web scenarios. If you click on it, a filter becomes available where you can filter scenarios by template group, status and tags.

---

1 Template groups

Overview

In the Configuration → Templates groups section users can configure and maintain template groups.

A listing of existing template groups with their details is displayed. You can search and filter template groups by name.

### Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the template group. Clicking on the group name opens the group configuration form.</td>
</tr>
<tr>
<td>Templates</td>
<td>Number of templates in the group (displayed in gray) followed by the list of group members. Clicking on a template name will open the template configuration form. Clicking on the number opens the list of templates in this group.</td>
</tr>
</tbody>
</table>

Mass editing options

To delete several template groups at once, mark the checkboxes before the respective groups, then click on the Delete button below the list.

Using filter

You can use the filter to display only the template groups you are interested in. For better search performance, data is searched with macros unresolved.
### 2 Host groups

**Overview**

In the Configuration → Host groups section users can configure and maintain host groups.

A listing of existing host groups with their details is displayed. You can search and filter host groups by name.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the host group. Clicking on the group name opens the group configuration form.</td>
</tr>
<tr>
<td>Hosts</td>
<td>Number of hosts in the group (displayed in gray) followed by the list of group members. Clicking on a host name will open the host configuration form. Clicking on the number will, in the whole listing of hosts, filter out those that belong to the group.</td>
</tr>
<tr>
<td>Info</td>
<td>Error information (if any) regarding the host group is displayed.</td>
</tr>
</tbody>
</table>

**Mass editing options**

Buttons below the list offer some mass-editing options:

- Enable hosts - change the status of all hosts in the group to "Monitored"
- Disable hosts - change the status of all hosts in the group to "Not monitored"
- Delete - delete the host groups

To use these options, mark the checkboxes before the respective host groups, then click on the required button.

**Using filter**

You can use the filter to display only the host groups you are interested in. For better search performance, data is searched with macros unresolved.

### 3 Templates

**Overview**

In the Configuration → Templates section users can configure and maintain templates.

A listing of existing templates with their details is displayed.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Template OS Linux by Prism</td>
</tr>
<tr>
<td>Items</td>
<td>Template OS Linux by Zabbix agent</td>
</tr>
</tbody>
</table>

**Displayed data:**
To configure a new template, click on the Create template button in the top right-hand corner. To import a template from a YAML, XML, or JSON file, click on the Import button in the top right-hand corner.

Using filter

You can use the filter to display only the templates you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available below Create template and Import buttons. If you click on it, a filter becomes available where you can filter templates by template group, linked templates, name and tags.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template groups</td>
<td>Filter by one or more template groups.</td>
</tr>
<tr>
<td>Linked templates</td>
<td>Filter by directly linked templates.</td>
</tr>
<tr>
<td>Name</td>
<td>Filter by template name.</td>
</tr>
<tr>
<td>Tags</td>
<td>Filter by template tag name and value.</td>
</tr>
<tr>
<td></td>
<td>Filtering is possible only by template-level tags (not inherited ones). It is</td>
</tr>
<tr>
<td></td>
<td>possible to include as well as exclude specific tags and tag values. Several</td>
</tr>
<tr>
<td></td>
<td>conditions can be set. Tag name matching is always case-sensitive.</td>
</tr>
<tr>
<td></td>
<td>There are several operators available for each condition:</td>
</tr>
<tr>
<td></td>
<td><strong>Exists</strong> - include the specified tag names</td>
</tr>
<tr>
<td></td>
<td><strong>Equals</strong> - include the specified tag names and values (case-sensitive)</td>
</tr>
<tr>
<td></td>
<td><strong>Contains</strong> - include the specified tag names where the tag values contain</td>
</tr>
<tr>
<td></td>
<td>the entered string (substring match, case-insensitive)</td>
</tr>
<tr>
<td></td>
<td><strong>Does not exist</strong> - exclude the specified tag names</td>
</tr>
<tr>
<td></td>
<td><strong>Does not equal</strong> - exclude the specified tag names and values (case-sensitive)</td>
</tr>
<tr>
<td></td>
<td><strong>Does not contain</strong> - exclude the specified tag names where the tag values</td>
</tr>
<tr>
<td></td>
<td>contain the entered string (substring match, case-insensitive)</td>
</tr>
<tr>
<td></td>
<td>There are two calculation types for conditions:</td>
</tr>
<tr>
<td></td>
<td><strong>And/Or</strong> - all conditions must be met, conditions having the same tag name</td>
</tr>
<tr>
<td></td>
<td>will be grouped by the Or condition</td>
</tr>
<tr>
<td></td>
<td><strong>Or</strong> - enough if one condition is met</td>
</tr>
</tbody>
</table>

Mass editing options

Buttons below the list offer some mass-editing options:

- Export - export the template to a YAML, XML or JSON file
- Mass update - update several properties for a number of templates at once
- Delete - delete the template while leaving its linked entities (items, triggers etc.) with the hosts
• Delete and clear - delete the template and its linked entities from the hosts

To use these options, mark the checkboxes before the respective templates, then click on the required button.

4 Hosts

Overview

In the Configuration → Hosts section users can configure and maintain hosts.

A listing of existing hosts with their details is displayed.

![Hosts Table]

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the host. Clicking on the host name opens the host configuration form.</td>
</tr>
<tr>
<td>Entities (Items, Triggers, Graphs, Discovery, Web)</td>
<td>Clicking on the entity name will display items, triggers etc. of the host. The number of the respective entities is displayed in gray.</td>
</tr>
<tr>
<td>Interface</td>
<td>The main interface of the host is displayed.</td>
</tr>
<tr>
<td>Proxy</td>
<td>Proxy name is displayed, if the host is monitored by a proxy.</td>
</tr>
<tr>
<td></td>
<td>This column is only displayed if the Monitored by filter option is set to ‘Any’ or ‘Proxy’.</td>
</tr>
<tr>
<td>Templates</td>
<td>The templates linked to the host are displayed. If other templates are contained in the linked template, those are displayed in parentheses, separated by a comma. Clicking on a template name will open its configuration form.</td>
</tr>
<tr>
<td>Status</td>
<td>Host status is displayed - Enabled or Disabled. By clicking on the status you can change it.</td>
</tr>
<tr>
<td></td>
<td>An orange wrench icon ✂️ before the host status indicates that this host is in maintenance.</td>
</tr>
<tr>
<td></td>
<td>Maintenance details are displayed when the mouse pointer is positioned over the icon.</td>
</tr>
<tr>
<td>Column</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Availability</td>
<td>Host availability per configured interface is displayed. Icons represent only those interface types (Zabbix agent, SNMP, IPMI, JMX) that are configured. If you position the mouse on the icon, a popup list appears listing all interfaces of this type with details, status and errors (for the agent interface, availability of active checks is also listed). The column is empty for hosts with no interfaces. The current status of all interfaces of one type is displayed by the respective icon color: <strong>Green</strong> - all interfaces available <strong>Yellow</strong> - at least one interface available and at least one unavailable; others can have any value including ‘unknown’ <strong>Red</strong> - no interfaces available <strong>Gray</strong> - at least one interface unknown (none unavailable)</td>
</tr>
</tbody>
</table>

**Active check availability**

Since Zabbix 6.2 active checks also affect host availability, if there is at least one enabled active check on the host. To determine active check availability heartbeat messages are sent in the agent active check thread. The frequency of the heartbeat messages is set by the HeartbeatFrequency parameter in Zabbix agent and agent2 configurations (60 seconds by default, 0-3600 range). Active checks are considered unavailable when the active check heartbeat is older than 2 x HeartbeatFrequency seconds. Note that if Zabbix agents older than 6.2.x are used, they are not sending any active check heartbeats, so the availability of their hosts will remain unknown. Active agent availability is counted towards the total Zabbix agent availability in the same way as a passive interface is (for example, if a passive interface is available, while the active checks are unknown, the total agent availability is set to gray(unknown)).

**Agent encryption**

Encryption status for connections to the host is displayed:

- **None** - no encryption
- **PSK** - using pre-shared key
- **Cert** - using certificate

**Info**

Error information (if any) regarding the host is displayed.

**Tags**

Tags of the host, with macros unresolved.

To configure a new host, click on the Create host button in the top right-hand corner. To import a host from a YAML, XML or JSON file, click on the Import button in the top right-hand corner.

**Mass editing options**

Buttons below the list offer some mass-editing options:

- Enable - change host status to Monitored
- Disable - change host status to Not monitored
- Export - export the hosts to a YAML, XML or JSON file
- Mass update - update several properties for a number of hosts at once
- Delete - delete the hosts

To use these options, mark the checkboxes before the respective hosts, then click on the required button.

Using filter

You can use the filter to display only the hosts you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available above the list of hosts. If you click on it, a filter becomes available where you can filter hosts by host group, linked templates, name, DNS, IP, port number, if they are monitored by server or by proxy, proxy name and tags.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host groups</td>
<td>Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups.</td>
</tr>
<tr>
<td>Templates</td>
<td>Filter by linked templates.</td>
</tr>
<tr>
<td>Name</td>
<td>Filter by visible host name.</td>
</tr>
<tr>
<td>DNS</td>
<td>Filter by DNS name.</td>
</tr>
<tr>
<td>IP</td>
<td>Filter by IP address.</td>
</tr>
<tr>
<td>Port</td>
<td>Filter by port number.</td>
</tr>
<tr>
<td>Monitored by</td>
<td>Filter hosts that are monitored by server only, proxy only or both.</td>
</tr>
<tr>
<td>Proxy</td>
<td>Filter hosts that are monitored by the proxy specified here.</td>
</tr>
</tbody>
</table>
| Tags          | Filter by host tag name and value. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition:  
**Exists** - include the specified tag names  
**Equals** - include the specified tag names and values (case-sensitive)  
**Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)  
**Does not exist** - exclude the specified tag names  
**Does not equal** - exclude the specified tag names and values (case-sensitive)  
**Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)  
There are two calculation types for conditions:  
**And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition  
**Or** - enough if one condition is met |

**Reading host availability**

Host availability icons reflect the current host interface status on Zabbix server. Therefore, in the frontend:
- If you disable a host, availability icons will not immediately turn gray (unknown status), because the server has to synchronize the configuration changes first;
- If you enable a host, availability icons will not immediately turn green (available), because the server has to synchronize the configuration changes and start polling the host first.

**Unknown interface status**

Zabbix server determines an unknown status for the corresponding agent interface (Zabbix, SNMP, IPMI, JMX) if:
- there are no enabled items on the interface (they were removed or disabled);
- there are only active Zabbix agent items;
- there are no pollers for that type of the interface (e.g. StartPollers=0);
- host is disabled;
- host is set to be monitored by proxy, a different proxy or by server if it was monitored by proxy;
- host is monitored by a proxy that appears to be offline (no updates received from the proxy during the maximum heartbeat interval - 1 hour).

Setting interface availability to unknown is done after server configuration cache synchronization. Restoring interface availability (available/unavailable) on hosts monitored by proxies is done after proxy configuration cache synchronization.

See also more details about host interface unreachability.

1 Items

**Overview**

The item list for a host can be accessed from Configuration → Hosts by clicking on Items for the respective host.

A list of existing items is displayed.
<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
</table>
| Item menu | Click on the three-dot icon to open the menu for the specific item with these options: **Latest data** - see latest data of the item  
**Create trigger** - create a trigger based on this item  
**Triggers** - click to see a list with links to already-configured trigger of this item  
**Create dependent item** - create a dependent item for this item  
**Create dependent discovery rule** - create a dependent discovery rule for this item |
| Host | Host of the item.  
This column is displayed only if multiple hosts are selected in the filter. |
| Name | Name of the item displayed as a blue link to item details.  
Clicking on the item name link opens the item configuration form.  
If the host item belongs to a template, the template name is displayed before the item name as a gray link.  
Clicking on the template link will open the item list on the template level.  
If the item has been created from an item prototype, its name is preceded by the low-level discovery rule name, in orange.  
Clicking on the discovery rule name will open the item prototype list. |
| Triggers | Moving the mouse over Triggers will display an infobox displaying the triggers associated with the item.  
The number of the triggers is displayed in gray. |
| Key | Item key is displayed. |
| Interval | Frequency of the check is displayed.  
Note that passive items can also be checked immediately by pushing the **Execute now** button. |
| History | How many days item data history will be kept is displayed. |
| Trends | How many days item trends history will be kept is displayed. |
| Type | Item type is displayed (Zabbix agent, SNMP agent, simple check, etc). |
| Status | Item status is displayed - Enabled, Disabled or Not supported. You can change the status by clicking on it - from Enabled to Disabled (and back); from Not supported to Disabled (and back). |
| Tags | Item tags are displayed.  
Up to three tags (name:value pairs) can be displayed. If there are more tags, a "..." link is displayed that allows to see all tags on mouseover. |
| Info | If the item is working correctly, no icon is displayed in this column. In case of errors, a square icon with the letter "i" is displayed. Hover over the icon to see a tooltip with the error description. |

To configure a new item, click on the **Create item** button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- Enable - change item status to Enabled
- Disable - change item status to Disabled
- Execute now - execute a check for new item values immediately. Supported for passive checks only (see more details). Note that when checking for values immediately, configuration cache is not updated, thus the values will not reflect very recent changes to item configuration.
- Clear history - delete history and trend data for items.
- Copy - copy the items to other hosts or templates.
- Mass update - update several properties for a number of items at once.
• Delete - delete the items.

To use these options, mark the checkboxes before the respective items, then click on the required button.

Using filter

You can use the filter to display only the items you are interested in. For better search performance, data is searched with macros unresolved.

The Filter icon is available at the top right corner. Clicking on it will open a filter where you can specify the desired filtering criteria.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host groups</td>
<td>Filter by one or more host groups.</td>
</tr>
<tr>
<td></td>
<td>Specifying a parent host group implicitly selects all nested host groups.</td>
</tr>
<tr>
<td></td>
<td>Host groups containing templates only cannot be selected.</td>
</tr>
<tr>
<td>Hosts</td>
<td>Filter by one or more hosts.</td>
</tr>
<tr>
<td>Name</td>
<td>Filter by item name.</td>
</tr>
<tr>
<td>Key</td>
<td>Filter by item key.</td>
</tr>
<tr>
<td>Value mapping</td>
<td>Filter by the value map used.</td>
</tr>
<tr>
<td></td>
<td>This parameter is not displayed if the Hosts option is empty.</td>
</tr>
<tr>
<td>Type</td>
<td>Filter by item type (Zabbix agent, SNMP agent, etc.).</td>
</tr>
<tr>
<td>Type of information</td>
<td>Filter by type of information (Numeric unsigned, float, etc.).</td>
</tr>
<tr>
<td>History</td>
<td>Filter by how long item history is kept.</td>
</tr>
<tr>
<td>Trends</td>
<td>Filter by how long item trends are kept.</td>
</tr>
<tr>
<td>Update interval</td>
<td>Filter by item update interval.</td>
</tr>
</tbody>
</table>
Parameter | Description
--- | ---
Tags | Specify tags to limit the number of items displayed. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive.

There are several operators available for each condition:
- **Exists** - include the specified tag names
- **Equals** - include the specified tag names and values (case-sensitive)
- **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)
- **Does not exist** - exclude the specified tag names
- **Does not equal** - exclude the specified tag names and values (case-sensitive)
- **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)

There are two calculation types for conditions:
- **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition
- **Or** - enough if one condition is met

State | Filter by item state - Normal or Not supported.

Status | Filter by item status - Enabled or Disabled.

Triggers | Filter items with (or without) triggers.

Inherited | Filter items inherited (or not inherited) from a template.

Discovery | Filter items discovered (or not discovered) by low-level discovery.

The **Subfilter** below the filter offers further filtering options (for the data already filtered). You can select groups of items with a common parameter value. Upon clicking on a group, it gets highlighted and only the items with this parameter value remain in the list.

2 Triggers

Overview

The trigger list for a host can be accessed from Configuration → Hosts by clicking on Triggers for the respective host.

**Displayed data:**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
</table>
| Severity | Severity of the trigger is displayed by both name and cell background color. Trigger value is displayed:  
**OK** - the trigger is in the OK state  
**PROBLEM** - the trigger is in the Problem state |
| Value | Host of the trigger. This column is displayed only if multiple hosts are selected in the filter. |
### Column Description

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Name of the trigger, displayed as a blue link to trigger details.</td>
</tr>
<tr>
<td></td>
<td>Clicking on the trigger name link opens the trigger <a href="#">configuration form</a>.</td>
</tr>
<tr>
<td></td>
<td>If the host trigger belongs to a template, the template name is displayed before the trigger name, as a gray link. Clicking on the template link will open the trigger list on the template level.</td>
</tr>
<tr>
<td></td>
<td>If the trigger has been created from a trigger prototype, its name is preceded by the low-level discovery rule name, in orange. Clicking on the discovery rule name will open the trigger prototype list.</td>
</tr>
<tr>
<td><strong>Operational data</strong></td>
<td>Operational data definition of the trigger, containing arbitrary strings and macros that will resolve dynamically in Monitoring → Problems.</td>
</tr>
<tr>
<td><strong>Expression</strong></td>
<td>Trigger expression is displayed. The host-item part of the expression is displayed as a link, leading to the item configuration form.</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Trigger status is displayed - Enabled, Disabled or Unknown. By clicking on the status you can change it - from Enabled to Disabled (and back); from Unknown to Disabled (and back). Problems of a disabled trigger are no longer displayed in the frontend, but are not deleted.</td>
</tr>
<tr>
<td><strong>Info</strong></td>
<td>If everything is working correctly, no icon is displayed in this column. In case of errors, a square icon with the letter &quot;i&quot; is displayed. Hover over the icon to see a tooltip with the error description.</td>
</tr>
<tr>
<td><strong>Tags</strong></td>
<td>If a trigger contains tags, tag name and value are displayed in this column.</td>
</tr>
</tbody>
</table>

To configure a new trigger, click on the Create trigger button at the top right corner.

**Mass editing options**

Buttons below the list offer some mass-editing options:

- Enable - change trigger status to Enabled.
- Disable - change trigger status to Disabled.
- Copy - copy the triggers to other hosts or templates.
- Mass update - update several properties for a number of triggers at once.
- Delete - delete the triggers.

To use these options, mark the checkboxes before the respective triggers, then click on the required button.

**Using filter**

You can use the filter to display only the triggers you are interested in. For better search performance, data is searched with macros unresolved.

The Filter icon is available at the top right corner. Clicking on it will open a filter where you can specify the desired filtering criteria.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host groups</strong></td>
<td>Filter by one or more host groups.</td>
</tr>
<tr>
<td></td>
<td>Specifying a parent host group implicitly selects all nested host groups.</td>
</tr>
<tr>
<td></td>
<td>Host groups containing templates only cannot be selected.</td>
</tr>
<tr>
<td><strong>Hosts</strong></td>
<td>Filter by one or more hosts.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>If host groups are already selected above, host selection is limited to those groups.</td>
</tr>
<tr>
<td><strong>Severity</strong></td>
<td>Filter by trigger name.</td>
</tr>
<tr>
<td><strong>State</strong></td>
<td>Filter by trigger state.</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Filter by trigger status.</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>Filter by trigger value.</td>
</tr>
</tbody>
</table>
### Tags

Filter by trigger tag name and value. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive.

There are several operators available for each condition:

- **Exists** - include the specified tag names
- **Equals** - include the specified tag names and values (case-sensitive)
- **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive)
- **Does not exist** - exclude the specified tag names
- **Does not equal** - exclude the specified tag names and values (case-sensitive)
- **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive)

There are two calculation types for conditions:

- **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition
- **Or** - enough if one condition is met

Macros and **macro functions** are supported both in tag name and tag value fields.

### Inherited

Filter triggers inherited (or not inherited) from a template.

### Discovered

Filter triggers discovered (or not discovered) by low-level discovery.

### With dependencies

Filter triggers with (or without) dependencies.

---

### 3 Graphs

#### Overview

The custom graph list for a host can be accessed from Configuration → Hosts by clicking on Graphs for the respective host.

A list of existing graphs is displayed.

#### Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the custom graph, displayed as a blue link to graph details. If the host graph belongs to a template, the template name is displayed before the graph name, as a gray link. Clicking on the template link will open the graph list on the template level. If the graph has been created from a graph prototype, its name is preceded by the low-level discovery rule name, in orange. Clicking on the discovery rule name will open the graph prototype list.</td>
</tr>
<tr>
<td>Width</td>
<td>Graph width is displayed.</td>
</tr>
<tr>
<td>Height</td>
<td>Graph height is displayed.</td>
</tr>
<tr>
<td>Graph type</td>
<td>Graph type is displayed - Normal, Stacked, Pie or Exploded.</td>
</tr>
<tr>
<td>Info</td>
<td>If the graph is working correctly, no icon is displayed in this column. In case of errors, a square icon with the letter “i” is displayed. Hover over the icon to see a tooltip with the error description.</td>
</tr>
</tbody>
</table>
To configure a new graph, click on the Create graph button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- **Copy**: copy the graphs to other hosts or templates
- **Delete**: delete the graphs

To use these options, mark the checkboxes before the respective graphs, then click on the required button.

Using filter

You can filter graphs by host group and host. For better search performance, data is searched with macros unresolved.

**4 Discovery rules**

**Overview**

The list of low-level discovery rules for a host can be accessed from Configuration → Hosts by clicking on Discovery for the respective host.

A list of existing low-level discovery rules is displayed. It is also possible to see all discovery rules independently of the host, or all discovery rules of a specific host group by changing the filter settings.

**Displayed data:**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>The visible host name is displayed. In the absence of a visible host name, the technical host name is displayed.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the rule, displayed as a blue link. Clicking on the rule name opens the low-level discovery rule configuration form. If the discovery rule belongs to a template, the template name is displayed before the rule name, as a gray link. Clicking on the template link will open the rule list on the template level.</td>
</tr>
<tr>
<td>Items</td>
<td>A link to the list of item prototypes is displayed. The number of existing item prototypes is displayed in gray.</td>
</tr>
<tr>
<td>Triggers</td>
<td>A link to the list of trigger prototypes is displayed. The number of existing trigger prototypes is displayed in gray.</td>
</tr>
<tr>
<td>Graphs</td>
<td>A link to the list of graph prototypes is displayed. The number of existing graph prototypes is displayed in gray.</td>
</tr>
<tr>
<td>Hosts</td>
<td>A link to the list of host prototypes is displayed. The number of existing host prototypes is displayed in gray.</td>
</tr>
<tr>
<td>Key</td>
<td>The item key used for discovery is displayed.</td>
</tr>
<tr>
<td>Interval</td>
<td>The frequency of performing discovery is displayed. Note that discovery can also be performed immediately by pushing the Execute now button below the list.</td>
</tr>
<tr>
<td>Type</td>
<td>The item type used for discovery is displayed (Zabbix agent, SNMP agent, etc).</td>
</tr>
<tr>
<td>Status</td>
<td>Discovery rule status is displayed - Enabled, Disabled or Not supported. By clicking on the status you can change it - from Enabled to Disabled (and back); from Not supported to Disabled (and back).</td>
</tr>
<tr>
<td>Info</td>
<td>If everything is fine, no icon is displayed in this column. In case of errors, a square icon with the letter “i” is displayed. Hover over the icon to see a tooltip with the error description.</td>
</tr>
</tbody>
</table>

To configure a new low-level discovery rule, click on the Create discovery rule button at the top right corner.
Mass editing options

Buttons below the list offer some mass-editing options:

- Enable - change the low-level discovery rule status to Enabled.
- Disable - change the low-level discovery rule status to Disabled.
- Execute now - perform discovery based on the discovery rules immediately. See more details. Note that when performing discovery immediately, the configuration cache is not updated, thus the result will not reflect very recent changes to discovery rule configuration.
- Delete - delete the low-level discovery rules.

To use these options, mark the checkboxes before the respective discovery rules, then click on the required button.

Using filter

You can use the filter to display only the discovery rules you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available above the list of discovery rules. If you click on it, a filter becomes available where you can filter discovery rules by host group, host, name, item key, item type, and other parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host groups</td>
<td>Filter by one or more host groups.</td>
</tr>
<tr>
<td></td>
<td>Specifying a parent host group implicitly selects all nested host groups.</td>
</tr>
<tr>
<td>Hosts</td>
<td>Filter by one or more hosts.</td>
</tr>
<tr>
<td>Name</td>
<td>Filter by discovery rule name.</td>
</tr>
<tr>
<td>Key</td>
<td>Filter by discovery item key.</td>
</tr>
<tr>
<td>Type</td>
<td>Filter by discovery item type.</td>
</tr>
<tr>
<td>Update interval</td>
<td>Filter by update interval.</td>
</tr>
<tr>
<td></td>
<td>Not available for Zabbix trapper and dependent items.</td>
</tr>
<tr>
<td>Keep lost resources period</td>
<td>Filter by Keep lost resources period.</td>
</tr>
<tr>
<td>SNMP OID</td>
<td>Filter by SNMP OID.</td>
</tr>
<tr>
<td></td>
<td>Only available if SNMP agent is selected as type.</td>
</tr>
<tr>
<td>State</td>
<td>Filter by discovery rule state (All/Normal/Not supported).</td>
</tr>
<tr>
<td>Status</td>
<td>Filter by discovery rule status (All/Enabled/Disabled).</td>
</tr>
</tbody>
</table>

1 Item prototypes

Overview

In this section the item prototypes of a low-level discovery rule on the host are displayed. Item prototypes are the basis of real host items that are created during low-level discovery.
Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the item prototype, displayed as a blue link. Clicking on the name opens the item prototype configuration form. If the item prototype belongs to a template, the template name is displayed before the rule name, as a gray link. Clicking on the template link will open the item prototype list on the template level.</td>
</tr>
<tr>
<td>Key</td>
<td>Key of the item prototype is displayed.</td>
</tr>
<tr>
<td>Interval</td>
<td>Frequency of the check is displayed.</td>
</tr>
<tr>
<td>History</td>
<td>How many days to keep item data history is displayed.</td>
</tr>
<tr>
<td>Trends</td>
<td>How many days to keep item trends history is displayed.</td>
</tr>
<tr>
<td>Type</td>
<td>Type of the item prototype is displayed (Zabbix agent, SNMP agent, simple check, etc.).</td>
</tr>
</tbody>
</table>
| Create enabled          | Create the item based on this prototype as: Yes - enabled  
|                         | No - disabled. You can switch between 'Yes' and 'No' by clicking on them. |
| Discover                | Discover the item based on this prototype: Yes - discover  
|                         | No - do not discover. You can switch between 'Yes' and 'No' by clicking on them. |
| Tags                    | Tags of the item prototype are displayed. |

To configure a new item prototype, click on the Create item prototype button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- Create enabled - create these items as Enabled
- Create disabled - create these items as Disabled
- Mass update - mass update these item prototypes
- Delete - delete these item prototypes

To use these options, mark the checkboxes before the respective item prototypes, then click on the required button.

2 Trigger prototypes

Overview

In this section the trigger prototypes of a low-level discovery rule on the host are displayed. Trigger prototypes are the basis of real host triggers that are created during low-level discovery.
Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the trigger prototype, displayed as a blue link. Clicking on the name opens the trigger prototype configuration form. If the trigger prototype belongs to a linked template, the template name is displayed before the trigger name, as a gray link. Clicking on the template link will open the trigger prototype list on the linked template level.</td>
</tr>
<tr>
<td>Operational data</td>
<td>Format of the operational data of the trigger is displayed, containing arbitrary strings and macros that will resolve dynamically in Monitoring → Problems. Create the trigger based on this prototype as: Yes - enabled No - disabled. You can switch between ‘Yes’ and ‘No’ by clicking on them. Discover the trigger based on this prototype: Yes - discover No - do not discover. You can switch between ‘Yes’ and ‘No’ by clicking on them.</td>
</tr>
<tr>
<td>Tags</td>
<td>Tags of the trigger prototype are displayed. To use these options, mark the checkboxes before the respective trigger prototypes, then click on the required button.</td>
</tr>
</tbody>
</table>

To configure a new trigger prototype, click on the Create trigger prototype button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- Create enabled - create these triggers as Enabled
- Create disabled - create these triggers as Disabled
- Mass update - mass update these trigger prototypes
- Delete - delete these trigger prototypes

To use these options, mark the checkboxes before the respective trigger prototypes, then click on the required button.

3 Graph prototypes

Overview

In this section the graph prototypes of a low-level discovery rule on the host are displayed. Graph prototypes are the basis of real host graphs that are created during low-level discovery.
Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the graph prototype, displayed as a blue link. Clicking on the name opens the graph prototype configuration form. If the graph prototype belongs to a linked template, the template name is displayed before the graph name, as a gray link. Clicking on the template link will open the graph prototype list on the linked template level.</td>
</tr>
<tr>
<td>Width</td>
<td>Width of the graph prototype is displayed.</td>
</tr>
<tr>
<td>Height</td>
<td>Height of the graph prototype is displayed.</td>
</tr>
<tr>
<td>Type</td>
<td>Type of the graph prototype is displayed - Normal, Stacked, Pie or Exploded.</td>
</tr>
<tr>
<td>Discover</td>
<td>Discover the graph based on this prototype: <strong>Yes</strong> - discover  <strong>No</strong> - do not discover. You can switch between ‘Yes’ and ‘No’ by clicking on them.</td>
</tr>
</tbody>
</table>

To configure a new graph prototype, click on the Create graph prototype button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- **Delete** - delete these graph prototypes

To use these options, mark the checkboxes before the respective graph prototypes, then click on the required button.

4 Host prototypes

Overview

In this section the host prototypes of a low-level discovery rule on the host are displayed. Host prototypes are the basis of real hosts that are created during low-level discovery.

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the host prototype, displayed as a blue link. Clicking on the name opens the host prototype configuration form. If the host prototype belongs to a linked template, the template name is displayed before the host name, as a gray link. Clicking on the template link will open the host prototype list on the linked template level.</td>
</tr>
<tr>
<td>Templates</td>
<td>Templates of the host prototype are displayed.</td>
</tr>
<tr>
<td>Create enabled</td>
<td>Create the host based on this prototype as: <strong>Yes</strong> - enabled  <strong>No</strong> - disabled. You can switch between ‘Yes’ and ‘No’ by clicking on them.</td>
</tr>
<tr>
<td>Discover</td>
<td>Discover the host based on this prototype: <strong>Yes</strong> - discover  <strong>No</strong> - do not discover. You can switch between ‘Yes’ and ‘No’ by clicking on them.</td>
</tr>
</tbody>
</table>
To configure a new host prototype, click on the Create host prototype button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:
- Create enabled - create these hosts as Enabled
- Create disabled - create these hosts as Disabled
- Delete - delete these host prototypes

To use these options, mark the checkboxes before the respective host prototypes, then click on the required button.

5 Web scenarios

Overview

The web scenario list for a host can be accessed from Configuration → Hosts by clicking on Web for the respective host.

A list of existing web scenarios is displayed.

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the web scenario. Clicking on the web scenario name opens the web scenario configuration form. If the host web scenario belongs to a template, the template name is displayed before the web scenario name as a gray link. Clicking on the template link will open the web scenario list on the template level.</td>
</tr>
<tr>
<td>Number of steps</td>
<td>The number of steps the scenario contains.</td>
</tr>
<tr>
<td>Update interval</td>
<td>How often the scenario is performed.</td>
</tr>
<tr>
<td>Attempts</td>
<td>How many attempts for executing web scenario steps are performed.</td>
</tr>
<tr>
<td>Authentication</td>
<td>Authentication method is displayed - Basic, NTLM, or None.</td>
</tr>
<tr>
<td>HTTP proxy</td>
<td>Displays HTTP proxy or 'No' if not used.</td>
</tr>
<tr>
<td>Status</td>
<td>Web scenario status is displayed - Enabled or Disabled. By clicking on the status you can change it.</td>
</tr>
<tr>
<td>Tags</td>
<td>Web scenario tags are displayed. Up to three tags (name:value pairs) can be displayed. If there are more tags, a &quot;...&quot; link is displayed that allows to see all tags on mouseover.</td>
</tr>
<tr>
<td>Info</td>
<td>If everything is working correctly, no icon is displayed in this column. In case of errors, a square icon with the letter &quot;i&quot; is displayed. Hover over the icon to see a tooltip with the error description.</td>
</tr>
</tbody>
</table>

To configure a new web scenario, click on the Create web scenario button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:
- Enable - change the scenario status to Enabled
- Disable - change the scenario status to Disabled
- Clear history - clear history and trend data for the scenarios
- Delete - delete the web scenarios
To use these options, mark the checkboxes before the respective web scenarios, then click on the required button.

Using filter

You can use the filter to display only the scenarios you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available above the list of web scenarios. If you click on it, a filter becomes available where you can filter scenarios by host group, host, status and tags.

5 Maintenance

Overview

In the Configuration → Maintenance section users can configure and maintain maintenance periods for hosts.

A listing of existing maintenance periods with their details is displayed.

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the maintenance period. Clicking on the maintenance period name opens the maintenance period configuration form.</td>
</tr>
<tr>
<td>Type</td>
<td>The type of maintenance is displayed: With data collection or No data collection</td>
</tr>
<tr>
<td>Active since</td>
<td>The date and time when executing maintenance periods becomes active. Note: This time does not activate a maintenance period; maintenance periods need to be set separately.</td>
</tr>
<tr>
<td>Active till</td>
<td>The date and time when executing maintenance periods stops being active.</td>
</tr>
<tr>
<td>State</td>
<td>The state of the maintenance period:</td>
</tr>
<tr>
<td></td>
<td>Approaching - will become active soon</td>
</tr>
<tr>
<td></td>
<td>Active - is active</td>
</tr>
<tr>
<td></td>
<td>Expired - is not active any more</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the maintenance period is displayed.</td>
</tr>
</tbody>
</table>

To configure a new maintenance period, click on the Create maintenance period button in the top right-hand corner.

Mass editing options

A button below the list offers one mass-editing option:

- Delete - delete the maintenance periods

To use this option, mark the checkboxes before the respective maintenance periods and click on Delete.

Using filter

You can use the filter to display only the maintenance periods you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available above the list of maintenance periods. If you click on it, a filter becomes available where you can filter maintenance periods by host group, name and state.
6 Actions

Overview

In the Configuration → Actions section users can configure and maintain actions.

The actions displayed are actions assigned to the selected event source (trigger, discovery, autoregistration, internal actions).

Actions are grouped into subsections by event source (trigger, service, discovery, autoregistration, internal actions). The list of available subsections appears upon pressing on Actions in the Configuration menu section. It is also possible to switch between subsections by using a title dropdown in the top left corner.

After selecting a subsection, a page with a list of existing actions with their details will be displayed.

For users without Super admin rights actions are displayed according to permission settings. That means in some cases a user without Super admin rights isn’t able to view the complete action list because of certain permission restrictions. An action is displayed to the user without Super admin rights if the following conditions are fulfilled:

- The user has read-write access to host groups, hosts, templates, and triggers in action conditions
- The user has read-write access to host groups, hosts, and templates in action operations, recovery operations, and update operations
- The user has read access to user groups and users in action operations, recovery operations, and update operations

Actions for services are maintained in a similar way in the Services → Service actions menu section. User’s access to specific service actions depends on the user role permissions set in the Access to services menu section.

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the action. Clicking on the action name opens the action configuration form.</td>
</tr>
<tr>
<td>Conditions</td>
<td>Action conditions are displayed.</td>
</tr>
<tr>
<td>Operations</td>
<td>Action operations are displayed. Since Zabbix 2.2, the operation list also displays the media type (e-mail, SMS or script) used for notification as well as the name and surname (in parentheses after the username) of a notification recipient. Action operation can both be a notification or a remote command depending on the selected type of operation.</td>
</tr>
<tr>
<td>Status</td>
<td>Action status is displayed - Enabled or Disabled. By clicking on the status you can change it. See the Escalations section for more details as to what happens if an action is disabled during an escalation in progress.</td>
</tr>
</tbody>
</table>

To configure a new action, click on the Create action button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- Enable - change the action status to Enabled
- Disable - change the action status to Disabled
- Delete - delete the actions
To use these options, mark the checkboxes before the respective actions, then click on the required button.

Using filter

You can use the filter to display only the actions you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available above the list of actions. If you click on it, a filter becomes available where you can filter actions by name and status.

7 Event correlation

Overview

In the Configuration → Event correlation section users can configure and maintain global correlation rules for Zabbix events.

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the correlation rule. Clicking on the correlation rule name opens the rule configuration form.</td>
</tr>
<tr>
<td>Conditions</td>
<td>Correlation rule conditions are displayed.</td>
</tr>
<tr>
<td>Operations</td>
<td>Correlation rule operations are displayed.</td>
</tr>
<tr>
<td>Status</td>
<td>Correlation rule status is displayed - Enabled or Disabled. By clicking on the status you can change it.</td>
</tr>
</tbody>
</table>

To configure a new correlation rule, click on the Create correlation button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- Enable - change the correlation rule status to Enabled
- Disable - change the correlation rule status to Disabled
- Delete - delete the correlation rules

To use these options, mark the checkboxes before the respective correlation rules, then click on the required button.

Using filter

You can use the filter to display only the correlation rules you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available above the list of correlation rules. If you click on it, a filter becomes available where you can filter correlation rules by name and status.
8 Discovery

Overview

In the Configuration → Discovery section users can configure and maintain discovery rules. A listing of existing discovery rules with their details is displayed.

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the discovery rule. Clicking on the discovery rule name opens the discovery rule configuration form.</td>
</tr>
<tr>
<td>IP range</td>
<td>The range of IP addresses to use for network scanning is displayed.</td>
</tr>
<tr>
<td>Proxy</td>
<td>The proxy name is displayed, if discovery is performed by the proxy.</td>
</tr>
<tr>
<td>Interval</td>
<td>The frequency of performing discovery displayed.</td>
</tr>
<tr>
<td>Checks</td>
<td>The types of checks used for discovery are displayed.</td>
</tr>
<tr>
<td>Status</td>
<td>Action status is displayed - Enabled or Disabled.</td>
</tr>
<tr>
<td></td>
<td>By clicking on the status you can change it.</td>
</tr>
</tbody>
</table>

To configure a new discovery rule, click on the Create discovery rule button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- Enable - change the discovery rule status to Enabled
- Disable - change the discovery rule status to Disabled
- Delete - delete the discovery rules

To use these options, mark the checkboxes before the respective discovery rules, then click on the required button.

Using filter

You can use the filter to display only the discovery rules you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available above the list of discovery rules. If you click on it, a filter becomes available where you can filter discovery rules by name and status.

6 Administration

Overview

The Administration menu is for administrative functions of Zabbix. This menu is available to users of Super Administrators type only.

1 General

Overview
The Administration → General section contains a number of subsections for setting frontend-related defaults and customizing Zabbix.

The list of available subsections appears upon pressing on General in the Administration menu section. It is also possible to switch between subsections by using the title dropdown in the top left corner.

1 GUI

This section provides customization of several frontend-related defaults.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default language</td>
<td>Default language for users who have not specified a language in their profiles and guest users. For more information, see <a href="#">Installation of additional frontend languages</a>.</td>
</tr>
<tr>
<td>Default time zone</td>
<td>Default time zone for users who have not specified a time zone in their profiles and guest users.</td>
</tr>
<tr>
<td>Default theme</td>
<td>Default theme for users who have not specified a theme in their profiles and guest users.</td>
</tr>
<tr>
<td>Limit for search and filter results</td>
<td>Maximum amount of elements (rows) that will be displayed in a web-interface list, for example, in Configuration → Hosts. Note: If set to, for example, '50', only the first 50 elements will be displayed in all affected frontend lists. If some list contains more than fifty elements, the indication of that will be the '+' sign in &quot;Displaying 1 to 50 of 50+ found&quot;. Also, if filtering is used and still there are more than 50 matches, only the first 50 will be displayed.</td>
</tr>
<tr>
<td>Max number of columns and rows in overview tables</td>
<td>Maximum number of columns and rows to display in Data overview and Trigger overview dashboard widgets. The same limit applies to both columns and rows. If more rows and/or columns than shown exist, the system will display a warning at the bottom of the table: &quot;Not all results are displayed. Please provide more specific search criteria.&quot;</td>
</tr>
<tr>
<td>Max count of elements to show inside table cell</td>
<td>For entries that are displayed in a single table cell, no more than configured here will be shown.</td>
</tr>
</tbody>
</table>
Parameter | Description
--- | ---
Show warning if Zabbix server is down | This parameter enables a warning message to be displayed in a browser window if the Zabbix server cannot be reached (possibly down). The message remains visible even if the user scrolls down the page. When hovered over, the message is temporarily hidden to reveal the contents underneath it.
   This parameter is supported since Zabbix 2.0.1.
Working time | This system-wide parameter defines working hours. In graphs, working time is displayed as a white background and non-working time is displayed as gray.
   See [Time period specification](#) page for description of the time format.
   User macros are supported (since Zabbix 3.4.0).
Show technical errors | If checked, all registered users will be able to see technical errors (PHP/SQL). If unchecked, the information is only available to Zabbix Super Admins and users belonging to the user groups with enabled debug mode.
Max history display period | Maximum time period for which to display historical data in Monitoring subsections: Latest data, Web, and in the Data overview dashboard widget.
   Allowed range: 24 hours (default) - 1 week. Time suffixes, e.g. 1w (one week), 36h (36 hours), are supported.
Time filter default period | Time period to be used in graphs and dashboards by default. Allowed range: 1 minute - 10 years (default: 1 hour).
   Time suffixes, e.g. 10m (ten minutes), 5w (five weeks), are supported.
   Note: when a user changes the time period while viewing a graph, this time period is stored as user preference, replacing the global default or a previous user selection.
Max period for time selector | Maximum available time period for graphs and dashboards. Users will not be able to visualize older data. Allowed range: 1 year - 10 years (default: 2 years).
   Time suffixes, e.g. 1y (one year), 365w (365 weeks), are supported.

### Autoregistration

In this section, you can configure the encryption level for active agent autoregistration.

Parameters marked with an asterisk are mandatory.

**Configuration parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| Encryption level | Select one or both options for encryption level:
   **No encryption** - unencrypted connections are allowed
   **PSK** - TLS encrypted connections with a pre-shared key are allowed
| PSK identity | Enter the pre-shared key identity string.
   This field is only available if ‘PSK’ is selected as Encryption level.
   Do not put sensitive information in the PSK identity, it is transmitted unencrypted over the network to inform a receiver which PSK to use.
| PSK | Enter the pre-shared key (an even number of hexadecimal characters).
   Maximum length: 512 hex-digits (256-byte PSK) if Zabbix uses GnuTLS or OpenSSL library, 64 hex-digits (32-byte PSK) if Zabbix uses mbed TLS (PolarSSL) library.
   Example: 1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952
   This field is only available if ‘PSK’ is selected as Encryption level.
The housekeeper is a periodical process, executed by Zabbix server. The process removes outdated information and information deleted by user.

In this section housekeeping tasks can be enabled or disabled on a per-task basis separately for: events and alerts/IT services/user sessions/history/trends. Audit housekeeping settings are available in a separate menu section.

If housekeeping is enabled, it is possible to set for how many days data records will be kept before being removed by the housekeeper.
Deleting an item/trigger will also delete problems generated by that item/trigger.

Also, an event will only be deleted by the housekeeper if it is not associated with a problem in any way. This means that if an event is either a problem or recovery event, it will not be deleted until the related problem record is removed. The housekeeper will delete problems first and events after, to avoid potential problems with stale events or problem records.

For history and trends an additional option is available: Override item history period and Override item trend period. This option allows to globally set for how many days item history/trends will be kept (1 hour to 25 years; or "0"), in this case overriding the values set for individual items in History storage period/Trend storage period fields in item configuration. Note, that the storage period will not be overridden for items that have configuration option Do not keep history and/or Do not keep trends enabled.

It is possible to override the history/trend storage period even if internal housekeeping is disabled. Thus, when using an external housekeeper, the history storage period could be set using the history Data storage period field.

If using TimescaleDB, in order to take full advantage of TimescaleDB automatic partitioning of history and trends tables, Override item history period and Override item trend period options must be enabled as well as Enable internal housekeeping option for history and trends. Otherwise, the data kept in these tables will still be stored in partitions, however, the housekeeper will be cleaning the history and trends by deleting individual records rather than by dropping outdated partitions. When dropping of outdated partitions is enabled, Zabbix server and frontend will no longer keep track of deleted items and history for deleted items will be cleared when an outdated partition is deleted.

Time suffixes are supported in the period fields, e.g. 1d (one day), 1w (one week). The minimum is 1 day (1 hour for history), the maximum - 25 years.

Reset defaults button allows to revert any changes made.

4 Audit log
This section allows configuring audit log settings.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable audit logging</td>
<td>Enable/disable audit logging. Marked by default.</td>
</tr>
<tr>
<td>Enable internal housekeeping</td>
<td>Enable/disable internal housekeeping for audit. Marked by default.</td>
</tr>
<tr>
<td>Data storage period</td>
<td>Amount of days audit records should be kept for before being removed by the housekeeper. Mandatory if housekeeping is enabled. Default: 365 days.</td>
</tr>
</tbody>
</table>

5 Images
The Images section displays all the images available in Zabbix. Images are stored in the database.
The Type dropdown allows you to switch between icon and background images:

- Icons are used to display network map elements
- Backgrounds are used as background images of network maps

**Adding image**

You can add your own image by clicking on the Create icon or Create background button in the top right corner.

---

**Image attributes:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Unique name of an image.</td>
</tr>
<tr>
<td>Upload</td>
<td>Select the file (PNG, JPEG, GIF) from a local system to be uploaded to Zabbix. Note that it may be possible to upload other formats that will be converted to PNG during upload. GD library is used for image processing, therefore formats that are supported depend on the library version used (2.0.28 or higher is required by Zabbix).</td>
</tr>
</tbody>
</table>

Maximum size of the upload file is limited by the value of ZBX_MAX_IMAGE_SIZE that is 1024x1024 bytes or 1 MB.

The upload of an image may fail if the image size is close to 1 MB and the max_allowed_packet MySQL configuration parameter is at a default of 1MB. In this case, increase the max_allowed_packet parameter.

**6 Icon mapping**

This section allows creating the mapping of certain hosts with certain icons. Host inventory field information is used to create the mapping.

The mappings can then be used in network map configuration to assign appropriate icons to matching hosts automatically.

To create a new icon map, click on Create icon map in the top right corner.
Configuration parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Unique name of icon map.</td>
</tr>
<tr>
<td>Mappings</td>
<td>A list of mappings. The order of mappings determines which one will have priority. You can move mappings up and down the list with drag-and-drop.</td>
</tr>
<tr>
<td>Inventory field</td>
<td>Host inventory field that will be looked into to seek a match.</td>
</tr>
<tr>
<td>Expression</td>
<td>Regular expression describing the match.</td>
</tr>
<tr>
<td>Icon</td>
<td>Icon to use if a match for the expression is found.</td>
</tr>
<tr>
<td>Default</td>
<td>Default icon to use.</td>
</tr>
</tbody>
</table>

7 Regular expressions

This section allows creating custom regular expressions that can be used in several places in the frontend. See Regular expressions section for details.

8 Macros

This section allows to define system-wide user macros as name-value pairs. Note that macro values can be kept as plain text, secret text or Vault secret. Adding a description is also supported.

9 Trigger displaying options

This section allows customizing how trigger status is displayed in the frontend and trigger severity names and colors.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use custom event status colors</td>
<td>Checking this parameter turns on the customization of colors for acknowledged/unacknowledged problems.</td>
</tr>
<tr>
<td>Unacknowledged PROBLEM events, Acknowledged PROBLEM events, Unacknowledged RESOLVED events, Acknowledged RESOLVED events</td>
<td>Enter new color code or click on the color to select a new one from the provided palette. If blinking checkbox is marked, triggers will blink for some time upon the status change to become more visible.</td>
</tr>
<tr>
<td>Display OK triggers for</td>
<td>Time period for displaying OK triggers. Allowed range: 0 - 24 hours. <strong>Time suffixes</strong>, e.g. 5m, 2h, 1d, are supported.</td>
</tr>
<tr>
<td>On status change triggers blink for</td>
<td>Length of trigger blinking. Allowed range: 0 - 24 hours. <strong>Time suffixes</strong>, e.g. 5m, 2h, 1d, are supported.</td>
</tr>
<tr>
<td>Not classified</td>
<td>Custom severity names and/or colors to display instead of system default. Enter new color code or click on the color to select a new one from the provided palette.</td>
</tr>
<tr>
<td>Information</td>
<td>Note, that custom severity names entered here will be used in all locales. If you need to translate them to other languages for certain users, see Customizing trigger severities page.</td>
</tr>
<tr>
<td>Warning, Average, High, Disaster</td>
<td></td>
</tr>
</tbody>
</table>
This section allows selecting geographical map tile service provider and configuring service provider settings for the Geomap dashboard widget. To provide visualization using the geographical maps, Zabbix uses open-source JavaScript interactive maps library Leaflet. Please note that Zabbix has no control over the quality of images provided by third-party tile providers, including the predefined tile providers.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tile provider</td>
<td>Select one of the available tile service providers or select Other to add another tile provider or self-hosted tiles (see Using a custom tile service provider).</td>
</tr>
</tbody>
</table>
| Tile URL        | The URL template for loading and displaying the tile layer on geographical maps. This field is editable only if Tile provider is set to Other. The following placeholders are supported:  
{s} represents one of the available subdomains;  
{z} represents zoom level parameter in the URL;  
{x} and (y) represent tile coordinates;  
{r} can be used to add "@2x" to the URL to load retina tiles.  
Example: https://{s}.example.com/{z}/{x}/{y}{r}.png                                                                                       |
| Attribution     | Tile provider attribution data to be displayed in a small text box on the map. This field is editable only if Tile provider is set to Other.                                                                  |
| Max zoom level  | Maximum zoom level of the map. This field is editable only if Tile provider is set to Other.                                                                                                               |

Using a custom tile service provider

The Geomap widget is capable to load raster tile images from a custom self-hosted or a third-party tile provider service. To use a custom third-party tile provider service or a self-hosted tile folder or server, select Other in the Tile provider field and specify the custom URL in the Tile URL field using proper placeholders.

11 Modules

This section allows to administer custom frontend modules.

Click on Scan directory to register/unregister any custom modules. Registered modules will appear in the list, along with their details. Unregistered modules will be removed from the list.

You may filter modules by name or status (enabled/disabled). Click on the module status in the list to enable/disable a module. You may also mass enable/disable modules by selecting them in the list and then clicking on the Enable/Disable buttons below the list.
This section allows to create and manage API tokens.

You may filter API tokens by name, users to whom the tokens are assigned, expiry date, users that created tokens, or status (enabled/disabled). Click on the token status in the list to quickly enable/disable a token. You may also mass enable/disable tokens by selecting them in the list and then clicking on the Enable/Disable buttons below the list.

To create a new token, press Create API token button at the top right corner, then fill out the required fields in the token configuration screen:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Token’s visible name.</td>
</tr>
<tr>
<td>User</td>
<td>User the token should be assigned to. To quickly select a user, start typing the username, first or last name, then select the required user from the auto-complete list. Alternatively, you can press the Select button and select a user from the full user list. A token can be assigned only to one user.</td>
</tr>
<tr>
<td>Description</td>
<td>Optional token description.</td>
</tr>
<tr>
<td>Set expiration date and time</td>
<td>Unmark this checkbox if a token should not have an expiry date.</td>
</tr>
<tr>
<td>Expiry date</td>
<td>Click on the calendar icon to select token expiry date or enter the date manually in a format YYYY-MM-DD hh:mm:ss</td>
</tr>
<tr>
<td>Enabled</td>
<td>Unmark this checkbox if you need to create a token in a disabled state.</td>
</tr>
</tbody>
</table>

Press Add to create a token. On the next screen, copy and save in a safe place Auth token value **before closing the page**, then press Close. The token will appear in the list.

Auth token value cannot be viewed again later. It is only available immediately after creating a token. If you lose a saved token you will have to regenerate it and doing so will create a new authorization string.

Click on the token name to edit the name, description, expiry date settings, or token status. Note, that it is not possible to change to which user the token is assigned. Press Update button to save changes. If a token has been lost or exposed, you may press
Regenerate button to generate new token value. A confirmation dialog box will appear, asking you to confirm this operation since after proceeding the previously generated token will become invalid.

Users without access to the Administration menu section can see and modify details of tokens assigned to them in the User profile → API tokens section only if Manage API tokens is allowed in their user role permissions.

13 Other parameters

This section allows configuring miscellaneous other frontend parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontend URL</td>
<td>Example: <a href="https://localhost/zabbix/ui/">https://localhost/zabbix/ui/</a></td>
</tr>
<tr>
<td>* Group for discovered hosts</td>
<td>type here to search</td>
</tr>
<tr>
<td>Default host inventory mode</td>
<td>Disabled, Manual, Automatic</td>
</tr>
<tr>
<td>User group for database down message</td>
<td>type here to search</td>
</tr>
<tr>
<td>Log unmatched SNMP traps</td>
<td>true</td>
</tr>
<tr>
<td>Authorization</td>
<td></td>
</tr>
<tr>
<td>* Login attempts</td>
<td>5</td>
</tr>
<tr>
<td>* Login blocking interval</td>
<td>30s</td>
</tr>
<tr>
<td>Storage of secrets</td>
<td></td>
</tr>
<tr>
<td>Vault provider</td>
<td>HashiCorp Vault, CyberArk Vault</td>
</tr>
<tr>
<td>Security</td>
<td></td>
</tr>
<tr>
<td>Validate URI schemes</td>
<td>true</td>
</tr>
<tr>
<td>Valid URI schemes</td>
<td>http,https,ftp,tar,mime,mailto,tel,ssh</td>
</tr>
<tr>
<td>* X-Frame-Options HTTP header</td>
<td>SAMEORIGIN</td>
</tr>
<tr>
<td>Use iframe sandboxing</td>
<td>true</td>
</tr>
<tr>
<td>Iframe sandboxing exceptions</td>
<td></td>
</tr>
<tr>
<td>Communication with Zabbix server</td>
<td></td>
</tr>
<tr>
<td>* Network timeout</td>
<td>3s</td>
</tr>
<tr>
<td>* Connection timeout</td>
<td>3s</td>
</tr>
<tr>
<td>* Network timeout for media type test</td>
<td>65s</td>
</tr>
<tr>
<td>* Network timeout for script execution</td>
<td>60s</td>
</tr>
<tr>
<td>* Network timeout for item test</td>
<td>60s</td>
</tr>
<tr>
<td>* Network timeout for scheduled report test</td>
<td>60s</td>
</tr>
</tbody>
</table>

Update | Reset defaults
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontend URL</td>
<td>URL to Zabbix web interface. This parameter is used by Zabbix web service for communication with frontend and should be specified to enable scheduled reports.</td>
</tr>
<tr>
<td>Group for discovered hosts</td>
<td>Hosts discovered by network discovery and agent autoregistration will be automatically placed in the host group, selected here.</td>
</tr>
<tr>
<td>Default host inventory mode</td>
<td>Default mode for host inventory. It will be followed whenever a new host or host prototype is created by server or frontend unless overridden during host discovery/autoregistration by the Set host inventory mode operation.</td>
</tr>
<tr>
<td>User group for database down message</td>
<td>Zabbix depends on the availability of the backend database. It cannot work without a database. If the database is down, selected users can be notified by Zabbix. Notifications will be sent to the user group set here using all configured user media entries. Zabbix server will not stop; it will wait until the database is back again to continue processing. Notification consists of the following content:</td>
</tr>
<tr>
<td>Log unmatched SNMP traps</td>
<td>Log SNMP trap if no corresponding SNMP interfaces have been found.</td>
</tr>
</tbody>
</table>

Authorization

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login attempts</td>
<td>Number of unsuccessful login attempts before the possibility to log in gets blocked.</td>
</tr>
<tr>
<td>Login blocking interval</td>
<td>Period of time for which logging in will be prohibited when Login attempts limit is exceeded.</td>
</tr>
</tbody>
</table>

Storage of secrets

Vault provider parameter allows selecting secret management software for storing user macro values. Supported options: - HashiCorp Vault (default) - CyberArk Vault

See also: Storage of secrets.

Security

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate URI schemes</td>
<td>Uncheck the box to disable URI scheme validation against the whitelist defined in Valid URI schemes. (enabled by default).</td>
</tr>
<tr>
<td>Valid URI schemes</td>
<td>A comma-separated list of allowed URI schemes. Applies to all fields in the frontend where URIs are used (for example, map element URLs). this field is editable only if Validate URI schemes is selected.</td>
</tr>
</tbody>
</table>
| X-Frame-Options HTTP header    | Value of HTTP X-Frame-options header. Supported values:  
SAMEORIGIN (default) - the page can only be displayed in a frame on the same origin as the page itself.  
DENY - the page cannot be displayed in a frame, regardless of the site attempting to do so.  
null - disable X-Frame-options header (not recommended).  
Or a list (string) of comma-separated hostnames. If a listed hostname is not among allowed, the SAMEORIGIN option is used. |
| Use iframe sandboxing          | This parameter determines whether retrieved URL content should be put into the sandbox or not. Note, that turning off sandboxing is not recommended.                                                           |
| Iframe sandboxing exceptions   | If sandboxing is enabled and this field is empty, all sandbox attribute restrictions apply. To disable some of the restrictions, specified them in this field. This disables only restrictions listed here, other restrictions will still be applied. See sandbox attribute description for additional information. |

Communication with Zabbix server
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network timeout</td>
<td>How many seconds to wait before closing an idle socket (if a connection to Zabbix server has been established earlier, but frontend cannot finish read/send data operation during this time, the connection will be dropped). Allowed range: 1 - 300s (default: 3s).</td>
</tr>
<tr>
<td>Connection timeout</td>
<td>How many seconds to wait before stopping an attempt to connect to Zabbix server. Allowed range: 1 - 300s (default: 3s).</td>
</tr>
<tr>
<td>Network timeout for media type test</td>
<td>How many seconds to wait for a response when testing a media type. Allowed range: 1 - 300s (default: 65s).</td>
</tr>
<tr>
<td>Network timeout for script execution</td>
<td>How many seconds to wait for a response when executing a script. Allowed range: 1 - 300s (default: 60s).</td>
</tr>
<tr>
<td>Network timeout for item test</td>
<td>How many seconds to wait for returned data when testing an item. Allowed range: 1 - 300s (default: 60s).</td>
</tr>
<tr>
<td>Network timeout for scheduled report test</td>
<td>How many seconds to wait for returned data when testing a scheduled report. Allowed range: 1 - 300s (default: 60s).</td>
</tr>
</tbody>
</table>

2 Proxies

Overview

In the Administration → Proxies section proxies for distributed monitoring can be configured in the Zabbix frontend.

Proxies

A listing of existing proxies with their details is displayed.

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the proxy. Clicking on the proxy name opens the proxy configuration form.</td>
</tr>
<tr>
<td>Mode</td>
<td>Proxy mode is displayed - Active or Passive.</td>
</tr>
<tr>
<td>Encryption</td>
<td>Encryption status for connections from the proxy is displayed:</td>
</tr>
<tr>
<td></td>
<td><strong>None</strong> - no encryption</td>
</tr>
<tr>
<td></td>
<td><strong>PSK</strong> - using pre-shared key</td>
</tr>
<tr>
<td></td>
<td><strong>Cert</strong> - using certificate</td>
</tr>
<tr>
<td>Last seen (age)</td>
<td>The time when the proxy was last seen by the server is displayed.</td>
</tr>
<tr>
<td>Host count</td>
<td>The number of enabled hosts assigned to the proxy is displayed.</td>
</tr>
<tr>
<td>Item count</td>
<td>The number of enabled items on enabled hosts assigned to the proxy is displayed.</td>
</tr>
<tr>
<td>Required performance (vps)</td>
<td>Required proxy performance is displayed (the number of values that need to be collected per second).</td>
</tr>
<tr>
<td>Hosts</td>
<td>All hosts monitored by the proxy are listed. Clicking on the host name opens the host configuration form.</td>
</tr>
</tbody>
</table>

To configure a new proxy, click on the Create proxy button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- Refresh configuration - refresh configuration of the proxies
- Enable hosts - change the status of hosts monitored by the proxy to Monitored
- Disable hosts - change the status of hosts monitored by the proxy to Not monitored
- Delete - delete the proxies
To use these options, mark the checkboxes before the respective proxies, then click on the required button.

Using filter

You can use the filter to display only the proxies you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available above the list of proxies. If you click on it, a filter becomes available where you can filter proxies by name and mode.

3 Authentication

Overview

The Administration → Authentication section allows to specify the global user authentication method to Zabbix and internal password requirements. The available methods are internal, HTTP, LDAP, and SAML authentication.

Default authentication

By default, Zabbix uses internal Zabbix authentication for all users. It is possible to change the default method to LDAP system-wide or enable LDAP authentication only for specific user groups.

To set LDAP as default authentication method for all users, navigate to the LDAP tab and configure authentication parameters, then return to the Authentication tab and switch Default authentication selector to LDAP.

Note that the authentication method can be fine-tuned on the user group level. Even if LDAP authentication is set globally, some user groups can still be authenticated by Zabbix. These groups must have frontend access set to Internal. Vice versa, if internal authentication is used globally, LDAP authentication details can be specified and used for specific user groups whose frontend access is set to LDAP. If a user is included into at least one user group with LDAP authentication, this user will not be able to use internal authentication method.

HTTP and SAML 2.0 authentication methods can be used in addition to the default authentication method.

Internal authentication

The Authentication tab allows defining custom password complexity requirements for internal Zabbix users.
The following password policy options can be configured:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum password length</td>
<td>By default, the minimum password length is set to 8. Supported range: 1-70. Note that passwords longer than 72 characters will be truncated.</td>
</tr>
<tr>
<td>Password must contain</td>
<td>Mark one or several checkboxes to require usage of specified characters in a password:</td>
</tr>
<tr>
<td></td>
<td>- an uppercase and a lowercase Latin letter</td>
</tr>
<tr>
<td></td>
<td>- a digit</td>
</tr>
<tr>
<td></td>
<td>- a special character</td>
</tr>
<tr>
<td>Avoid easy-to-guess passwords</td>
<td>Hover over the question mark to see a hint with the list of characters for each option. If marked, a password will be checked against the following requirements:</td>
</tr>
<tr>
<td></td>
<td>- must not contain user’s name, surname, or username</td>
</tr>
<tr>
<td></td>
<td>- must not be one of the common or context-specific passwords.</td>
</tr>
</tbody>
</table>

Changes in password complexity requirements will not affect existing user passwords, but if an existing user chooses to change a password, the new password will have to meet current requirements. A hint with the list of requirements will be displayed next to the Password field in the user profile and in the user configuration form accessible from the Administration→Users menu.

HTTP authentication

HTTP or web server-based authentication (for example: Basic Authentication, NTLM/Kerberos) can be used to check user names and passwords. Note that a user must exist in Zabbix as well, however its Zabbix password will not be used.

Be careful! Make sure that web server authentication is configured and works properly before switching it on.
Configuration parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable HTTP authentication</td>
<td>Mark the checkbox to enable HTTP authentication.</td>
</tr>
<tr>
<td>Default login form</td>
<td>Specify whether to direct non-authenticated users to:</td>
</tr>
<tr>
<td></td>
<td>Zabbix login form - standard Zabbix login page.</td>
</tr>
<tr>
<td></td>
<td>HTTP login form - HTTP login page.</td>
</tr>
<tr>
<td></td>
<td>It is recommended to enable web-server based authentication for the index_http.php page only.</td>
</tr>
<tr>
<td></td>
<td>If Default login form is set to 'HTTP login page' the user will be logged in automatically if</td>
</tr>
<tr>
<td></td>
<td>web server authentication module will set valid user login in the $_SERVER variable.</td>
</tr>
<tr>
<td></td>
<td>Supported $_SERVER keys are PHP_AUTH_USER, REMOTE_USER, AUTH_USER.</td>
</tr>
<tr>
<td>Remove domain name</td>
<td>A comma-delimited list of domain names that should be removed from the username.</td>
</tr>
<tr>
<td></td>
<td>E.g. comp, any - if username is 'Admin@any', 'comp\Admin', user will be logged in as 'Admin';</td>
</tr>
<tr>
<td></td>
<td>if username is 'notacompany\Admin', login will be denied.</td>
</tr>
<tr>
<td>Case sensitive login</td>
<td>Unmark the checkbox to disable case-sensitive login (enabled by default) for usernames.</td>
</tr>
<tr>
<td></td>
<td>E.g. disable case-sensitive login and log in with, for example, 'ADMIN' user even if the Zabbix</td>
</tr>
<tr>
<td></td>
<td>user is 'Admin'.</td>
</tr>
<tr>
<td></td>
<td>Note that with case-sensitive login disabled the login will be denied if multiple users exist in</td>
</tr>
<tr>
<td></td>
<td>Zabbix database with similar usernames (e.g. Admin, admin).</td>
</tr>
</tbody>
</table>

In case of web server authentication all users (even with frontend access set to LDAP/Internal) will be authenticated by the web server, not by Zabbix!

For internal users who are unable to log in using HTTP credentials (with HTTP login form set as default) leading to the 401 error, you may want to add a ErrorDocument 401 /index.php?form=default line to basic authentication directives, which will redirect to the regular Zabbix login form.

LDAP authentication

External LDAP authentication can be used to check user names and passwords. Note that a user must exist in Zabbix as well, however its Zabbix password will not be used.

Several LDAP servers can be defined, if necessary. For example, a different server can be used to authenticate a different user group. Once LDAP servers are configured, in user group configuration it becomes possible to select the required LDAP server for the respective user group.

If a user is in multiple user groups and multiple LDAP servers, the first server in the list of LDAP servers sorted by name in ascending order will be used for authentication.

Zabbix LDAP authentication works at least with Microsoft Active Directory and OpenLDAP.
## Configuration parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable LDAP authentication</td>
<td>Mark the checkbox to enable LDAP authentication.</td>
</tr>
<tr>
<td>Servers</td>
<td>Click on Add to configure an LDAP server (see LDAP server configuration parameters below). Unmark the checkbox to disable case-sensitive login (enabled by default) for usernames. E.g. disable case-sensitive login and log in with, for example, ‘ADMIN’ user even if the Zabbix user is ’Admin’. Note that with case-sensitive login disabled the login will be denied if multiple users exist in Zabbix database with similar usernames (e.g. Admin, admin).</td>
</tr>
<tr>
<td>Case-sensitive login</td>
<td></td>
</tr>
</tbody>
</table>

### LDAP server configuration parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Host</th>
<th>User groups</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP server</td>
<td>ldap://ldap.example.com</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>LDAP server2</td>
<td>ldap://ldap2.example.com</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Add
LDAP server configuration parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the LDAP server in Zabbix configuration.</td>
</tr>
<tr>
<td>Host</td>
<td>Host of the LDAP server. For example: ldap://ldap.example.com. For secure LDAP server use ldaps protocol. ldaps://ldap.example.com. With OpenLDAP 2.x.x and later, a full LDAP URI of the form ldaps://hostname:port or ldaps://hostname:port may be used.</td>
</tr>
<tr>
<td>Port</td>
<td>Port of the LDAP server. Default is 389. For secure LDAP connection port number is normally 636. Not used when using full LDAP URIs.</td>
</tr>
<tr>
<td>Base DN</td>
<td>Base path to search accounts: ou=Users,ou=system (for OpenLDAP), DC=company,DC=com (for Microsoft Active Directory)</td>
</tr>
<tr>
<td>Search attribute</td>
<td>LDAP account attribute used for search: uid (for OpenLDAP), sAMAccountName (for Microsoft Active Directory)</td>
</tr>
<tr>
<td>Bind DN</td>
<td>LDAP account for binding and searching over the LDAP server, examples: uid=ldap_search,ou=system (for OpenLDAP), CN=ldap_search,OU=user_group,DC=company,DC=com (for Microsoft Active Directory) Anonymous binding is also supported. Note that anonymous binding potentially opens up domain configuration to unauthorized users (information about users, computers, servers, groups, services, etc.). For security reasons, disable anonymous binds on LDAP hosts and use authenticated access instead.</td>
</tr>
</tbody>
</table>
Parameter | Description
--- | ---
Bind password | LDAP password of the account for binding and searching over the LDAP server.
Description | Description of the LDAP server.
StartTLS | Mark the checkbox to use the StartTLS operation when connecting to LDAP server. The connection will fail if the server doesn’t support StartTLS. StartTLS cannot be used with servers that use the ldaps protocol
To access this option, mark the Advanced configuration checkbox first.
Search filter | Define a custom string when authenticating user in LDAP. The following placeholders are supported:
%{attr} - search attribute name (uid, sAMAccountName)
%{user} - user username value to authenticate.
If omitted then LDAP will use the default filter: (%{attr}=%{user})
To access this option, mark the Advanced configuration checkbox first.

The Test button allows to test user access:

Parameter | Description
--- | ---
Login | LDAP user name to test (prefilled with the current user name from Zabbix frontend). This user name must exist in the LDAP server.
User password | LDAP user password to test.

In case of trouble with certificates, to make a secure LDAP connection (ldaps) work you may need to add a TLS_REQCERT allow line to the /etc/openldap/ldap.conf configuration file. It may decrease the security of connection to the LDAP catalog.

It is recommended to create a separate LDAP account (Bind DN) to perform binding and searching over the LDAP server with minimal privileges in the LDAP instead of using real user accounts (used for logging in the Zabbix frontend).

Such an approach provides more security and does not require changing the Bind password when the user changes his own password in the LDAP server.

In the table above it’s ldap_search account name.

SAML authentication

SAML 2.0 authentication can be used to sign in to Zabbix. Note that a user must exist in Zabbix, however, its Zabbix password will not be used. If authentication is successful, then Zabbix will match a local username with the username attribute returned by SAML.

If SAML authentication is enabled, users will be able to choose between logging in locally or via SAML Single Sign-On.

Setting up the identity provider

In order to work with Zabbix, a SAML identity provider (onelogin.com, auth0.com, okta.com, etc.) needs to be configured in the following way:

- Assertion Consumer URL should be set to <path_to_zabbix_ui>/index_sso.php?acs
- Single Logout URL should be set to <path_to_zabbix_ui>/index_sso.php?sls


Setting up Zabbix

It is required to install php-openssl if you want to use SAML authentication in the frontend.

To use SAML authentication Zabbix should be configured in the following way:

1. Private key and certificate should be stored in the ui/conf/certs/, unless custom paths are provided in zabbix.conf.php.

By default, Zabbix will look in the following locations:

- ui/conf/certs/sp.key - SP private key file
- ui/conf/certs/sp.crt - SP cert file
- ui/conf/certs/idp.crt - IDP cert file

2. All of the most important settings can be configured in the Zabbix frontend. However, it is possible to specify additional settings in the configuration file.
Configuration parameters, available in the Zabbix frontend:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable SAML authentication</td>
<td>Mark the checkbox to enable SAML authentication.</td>
</tr>
<tr>
<td>IDP entity ID</td>
<td>The unique identifier of SAML identity provider.</td>
</tr>
<tr>
<td>SSO service URL</td>
<td>The URL users will be redirected to when logging in.</td>
</tr>
<tr>
<td>SLO Service URL</td>
<td>The URL users will be redirected to when logging out. If left empty, the SLO service will not be used.</td>
</tr>
<tr>
<td>// Username attribute//</td>
<td>SAML attribute to be used as a username when logging into Zabbix.</td>
</tr>
<tr>
<td></td>
<td>List of supported values is determined by the identity provider.</td>
</tr>
<tr>
<td>Examples:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>uid</td>
</tr>
<tr>
<td></td>
<td>userprincipalname</td>
</tr>
<tr>
<td></td>
<td>samaccountname</td>
</tr>
<tr>
<td></td>
<td>username</td>
</tr>
<tr>
<td></td>
<td>userusername</td>
</tr>
<tr>
<td></td>
<td>urn:oid:0.9.2342.19200300.100.1.1</td>
</tr>
<tr>
<td></td>
<td>urn:oid:1.3.6.1.4.1.5923.1.1.1.13</td>
</tr>
<tr>
<td></td>
<td>urn:oid:0.9.2342.19200300.100.1.44</td>
</tr>
<tr>
<td>SP entity ID</td>
<td>The unique identifier of SAML service provider.</td>
</tr>
</tbody>
</table>
Parameter | Description
--- | ---
SP name ID format | Defines which name identifier format should be used.
  
Examples:
urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
urn:oasis:names:tc:SAML:2.0:nameid-format:transient
urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos
urn:oasis:names:tc:SAML:2.0:nameid-format:entity

Sign | Mark the checkboxes to select entities for which SAML signature should be enabled:
- Messages
- Assertions
- AuthN requests
- Logout requests
- Logout responses

Encrypt | Mark the checkboxes to select entities for which SAML encryption should be enabled:
- Assertions
- Name ID

Case-sensitive login | Mark the checkbox to enable case-sensitive login (disabled by default) for usernames. E.g. disable case-sensitive login and log in with, for example, ‘ADMIN’ user even if the Zabbix user is ‘Admin’.
Note that with case-sensitive login disabled the login will be denied if multiple users exist in Zabbix database with similar usernames (e.g. Admin, admin).

Advanced settings

Additional SAML parameters can be configured in the Zabbix frontend configuration file (zabbix.conf.php):

- $SSO['SP_KEY'] = '<path to the SP private key file>';
- $SSO['SP_CERT'] = '<path to the SP cert file>';
- $SSO['IDP_CERT'] = '<path to the IDP cert file>';
- $SSO['SETTINGS']

Zabbix uses OneLogin’s SAML PHP Toolkit library (version 3.4.1). The structure of $SSO['SETTINGS'] section should be similar to the structure used by the library. For the description of configuration options, see official library documentation.

Only the following options can be set as part of $SSO['SETTINGS']:

- strict
- baseurl
- compress
- contactPerson
- organization
- sp (only options specified in this list)
  - attributeConsumingService
  - x509certNew
- idp (only options specified in this list)
  - singleLogoutService (only one option)
    * responseUrl
  - certFingerprint
  - certFingerprintAlgorithm
  - x509certMulti
- security (only options specified in this list)
  - signMetadata
  - wantNameId
  - requestedAuthnContext
  - requestedAuthnContextComparison
  - wantXMLValidation
  - relaxDestinationValidation
  - destinationStrictlyMatches
  - rejectUnsolicitedResponsesWithInResponseTo
  - signatureAlgorithm
  - digestAlgorithm
  - lowercaseUrlEncoding

All other options will be taken from the database and cannot be overridden. The debug option will be ignored.
In addition, if Zabbix UI is behind a proxy or a load balancer, the custom `use_proxy_headers` option can be used:

- false (default) - ignore the option;
- true - use X-Forwarded-* HTTP headers for building the base URL.

If using a load balancer to connect to Zabbix instance, where the load balancer uses TLS/SSL and Zabbix does not, you must indicate 'baseurl', 'strict' and 'use_proxy_headers' parameters as follows:

```php
$SSO_SETTINGS = ['strict' => false, 'baseurl' => 'https://zabbix.example.com/zabbix/', 'use_proxy_headers' => true];
```

Configuration example:

```php
$SSO['SETTINGS'] = [
    'security' => [
        'signatureAlgorithm' => 'http://www.w3.org/2001/04/xmldsig-more#rsa-sha384',
        'digestAlgorithm' => 'http://www.w3.org/2001/04/xmldsig-more#sha384',
        // ...
    ],
    // ...
];
```

### 4 User groups

**Overview**

In the Administration → User groups section user groups of the system are maintained.

**User groups**

A listing of existing user groups with their details is displayed.

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the user group. Clicking on the user group name opens the user group configuration form.</td>
</tr>
<tr>
<td>#</td>
<td>The number of users in the group. Clicking on Users will display the respective users filtered out in the user list.</td>
</tr>
<tr>
<td>Members</td>
<td>Usernames of individual users in the user group (with name and surname in parentheses). Clicking on the username will open the user configuration form. Users from disabled groups are displayed in red.</td>
</tr>
<tr>
<td>Frontend access</td>
<td>Frontend access level is displayed: System default - Zabbix, LDAP or HTTP authentication; depending on the chosen authentication method Internal - the user is authenticated by Zabbix regardless of system settings Disabled - frontend access for this user is disabled. By clicking on the current level you can change it.</td>
</tr>
<tr>
<td>Status</td>
<td>User group status is displayed - Enabled or Disabled. By clicking on the status you can change it.</td>
</tr>
</tbody>
</table>

To configure a new user group, click on the Create user group button in the top right-hand corner.

**Mass editing options**
Buttons below the list offer some mass-editing options:

- Enable - change the user group status to Enabled
- Disable - change the user group status to Disabled
- Enable debug mode - enable debug mode for the user groups
- Disable debug mode - disable debug mode for the user groups
- Delete - delete the user groups

To use these options, mark the checkboxes before the respective user groups, then click on the required button.

Using filter

You can use the filter to display only the user groups you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available above the list of user groups. If you click on it, a filter becomes available where you can filter user groups by name and status.

5 User roles

Overview

In the Administration → User roles section roles that can be assigned to system users and specific permissions for each role are maintained.

Default user roles

By default, Zabbix is configured with four user roles, which have a pre-defined set of permissions:

- Admin role
- Guest role
- Super admin role
- User role

Default Super admin role cannot be modified or deleted, because at least one Super admin user with unlimited privileges must exist in Zabbix.

Zabbix users with type Super admins and proper permissions can modify or delete existing roles or create new custom roles.

To create a new role, click on the Create user role button at the top right corner. To update an existing role, press on the role name to open the configuration form.
Available permission options along with default permission sets for pre-configured user roles in Zabbix are described below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default user roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>User type</td>
<td>Selected user type determines the list of available permissions. Upon selecting a user type, all available permissions for this user type are granted by default. Uncheck the checkbox(es) to revoke certain permissions for the user role. Checkboxes for permissions not available for this user type are grayed out.</td>
<td></td>
</tr>
<tr>
<td>Access to UI elements</td>
<td>Monitoring: Enable/disable access to a specific Monitoring menu section and underlying pages.</td>
<td>Yes Yes Yes Yes</td>
</tr>
<tr>
<td></td>
<td>Problems: No No No No</td>
<td>Super admin role</td>
</tr>
<tr>
<td></td>
<td>Hosts: No No No No</td>
<td>Admin role</td>
</tr>
<tr>
<td></td>
<td>Latest data: No No No</td>
<td>User role</td>
</tr>
<tr>
<td></td>
<td>Maps: No No No No</td>
<td>Guest role</td>
</tr>
<tr>
<td></td>
<td>Discovery: No No No</td>
<td>Admin role</td>
</tr>
<tr>
<td></td>
<td>Services: Enable/disable access to a specific Services menu section and underlying pages.</td>
<td>Yes Yes Yes Yes</td>
</tr>
<tr>
<td></td>
<td>Service actions: No No</td>
<td>Super admin role</td>
</tr>
<tr>
<td></td>
<td>SLA: No No No No</td>
<td>Admin role</td>
</tr>
<tr>
<td></td>
<td>SLA report: Yes Yes</td>
<td>User role</td>
</tr>
<tr>
<td></td>
<td>Inventory: Enable/disable access to a specific Inventory menu section and underlying pages.</td>
<td>Yes Yes Yes Yes</td>
</tr>
<tr>
<td></td>
<td>Overview: Yes No</td>
<td>Guest role</td>
</tr>
<tr>
<td></td>
<td>Hosts: Yes Yes</td>
<td>Admin role</td>
</tr>
<tr>
<td></td>
<td>818</td>
<td>User role</td>
</tr>
<tr>
<td>Reports</td>
<td>Enable/disable access to a specific Reports menu section and underlying pages.</td>
<td>Yes</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>System information</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Availability report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Triggers top 100 Audit Action log Notifications Scheduled reports</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration</td>
<td>Enable/disable access to a specific Configuration menu section and underlying pages.</td>
<td>Yes</td>
</tr>
<tr>
<td>Template groups Host groups Templates Hosts Maintenance Actions Event correlation Discovery</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Administration</td>
<td>Enable/disable access to a specific Administration menu section and underlying pages.</td>
<td>Yes</td>
</tr>
<tr>
<td>General Proxies Authentication User groups User roles Users Media types Scripts Queue</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default access to new UI elements</td>
<td>Enable/disable access to the custom UI elements. Modules, if present, will be listed below.</td>
<td>Yes</td>
</tr>
<tr>
<td>Access to services</td>
<td>Select read-write access to services: <strong>None</strong> - no access at all <strong>All</strong> - access to all services is read-write <strong>Service list</strong> - select services for read-write access</td>
<td>All</td>
</tr>
<tr>
<td>Read-write access to services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read-write access to services with tag</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The read-write access, if granted, takes precedence over the read-only access settings and is dynamically inherited by the child services.

Specify tag name and, optionally, value to additionally grant read-write access to services matching the tag.

This option is available if 'Service list' is selected in the Read-write access to services parameter.

The read-write access, if granted, takes precedence over the read-only access settings and is dynamically inherited by the child services.
<table>
<thead>
<tr>
<th><strong>Read-only access to services</strong></th>
<th>Select read-only access to services:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All - access to all services is read-only</td>
</tr>
<tr>
<td></td>
<td>Service list - select services for read-only access</td>
</tr>
</tbody>
</table>

The read-only access does not take precedence over the read-write access and is dynamically inherited by the child services.

<table>
<thead>
<tr>
<th><strong>Read-only access to services with tag</strong></th>
<th>Specify tag name and, optionally, value to additionally grant read-only access to services matching the tag.</th>
</tr>
</thead>
<tbody>
<tr>
<td>This option is available if 'Service list' is selected in the Read-only access to services parameter.</td>
<td></td>
</tr>
<tr>
<td>The read-only access does not take precedence over the read-write access and is dynamically inherited by the child services.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Access to modules</strong></th>
<th>Allow/deny access to a specific module. Only enabled modules are shown in this section. It is not possible to grant or restrict access to a module that is currently disabled.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default access to new modules</td>
<td>Enable/disable access to modules that may be added in the future.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Access to API</strong></th>
<th>Enable/disable access to API.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Select Allow list to allow only specified API methods or Deny list to restrict only specified API methods.</td>
</tr>
</tbody>
</table>

In the search field, start typing the method name, then select the method from the auto-complete list. You can also press the Select button and select methods from the full list available for this user type. Note, that if certain action from the Access to actions block is unchecked, users will not be able to use API methods related to this action.

Wildcards are supported. Examples: dashboard.* (all methods of 'dashboard.' API service) *.export (method with '.export' name from all API services).

If no methods have been specified the Allow/Deny list rule will be ignored.

<table>
<thead>
<tr>
<th><strong>Access to actions</strong></th>
<th>Clearing this checkbox will also revoke the rights to use .create, .update and .delete API methods for the corresponding elements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create and edit dashboards</td>
<td>Yes</td>
</tr>
<tr>
<td>Create and edit maps</td>
<td>Yes</td>
</tr>
<tr>
<td>Create and edit maintenance</td>
<td>Yes</td>
</tr>
<tr>
<td>Add problem comments</td>
<td>Yes</td>
</tr>
<tr>
<td>Change severity</td>
<td>No</td>
</tr>
</tbody>
</table>

Wildcards are supported. Examples: dashboard.* (all methods of 'dashboard.' API service) *.export (method with '.export' name from all API services).

If no methods have been specified the Allow/Deny list rule will be ignored.
### Acknowledge problems
Clearing this checkbox will also revoke the rights to use the `script.execute` API method.

### Suppress problems
Clearing this checkbox will also revoke the rights to use all `token` API methods.

### Close problems
Clearing this checkbox will also revoke the rights to use all `report` API methods.

### Execute scripts
Clearing this checkbox will also revoke the rights to use the `script.execute` API method.

### Manage API tokens
Clearing this checkbox will also revoke the rights to use all API methods.

### Manage scheduled reports
Clearing this checkbox will also revoke the rights to use all `report` API methods.

### Manage SLA
Enable/disable the rights to manage SLA.

### Invoke "Execute now" on read-only hosts
Allow to use the “Execute now” option in latest data for items of read-only hosts.

### Default access to new actions
Enable/disable access to new actions.

**Notes:**
- Each user may have only one role assigned.
- If an element is restricted, users will not be able to access it even by entering a direct URL to this element into the browser.
- Users of type User or Admin cannot change their own role settings.
- Users of type Super admin can modify settings of their own role (not available for the default Super admin role), but not the user type.
- Users of all levels cannot change their own user type.

**See also:**
- [Configuring a user](#)

## 6 Users
### Overview
In the Administration → Users section users of the system are maintained.

**Users**

A listing of existing users with their details is displayed.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>Username for logging into Zabbix. Clicking on the username opens the user configuration form.</td>
</tr>
<tr>
<td>Name</td>
<td>First name of the user.</td>
</tr>
<tr>
<td>Last name</td>
<td>Second name of the user.</td>
</tr>
<tr>
<td>User role</td>
<td><code>User role</code> is displayed.</td>
</tr>
<tr>
<td>Groups</td>
<td>Groups that the user is a member of are listed. Clicking on the user group name opens the user group configuration form. Disabled groups are displayed in red.</td>
</tr>
<tr>
<td>Column</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Is online?</td>
<td>The on-line status of the user is displayed - Yes or No. The time of last user activity is displayed in parentheses.</td>
</tr>
<tr>
<td>Login</td>
<td>The login status of the user is displayed - Ok or Blocked. A user can become temporarily blocked upon exceeding the number of unsuccessful login attempts set in the Administration→General section (five by default). By clicking on Blocked you can unblock the user.</td>
</tr>
<tr>
<td>Frontend access</td>
<td>Frontend access level is displayed - System default, Internal or Disabled, depending on the one set for the whole user group.</td>
</tr>
<tr>
<td>API access</td>
<td>API access status is displayed - Enabled or Disabled, depending on the one set for the user role.</td>
</tr>
<tr>
<td>Debug mode</td>
<td>Debug mode status is displayed - Enabled or Disabled, depending on the one set for the whole user group.</td>
</tr>
<tr>
<td>Status</td>
<td>User status is displayed - Enabled or Disabled, depending on the one set for the whole user group.</td>
</tr>
</tbody>
</table>

To configure a new user, click on the Create user button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- Unblock - re-enable system access to blocked users
- Delete - delete the users

To use these options, mark the check-boxes before the respective users, then click on the required button.

Using filter

You can use the filter to display only the users you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available above the list of users. If you click on it, a filter becomes available where you can filter users by username, name, last name and user role.

7 Media types

Overview

In the Administration → Media types section users can configure and maintain media type information.

Media type information contains general instructions for using a medium as delivery channel for notifications. Specific details, such as the individual e-mail addresses to send a notification to are kept with individual users.

A listing of existing media types with their details is displayed.

Displayed data:
## Column Description

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the media type. Clicking on the name opens the media type configuration form.</td>
</tr>
<tr>
<td>Type</td>
<td>Type of the media (e-mail, SMS, etc) is displayed.</td>
</tr>
<tr>
<td>Status</td>
<td>Media type status is displayed - Enabled or Disabled. By clicking on the status you can change it.</td>
</tr>
<tr>
<td>Used in actions</td>
<td>All actions where the media type is used directly (selected in the Send only to dropdown) are displayed. Clicking on the action name opens the action configuration form.</td>
</tr>
<tr>
<td>Details</td>
<td>Detailed information of the media type is displayed.</td>
</tr>
<tr>
<td>Actions</td>
<td>The following action is available: <strong>Test</strong> - click to open a testing form where you can enter media type parameters (e.g. a recipient address with test subject and body) and send a test message to verify that the configured media type works. See also: Media type testing.</td>
</tr>
</tbody>
</table>

To configure a new media type, click on the Create media type button in the top right-hand corner.

To import a media type from XML, click on the Import button in the top right-hand corner.

### Mass editing options

Buttons below the list offer some mass-editing options:

- Enable - change the media type status to Enabled
- Disable - change the media type status to Disabled
- Export - export the media types to a YAML, XML or JSON file
- Delete - delete the media types

To use these options, mark the checkboxes before the respective media types, then click on the required button.

### Using filter

You can use the filter to display only the media types you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available above the list of media types. If you click on it, a filter becomes available where you can filter media types by name and status.

### 8 Scripts

#### Overview

In the Administration → Scripts section user-defined global scripts can be configured and maintained.

Global scripts, depending on the configured scope and also user permissions, are available for execution:

- from the **host menu** in various frontend locations (Dashboard, Problems, Latest data, Maps, etc)
- from the **event menu**
- can be run as an action operation

The scripts are executed on Zabbix agent, Zabbix server (proxy) or Zabbix server only. See also Command execution.

Both on Zabbix agent and Zabbix proxy remote scripts are disabled by default. They can be enabled by:

- For remote commands executed on Zabbix agent:
  - adding an AllowKey=system.run[<command>,*] parameter for each allowed command in agent configuration, * stands for wait and nowait mode;
- For remote commands executed on Zabbix proxy:
  - **Warning:** It is not required to enable remote commands on Zabbix proxy if remote commands are executed on Zabbix agent that is monitored by Zabbix proxy. If, however, it is required to execute remote commands on Zabbix proxy, set EnableRemoteCommands parameter to ‘1’ in the proxy configuration.
A listing of existing scripts with their details is displayed.

Table:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the script. Clicking on the script name opens the script configuration form.</td>
</tr>
<tr>
<td>Scope</td>
<td>Scope of the script - action operation, manual host action or manual event action. This setting determines where the script is available.</td>
</tr>
<tr>
<td>Used in actions</td>
<td>Actions where the script is used are displayed.</td>
</tr>
<tr>
<td>Type</td>
<td>Script type is displayed - Webhook, Script, SSH, Telnet or IPMI command.</td>
</tr>
<tr>
<td>Execute on</td>
<td>It is displayed whether the script will be executed on Zabbix agent, Zabbix server (proxy) or Zabbix server only.</td>
</tr>
<tr>
<td>Commands</td>
<td>All commands to be executed within the script are displayed.</td>
</tr>
<tr>
<td>User group</td>
<td>The user group that the script is available to is displayed (or All for all user groups).</td>
</tr>
<tr>
<td>Host group</td>
<td>The host group that the script is available for is displayed (or All for all host groups).</td>
</tr>
<tr>
<td>Host access</td>
<td>The permission level for the host group is displayed - Read or Write. Only users with the required permission level will have access to executing the script.</td>
</tr>
</tbody>
</table>

To configure a new script, click on the Create script button in the top right-hand corner.

Mass editing options

A button below the list offers one mass-editing option:

- Delete - delete the scripts

To use this option, mark the checkboxes before the respective scripts and click on Delete.

Using filter

You can use the filter to display only the scripts you are interested in. For better search performance, data is searched with macros unresolved.

The Filter link is available above the list of scripts. If you click on it, a filter becomes available where you can filter scripts by name and scope.

Configuring a global script
Script attributes:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Unique name of the script. E.g. Clear /tmp filesystem</td>
</tr>
</tbody>
</table>
| Scope     | Scope of the script - action operation, manual host action or manual event action. This setting determines where the script can be used - in remote commands of action operations, from the host menu or from the event menu respectively. Setting the scope to ‘Action operation’ makes the script available for all users with access to Configuration → Actions. If a script is actually used in an action, its scope cannot be changed away from ‘action operation’.

**Macro support**
The scope affects the range of available macros. For example, user-related macros ({{USER.*}}) are supported in scripts to allow passing information about the user that launched the script. However, they are not supported if the script scope is action operation, as action operations are executed automatically.

To find out which macros are supported, do a search for ‘Trigger-based notifications and commands/Trigger-based commands’, ‘Manual host action scripts’ and ‘Manual event action scripts’ in the supported macro table. Note that if a macro may resolve to a value with spaces (for example, host name), don’t forget to quote as needed.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu path</td>
<td>The desired menu path to the script. For example, Default or Default/ will display the script in the respective directory. Menus can be nested, e.g. Main menu/Sub menu1/Sub menu2. When accessing scripts through the host/event menu in monitoring sections, they will be organized according to the given directories. This field is displayed only if ‘Manual host action’ or ‘Manual event action’ is selected as Scope.</td>
</tr>
<tr>
<td>Type</td>
<td>Click the respective button to select script type: <strong>Webhook, Script, SSH, Telnet</strong> or <strong>IPMI</strong> command.</td>
</tr>
<tr>
<td>Script type: Webhook</td>
<td>Specify the webhook variables as attribute-value pairs. See also: <strong>Webhook media configuration.</strong> Macros and custom user macros are supported in parameter values. Macro support depends on the scope of the script (see Scope above).</td>
</tr>
<tr>
<td>Parameters</td>
<td>Enter the JavaScript code in the block that appears when clicking in the parameter field (or on the view/edit button next to it). Macro support depends on the scope of the script (see Scope above).</td>
</tr>
<tr>
<td>Script</td>
<td>JavaScript execution timeout (1-60s, default 30s). Time suffixes are supported, e.g. 30s, 1m.</td>
</tr>
<tr>
<td>Script type: Script</td>
<td>Click the respective button to execute the shell script on: <strong>Zabbix agent</strong> - the script will be executed by Zabbix agent (if the system.run item is allowed) on the host <strong>Zabbix server (proxy)</strong> - the script will be executed by Zabbix server or proxy (if enabled by EnableRemoteCommands) - depending on whether the host is monitored by server or proxy <strong>Zabbix server</strong> - the script will be executed by Zabbix server only</td>
</tr>
<tr>
<td>Execute on</td>
<td>Enter full path to the commands to be executed within the script. Macro support depends on the scope of the script (see Scope above). Custom user macros are supported.</td>
</tr>
<tr>
<td>Authentication method</td>
<td>Select authentication method - password or public key.</td>
</tr>
<tr>
<td>Username</td>
<td>Enter the username.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the password. This field is available if ‘Password’ is selected as the authentication method.</td>
</tr>
<tr>
<td>Public key file</td>
<td>Enter the path to the public key file. This field is available if ‘Public key’ is selected as the authentication method.</td>
</tr>
<tr>
<td>Private key file</td>
<td>Enter the path to the private key file. This field is available if ‘Public key’ is selected as the authentication method.</td>
</tr>
<tr>
<td>Passphrase</td>
<td>Enter the passphrase. This field is available if ‘Public key’ is selected as the authentication method.</td>
</tr>
<tr>
<td>Port</td>
<td>Enter the port.</td>
</tr>
<tr>
<td>Commands</td>
<td>Enter the commands. Macro support depends on the scope of the script (see Scope above). Custom user macros are supported.</td>
</tr>
<tr>
<td>Script type: Telnet</td>
<td>Enter the username.</td>
</tr>
<tr>
<td>Username</td>
<td>Enter the username.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the password.</td>
</tr>
<tr>
<td>Port</td>
<td>Enter the port.</td>
</tr>
<tr>
<td>Commands</td>
<td>Enter the commands. Macro support depends on the scope of the script (see Scope above). Custom user macros are supported.</td>
</tr>
<tr>
<td>Script type: IPMI</td>
<td>Enter the IPMI command. Macro support depends on the scope of the script (see Scope above). Custom user macros are supported.</td>
</tr>
<tr>
<td>Command</td>
<td>Enter a description for the script.</td>
</tr>
<tr>
<td>Host group</td>
<td>Select the host group that the script will be available for (or All for all host groups).</td>
</tr>
<tr>
<td>User group</td>
<td>Select the user group that the script will be available to (or All for all user groups). This field is displayed only if ‘Manual host action’ or ‘Manual event action’ is selected as Scope.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Required host</td>
<td>Select the permission level for the host group - Read or Write. Only users with the required permission level will have access to executing the script. This field is displayed only if 'Manual host action' or 'Manual event action' is selected as Scope.</td>
</tr>
<tr>
<td>Enable con-</td>
<td>Mark the checkbox to display a confirmation message before executing the script. This feature might be especially useful with potentially dangerous operations (like a reboot script) or ones that might take a long time. This option is displayed only if 'Manual host action' or 'Manual event action' is selected as Scope.</td>
</tr>
<tr>
<td>fir-ma-</td>
<td>Enter a custom confirmation text for the confirmation popup enabled with the checkbox above (for example, Remote system will be rebooted. Are you sure?). To see how the text will look like, click on Test confirmation next to the field. {HOST.<em>} and {USER.</em>} macros are supported. Custom user macros are supported. This field is displayed only if 'Manual host action' or 'Manual event action' is selected as Scope.</td>
</tr>
<tr>
<td>tion Confirmation text</td>
<td></td>
</tr>
</tbody>
</table>

**Script execution and result**

Scripts run by Zabbix server are executed by the order described in Command execution section including exit code checking. The script result will be displayed in a pop-up window that will appear after the script is run.

Note: The return value of the script is standard output together with standard error.

See an example of a script and the result window below:

```
uname -v
/tmp/non_existing_script.sh
echo "This script was started by {USER.USERNAME}"  
```

![Unname](achte63.png)  

**Output**

```
#70-18.04.1-Ubuntu SMP Tue Jan 12 17:18:00 UTC 2021
sh: 2: /tmp/non_existing_script.sh: not found
This script was started by Admin
```

The script result does not display the script itself.

**Script timeout**

**Zabbix agent**

You may encounter a situation when a timeout occurs while executing a script.

See an example of a script running on Zabbix agent and the result window below:

```
sleep 5
df -h
```

![Unname](achte63.png)
The error message, in this case, is the following:

Timeout while executing a shell script.

In order to avoid such a situation, it is advised to optimize the script itself (instead of adjusting Timeout parameter to a corresponding value (in our case, > '5') by modifying the Zabbix agent configuration and Zabbix server configuration).

In case still the Timeout parameter is changed in Zabbix agent configuration following error message appears:

Get value from agent failed: ZBX_TCP_READ() timed out.

It means that modification was made in Zabbix agent configuration and it is required to modify Timeout setting also in Zabbix server configuration.

Zabbix server/proxy

See an example of a script running on Zabbix server and the result window below:

```
sleep 11
df -h
```

It is also advised to optimize the script itself (instead of adjusting TrapperTimeout parameter to a corresponding value (in our case, > '11') by modifying the Zabbix server configuration).

9 Queue

Overview

In the Administration → Queue section items that are waiting to be updated are displayed.

Ideally, when you open this section it should all be “green” meaning no items in the queue. If all items are updated without delay, there are none waiting. However, due to lacking server performance, connection problems or problems with agents, some items may get delayed and the information is displayed in this section. For more details, see the Queue section.

Queue is available only if Zabbix server is running.

The Administration → Queue section contains the following pages:

- Queue overview — displays queue by item type;
- Queue overview by proxy — displays queue by proxy;
• Queue details — displays a list of delayed items.

The list of available pages appears upon pressing on Queue in the Administration menu section. It is also possible to switch between pages by using a title dropdown in the top left corner.

Overview by item type

In this screen it is easy to locate if the problem is related to one or several item types.

Each line contains an item type. Each column shows the number of waiting items - waiting for 5-10 seconds/10-30 seconds/30-60 seconds/1-5 minutes/5-10 minutes or over 10 minutes respectively.

Overview by proxy

In this screen it is easy to locate if the problem is related to one of the proxies or the server.

Each line contains a proxy, with the server last in the list. Each column shows the number of waiting items - waiting for 5-10 seconds/10-30 seconds/30-60 seconds/1-5 minutes/5-10 minutes or over 10 minutes respectively.

List of waiting items

In this screen, each waiting item is listed.

Displayed data:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduled check</td>
<td>The time when the check was due is displayed.</td>
</tr>
</tbody>
</table>
Possible error messages

You may encounter a situation when no data is displayed and the following error message appears:

```
Permission denied.
```

Error message in this case is the following:

**Cannot display item queue. Permission denied**

This happens when PHP configuration parameters `$ZBX_SERVER_PORT` or `$ZBX_SERVER` in `zabbix.conf.php` point to existing Zabbix server which uses different database.

### 3 User settings

Overview

Depending on user role permissions, the User settings section may contain the following pages:

- User profile - for customizing certain Zabbix frontend features;
- API tokens - for managing API tokens assigned to the current user.

The list of available pages appears upon pressing on the user icon near the bottom of the Zabbix menu (not available for a guest user). It is also possible to switch between pages by using a title dropdown in the top left corner.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delayed by</td>
<td>The length of the delay is displayed.</td>
</tr>
<tr>
<td>Host</td>
<td>Host of the item is displayed.</td>
</tr>
<tr>
<td>Name</td>
<td>Name of the waiting item is displayed.</td>
</tr>
<tr>
<td>Proxy</td>
<td>The proxy name is displayed, if the host is monitored by proxy.</td>
</tr>
</tbody>
</table>

1 User profile

The **User profile** section provides options to set custom interface language, color theme, number of rows displayed in the lists, etc. The changes made here will be applied to the current user only.

The **User** tab allows you to set various user preferences.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>Click on the link to display two fields for entering a new password.</td>
</tr>
<tr>
<td>Language</td>
<td>Select the interface language of your choice or select System default to use default system settings.</td>
</tr>
<tr>
<td>Time zone</td>
<td>Select the time zone to override global time zone on user level or select System default to use global time zone settings.</td>
</tr>
</tbody>
</table>
| Theme           | Select a color theme specifically for your profile:  
                  System default - use default system settings  
                  Blue - standard blue theme  
                  Dark - alternative dark theme  
                  High-contrast light - light theme with high contrast  
                  High-contrast dark - dark theme with high contrast |
| Auto-login      | Mark this checkbox to make Zabbix remember you and log you in automatically for 30 days.  
                  Browser cookies are used for this.                                                        |
| Auto-logout     | With this checkbox marked you will be logged out automatically, after the set amount of seconds (minimum 90 seconds, maximum 1 day).  
                  Time suffixes are supported, e.g. 90s, 5m, 2h, 1d.                                      |
| Refresh         | You can set how often the information in the pages will be refreshed on the Monitoring menu, except for Dashboard, which uses its own refresh parameters for every widget.  
                  Time suffixes are supported, e.g. 30s, 5m, 2h, 1d.                                      |
| Rows per page   | You can set how many rows will be displayed per page in the lists. Fewer rows (and fewer records to display) mean faster loading times. |
Parameter | Description
--- | ---
URL (after login) | You can set a specific URL to be displayed after the login. Instead of the default Monitoring → Dashboard it can be, for example, the URL of Monitoring → Triggers.

The **Media** tab allows you to specify the **media details** for the user, such as the types, the addresses to use and when to use them to deliver notifications.

![Media tab interface](image)

Only **admin level** users (Admin and Super admin) can change their own media details.

The **Messaging** tab allows you to set **global notifications**.

2 API tokens

API tokens section allows to view tokens assigned to the user, edit token details and **create new tokens**. This section is only available to a user if Manage API tokens action is allowed in the **user role** settings.

![API tokens](image)

You may filter API tokens by name, expiry date, or status (enabled/disabled). Click on the token status in the list to quickly enable/disable a token. You may also mass enable/disable tokens by selecting them in the list and then clicking on the Enable/Disable buttons below the list.

Users cannot view Auth token value of the tokens assigned to them in Zabbix. Auth token value is displayed only once - immediately after creating a token. If it has been lost, the token has to be regenerated.

1 Global notifications

Overview

Global notifications are a way of displaying issues that are currently happening right on the screen you’re at in Zabbix frontend. Without global notifications, working in some other location than Problems or the Dashboard would not show any information regarding issues that are currently happening. Global notifications will display this information regardless of where you are.

Global notifications involve both showing a message and **playing a sound**.

The auto play of sounds may be disabled in recent browser versions by default. In this case, you need to change this setting manually.

Configuration

Global notifications can be enabled per user in the Messaging tab of **profile configuration**.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontend messaging</td>
<td>Mark the checkbox to enable global notifications.</td>
</tr>
<tr>
<td>Message timeout</td>
<td>You can set for how long the message will be displayed. By default, messages will stay on screen for 60 seconds.</td>
</tr>
<tr>
<td></td>
<td><em>Time suffixes</em> are supported, e.g. 30s, 5m, 2h, 1d.</td>
</tr>
<tr>
<td>Play sound</td>
<td>You can set how long the sound will be played.</td>
</tr>
<tr>
<td></td>
<td><strong>Once</strong> - sound is played once and fully.</td>
</tr>
<tr>
<td></td>
<td><strong>10 seconds</strong> - sound is repeated for 10 seconds.</td>
</tr>
<tr>
<td></td>
<td><strong>Message timeout</strong> - sound is repeated while the message is visible.</td>
</tr>
<tr>
<td>Trigger severity</td>
<td>You can set the trigger severities that global notifications and sounds will be activated for. You can also select the sounds appropriate for various severities.</td>
</tr>
<tr>
<td></td>
<td>If no severity is marked then no messages will be displayed at all.</td>
</tr>
<tr>
<td></td>
<td>Also, recovery messages will only be displayed for those severities that are marked. So if you mark Recovery and Disaster, global notifications will be displayed for the problems and the recoveries of disaster severity triggers.</td>
</tr>
<tr>
<td>Show suppressed problems</td>
<td>Mark the checkbox to display notifications for problems which would otherwise be suppressed (not shown) because of host maintenance.</td>
</tr>
</tbody>
</table>

**Global messages displayed**

As the messages arrive, they are displayed in a floating section on the right hand side. This section can be repositioned freely by dragging the section header.
For this section, several controls are available:

- **Snooze** button silences the currently active alarm sound;
- **Mute/Unmute** button switches between playing and not playing the alarm sounds at all.

### 2 Sound in browsers

**Overview**

Sound is used in global notifications.

For the sounds to be played in Zabbix frontend, Frontend messaging must be enabled in the user profile Messaging tab, with all trigger severities checked, and sounds should also be enabled in the global notification pop-up window.

If for some reasons audio cannot be played on the device, the **Mute/Unmute** button in the global notification pop-up window will permanently remain in the “mute” state and the message “Cannot support notification audio for this device.” will be displayed upon hovering over the **Mute/Unmute** button.

Sounds, including the default audio clips, are supported in MP3 format only.

The sounds of Zabbix frontend have been successfully tested in recent Firefox/Opera browsers on Linux and Chrome, Firefox, Microsoft Edge, Opera and Safari browsers on Windows.

The auto play of sounds may be disabled in recent browser versions by default. In this case, you need to change this setting manually.

### 4 Global search

It is possible to search Zabbix frontend for hosts, host groups, templates and template groups.

The search input box is located below the Zabbix logo in the menu. The search can be started by pressing Enter or clicking on the **search icon.**
If there is a host that contains the entered string in any part of the name, a dropdown will appear, listing all such hosts (with the matching part highlighted in orange). The dropdown will also list a host if that host’s visible name is a match to the technical name entered as a search string; the matching host will be listed, but without any highlighting.

Searchable attributes

Hosts can be searched by the following properties:

- Host name
- Visible name
- IP address
- DNS name

Templates can be searched by name or visible name. If you search by a name that is different from the visible name (of a template/host), in the search results it is displayed below the visible name in parentheses.

Host and template groups can be searched by name. Specifying a parent group implicitly selects all nested groups.

Search results

Search results consist of four separate blocks for hosts, host groups, templates and template groups.

It is possible to collapse/expand each individual block. The entry count is displayed at the bottom of each block, for example, Displaying 13 of 13 found. Total entries displayed within one block are limited to 100.

Each entry provides links to monitoring and configuration data. See the full list of links.

For all configuration data (such as items, triggers, graphs) the amount of entities found is displayed by a number next to the entity name, in gray. Note that if there are zero entities, no number is displayed.

Enabled hosts are displayed in blue, disabled hosts in red.

Links available

For each entry the following links are available:

- Hosts
  - Monitoring
    - Latest data
    - Problems
    - Graphs
Overview

Zabbix web frontend can be temporarily disabled in order to prohibit access to it. This can be useful for protecting the Zabbix database from any changes initiated by users, thus protecting the integrity of database.

Zabbix database can be stopped and maintenance tasks can be performed while Zabbix frontend is in maintenance mode.

Users from defined IP addresses will be able to work with the frontend normally during maintenance mode.

Configuration

In order to enable maintenance mode, the maintenance.inc.php file (located in /conf of the Zabbix HTML document directory on the webserver) must be modified to uncomment the following lines:

```php
// Maintenance mode.
define('ZBX_DENY_GUI_ACCESS', 1);

// Array of IP addresses, which are allowed to connect to frontend (optional).
$ZBX_GUI_ACCESS_IP_RANGE = array('127.0.0.1');

// Message shown on warning screen (optional).
$ZBX_GUI_ACCESS_MESSAGE = 'We are upgrading MySQL database till 15:00. Stay tuned...';
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZBX_DENY_GUI_ACCESS</td>
<td>Enable maintenance mode:</td>
</tr>
<tr>
<td></td>
<td>1 - maintenance mode is enabled, disabled otherwise</td>
</tr>
<tr>
<td>ZBX_GUI_ACCESS_IP_RANGE</td>
<td>Array of IP addresses, which are allowed to connect to frontend (optional). For example: array('192.168.1.1', '192.168.1.2')</td>
</tr>
<tr>
<td>ZBX_GUI_ACCESS_MESSAGE</td>
<td>Message you can enter to inform users about the maintenance (optional).</td>
</tr>
</tbody>
</table>

Display
The following screen will be displayed when trying to access the Zabbix frontend while in maintenance mode. The screen is refreshed every 30 seconds in order to return to a normal state without user intervention when the maintenance is over.

![Warning message](image)

IP addresses defined in ZBX_GUI_ACCESS_IP_RANGE will be able to access the frontend as always.

### 6 Page parameters

#### Overview

Most Zabbix web interface pages support various HTTP GET parameters that control what will be displayed. They may be passed by specifying parameter=value pairs after the URL, separated from the URL by a question mark (?) and from each other by ampersands (&).

#### Monitoring → Problems

The following parameters are supported:

- **show** - filter option "Show": 1 - recent problems, 2 - all, 3 - in problem state
- **name** - filter option "Problem": freeform string
- **severities** - filter option "Severity": array of selected severities in a format 'severities[*]=*': (replace * with severity level):
  - 0 - not classified, 1 - information, 2 - warning, 3 - average, 4 - high, 5 - disaster
- **inventory** - filter option "Host inventory": array of inventory fields: [field], [value]
- **evaltype** - filter option "Host inventory": array of inventory fields: [field], [value]
- **tags** - filter option "Tags": array of defined tags: [tag], [operator], [value]
- **show_tags** - filter option "Show tags": 0 - none, 1 - one, 2 - two, 3 - three
- **tag_name_format** - filter option "Tag name": 0 - full name, 1 - shortened, 2 - none
- **tag_priority** - filter option "Tag display priority": comma-separated string of tag display priority
- **show_suppressed** - filter option "Show suppressed problems": should be 'show_suppressed=1' to show
- **unacknowledged** - filter option "Show unacknowledged only": should be 'unacknowledged=1' to show
- **compact_view** - filter option "Compact view": should be 'compact_view=1' to show
- **highlight_row** - filter option "Highlight whole row" (use problem color as background color for every problem row): should be '1' to highlight; can be set only when 'compact_view' is set
- **filter_name** - filter properties option "Name": freeform string
- **filter_show_counter** - filter properties option "Show number of records": 1 - show, 0 - do not show
- **filter_custom_time** - filter properties option "Set custom time period": 1 - set, 0 - do not set
- **sort** - sort-column: clock, host, severity, name
- **sortorder** - sort order or results: DESC - descending, ASC - ascending
- **age_state** - filter option "Age less than": should be 'age_state=1' to enable 'age'. Is used only when 'show' equals 3.
- **age** - filter option "Age less than": days
- **groupids** - filter option "Host groups": array of host groups IDs
- **hostids** - filter option "Hosts": array of host IDs
- **triggerids** - filter option "Triggers": array of trigger IDs
- **show_timeline** - filter option "Show timeline": should be 'show_timeline=1' to show
- **details** - filter option "Show details": should be 'details=1' to show
- **from** - date range start, can be 'relative' (e.g.: now-1m). Is used only when 'filter_custom_time' equals 1.
- **to** - date range end, can be 'relative' (e.g.: now-1m). Is used only when 'filter_custom_time' equals 1.

#### Kiosk mode

The kiosk mode in supported frontend pages can be activated using URL parameters. For example, in dashboards:

- /zabbix.php?action=dashboard.view&kiosk=1 - activate kiosk mode
- /zabbix.php?action=dashboard.view&kiosk=0 - activate normal mode

#### Slideshow

It is possible to activate a slideshow in the dashboard:

- /zabbix.php?action=dashboard.view&slideshow=1 - activate slideshow
7 Definitions

Overview

While many things in the frontend can be configured using the frontend itself, some customizations are currently only possible by editing a definitions file.

This file is defines.inc.php located in /include of the Zabbix HTML document directory.

Parameters

Parameters in this file that could be of interest to users:

- **ZBX_MIN_PERIOD**
  Minimum graph period, in seconds. One minute by default.

- **GRAPH_YAXIS_SIDE_DEFAULT**
  Default location of Y axis in simple graphs and default value for drop down box when adding items to custom graphs. Possible values: 0 - left, 1 - right.
  Default: 0

- **ZBX_SESSION_NAME (available since 4.0.0)**
  String used as the name of the Zabbix frontend session cookie.
  Default: zbx_sessionid

- **ZBX_DATA_CACHE_TTL (available since 5.2.0)**
  TTL timeout in seconds used to invalidate data cache of Vault response. Set 0 to disable Vault response caching.
  Default: 60

- **SUBFILTER_VALUES_PER_GROUP (available since 6.0.5)**
  Number of subfilter values per group (for example, in the latest data subfilter).
  Default: 1000

8 Creating your own theme

Overview

By default, Zabbix provides a number of predefined themes. You may follow the step-by-step procedure provided here in order to create your own. Feel free to share the result of your work with Zabbix community if you created something nice.

Step 1

To define your own theme you'll need to create a CSS file and save it in the assets/styles/ folder (for example, custom-theme.css). You can either copy the files from a different theme and create your theme based on it or start from scratch.

Step 2

Add your theme to the list of themes returned by the APP::getThemes() method. You can do this by overriding the ZBase::getThemes() method in the APP class. This can be done by adding the following code before the closing brace in include/classes/core/APP.php:

```php
public static function getThemes() {
    return array_merge(parent::getThemes(), [
        'custom-theme' => _('Custom theme')
    ]);}
```

Note that the name you specify within the first pair of quotes must match the name of the theme file without extension.

To add multiple themes, just list them under the first theme, for example:
public static function getThemes() {
    return array_merge(parent::getThemes(), [
    'custom-theme' => _('Custom theme'),
    'anothertheme' => _('Another theme'),
    'onemoretheme' => _('One more theme')
    ]);}

Note that every theme except the last one must have a trailing comma.

To change graph colors, the entry must be added in the graph_theme database table.

Step 3
Activate the new theme.

In Zabbix frontend, you may either set this theme to be the default one or change your theme in the user profile.

Enjoy the new look and feel!

9 Debug mode

Overview
Debug mode may be used to diagnose performance problems with frontend pages.

Configuration
Debug mode can be activated for individual users who belong to a user group:

• when configuring a user group;
• when viewing configured user groups.

When Debug mode is enabled for a user group, its users will see a Debug button in the lower right corner of the browser window:

Clicking on the Debug button opens a new window below the page contents which contains the SQL statistics of the page, along with a list of API calls and individual SQL statements:

In case of performance problems with the page, this window may be used to search for the root cause of the problem.

Enabled Debug mode negatively affects frontend performance.

10 Cookies used by Zabbix
This page provides a list of cookies used by Zabbix.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Values</th>
<th>Expires/Max-Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZBX_SESSION_NAME</td>
<td>Frontend session data, stored as JSON encoded by base64</td>
<td>Session (expires when the browsing session ends)</td>
<td>+ (only if HTTPS is enabled on a web server)</td>
</tr>
<tr>
<td>browserwarning</td>
<td>A warning about using an outdated browser should be ignored.</td>
<td>Session (expires when the browsing session ends)</td>
<td></td>
</tr>
<tr>
<td>system-message-ok</td>
<td>A message to show as soon as page is reloaded.</td>
<td>Plain text message</td>
<td></td>
</tr>
<tr>
<td>system-message-error</td>
<td>An error message to show as soon as page is reloaded.</td>
<td>Plain text message</td>
<td></td>
</tr>
</tbody>
</table>

Forcing ‘HttpOnly’ flag on Zabbix cookies by a webserver directive is not supported.

### 11 Time zones

Overview

The frontend time zone can be set globally in the frontend and adjusted for individual users.
If System is selected, the web server time zone will be used for the frontend (including the value of ‘date.timezone’ of php.ini, if set), while Zabbix server will use the time zone of the machine it is running on.

Zabbix server will only use the specified global/user time zone when expanding macros in notifications (e.g. {EVENT.TIME} can expand to a different time zone per user) and for the time limit when notifications are sent (see “When active” setting in user media configuration).

Configuration

The global timezone:

• can be set manually when installing the frontend
• can be modified in Administration → General → GUI

User-level time zone:

• can be set when configuring/updating a user
• can be set by each user in their user profile

12 Rebranding

Overview

There are several ways in which you can customize and rebrand your Zabbix frontend installation:

• replace the Zabbix logo with a desired one
• hide links to Zabbix Support and Zabbix Integrations
• set a custom link to the Help page
• change copyright in the footer

How to

To begin with, you need to create a PHP file and save it as local/conf/brand.conf.php. The contents of the file should be the following:

```php
<?php
return [];
```

This will hide the links to Zabbix Support and Zabbix Integrations.

Custom logo
To use a custom *logo*, add the following line to the array from the previous listing:

```
'BRAND_LOGO' => '{Path to an image on the disk or URL}',
```

With the redesign of the main menu in Zabbix 5.0, there are two additional images of the Zabbix logo that can be overridden:

- **BRAND_LOGO_SIDEBAR** - displayed when the sidebar is expanded
- **BRAND_LOGO_SIDEBAR_COMPACT** - displayed when the sidebar is collapsed

To override:

```
'BRAND_LOGO_SIDEBAR' => '{Path to an image on the disk or URL}',
'BRAND_LOGO_SIDEBAR_COMPACT' => '{Path to an image on the disk or URL}',
```

Any image format supported by modern browsers can be used: JPEG, PNG, SVG, BMP, WebP and GIF.

Custom logos will not be scaled, resized or modified in any way, and will be displayed in their original sizes and proportions, but may be cropped to fit in the corresponding place.

Custom copyright notice

To set a custom copyright notice, add **BRAND_FOOTER** to the array from the first listing. Please be aware that HTML is not supported here. Setting **BRAND_FOOTER** to an empty string will hide the copyright notes completely (but the footer will stay in place).

```
'BRAND_FOOTER' => '{text}',
```

Custom help location

To replace the default Help link with a link of your choice, add **BRAND_HELP_URL** to the array from the first listing.

```
'BRAND_HELP_URL' => '{URL}',
```

File example

```php
<?php
return [
    'BRAND_LOGO' => './images/custom_logo.png',
    'BRAND_LOGO_SIDEBAR' => './images/custom_logo_sidebar.png',
    'BRAND_LOGO_SIDEBAR_COMPACT' => './images/custom_logo_sidebar_compact.png',
    'BRAND_FOOTER' => '© Zabbix',
    'BRAND_HELP_URL' => 'https://www.example.com/help/
];
```

19. API

**Overview**  Zabbix API allows you to programmatically retrieve and modify the configuration of Zabbix and provides access to historical data. It is widely used to:

- Create new applications to work with Zabbix;
- Integrate Zabbix with third party software;
- Automate routine tasks.

The Zabbix API is a web based API and is shipped as part of the web frontend. It uses the JSON-RPC 2.0 protocol which means two things:

- The API consists of a set of separate methods;
- Requests and responses between the clients and the API are encoded using the JSON format.

More info about the protocol and JSON can be found in the [JSON-RPC 2.0 specification](https://www.jsonrpc.org/specification) and the [JSON format homepage](https://www.json.org).

**Structure**  The API consists of a number of methods that are nominally grouped into separate APIs. Each of the methods performs one specific task. For example, the `host.create` method belongs to the host API and is used to create new hosts. Historically, APIs are sometimes referred to as “classes”.

Most APIs contain at least four methods: `get`, `create`, `update` and `delete` for retrieving, creating, updating and deleting data respectively, but some of the APIs may provide a totally different set of methods.
Performing requests  Once you’ve set up the frontend, you can use remote HTTP requests to call the API. To do that you need to send HTTP POST requests to the api_jsonrpc.php file located in the frontend directory. For example, if your Zabbix frontend is installed under http://company.com/zabbix, the HTTP request to call the apiinfo.version method may look like this:

```
POST http://company.com/zabbix/api_jsonrpc.php HTTP/1.1
Content-Type: application/json-rpc

{"jsonrpc":"2.0","method":"apiinfo.version","id":1,"auth":null,"params":{}}
```

The request must have the Content-Type header set to one of these values: application/json-rpc, application/json or application/jsonrequest.

You can use any HTTP client or a JSON-RPC testing tool to perform API requests manually, but for developing applications we suggest you use one of the community maintained libraries.

Example workflow  The following section will walk you through some usage examples in more detail.

Authentication  Before you can access any data inside of Zabbix you’ll need to log in and obtain an authentication token. This can be done using the user.login method. Let us suppose that you want to log in as a standard Admin user. Then your JSON request will look like this:

```
{
   "jsonrpc": "2.0",
   "method": "user.login",
   "params": {
      "user": "Admin",
      "password": "zabbix"
   },
   "id": 1,
   "auth": null
}
```

Let’s take a closer look at the request object. It has the following properties:

- **jsonrpc** - the version of the JSON-RPC protocol used by the API; the Zabbix API implements JSON-RPC version 2.0;
- **method** - the API method being called;
- **params** - parameters that will be passed to the API method;
- **id** - an arbitrary identifier of the request;
- **auth** - a user authentication token; since we don’t have one yet, it’s set to null.

If you provided the credentials correctly, the response returned by the API will contain the user authentication token:

```
{
   "jsonrpc": "2.0",
   "result": "0424bd59b807674191e7d77572075f33",
   "id": 1
}
```

The response object in turn contains the following properties:

- **jsonrpc** - again, the version of the JSON-RPC protocol;
- **result** - the data returned by the method;
- **id** - identifier of the corresponding request.

Retrieving hosts  We now have a valid user authentication token that can be used to access the data in Zabbix. For example, let’s use the host.get method to retrieve the IDs, host names and interfaces of all configured hosts:

```
{
   "jsonrpc": "2.0",
   "method": "host.get",
   "params": {
      "output": ["hostid","host"],
      "selectInterfaces": ["interfaceid","ip"]
   }
}
```
Note that the auth property is now set to the authentication token we've obtained by calling `user.login`.

The response object will contain the requested data about the hosts:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "host": "Zabbix server",
            "interfaces": [
                {
                    "interfaceid": "1",
                    "ip": "127.0.0.1"
                }
            ]
        }
    ],
    "id": 2
}
```

For performance reasons we recommend to always list the object properties you want to retrieve and avoid retrieving everything.

**Creating a new item**  Let's create a new item on "Zabbix server" using the data we've obtained from the previous `host.get` request. This can be done by using the `item.create` method:

```json
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Free disk space on /home/joe/",
        "key_": "vfs.fs.size[/home/joe/,free]",
        "hostid": "10084",
        "type": 0,
        "value_type": 3,
        "interfaceid": "1",
        "delay": 30
    },
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 3
}
```

A successful response will contain the ID of the newly created item, which can be used to reference the item in the following requests:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": ["24759"
    ]
},
    "id": 3
}
```

The `item.create` method as well as other create methods can also accept arrays of objects and create multiple items with one API call.

**Creating multiple triggers**  So if create methods accept arrays, we can add multiple triggers like so:
A successful response will contain the IDs of the newly created triggers:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17369",
            "17370"
        ]
    },
    "id": 4
}
```

**Updating an item** Enable an item, that is, set its status to "0":

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "10092",
        "status": 0
    },
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 5
}
```

A successful response will contain the ID of the updated item:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "10092"
        ]
    },
    "id": 5
}
```

The `item.update` method as well as other update methods can also accept arrays of objects and update multiple items with one API call.

**Updating multiple triggers** Enable multiple triggers, that is, set their status to 0:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": [
        {
            "description": "Processor load is too high on {HOST.NAME}"
        },
        {
            "description": "Too many processes on {HOST.NAME}"
        }
    ],
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 4
}
```
A successful response will contain the IDs of the updated triggers:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938",
            "13939"
        ],
        "id": 6
    }
}
```

This is the preferred method of updating. Some API methods like `host.massupdate` allow to write more simple code, but it's not recommended to use those methods, since they will be removed in the future releases.

**Error handling**  Up to that point everything we've tried has worked fine. But what happens if we try to make an incorrect call to the API? Let's try to create another host by calling `host.create` but omitting the mandatory `groups` parameter.

```json
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "Linux server",
        "interfaces": [
            {
                "type": 1,
                "main": 1,
                "useip": 1,
                "ip": "192.168.3.1",
                "dns": "",
                "port": "10050"
            }
        ],
        "id": 7,
        "auth": "0424bd59b807674191e7d77572075f33"
    }
}
```

The response will then contain an error message:

```json
{
    "jsonrpc": "2.0",
    "error": {
        "code": -32602,
        "message": "Invalid params."
    }
}
```

If an error occurred, instead of the `result` property, the response object will contain an `error` property with the following data:

- code - an error code;
- message - a short error summary;
- data - a more detailed error message.

Errors can occur in different cases, such as, using incorrect input values, a session timeout or trying to access unexisting objects. Your application should be able to gracefully handle these kinds of errors.

**API versions**  To simplify API versioning, since Zabbix 2.0.4, the version of the API matches the version of Zabbix itself. You can use the `apiinfo.version` method to find out the version of the API you’re working with. This can be useful for adjusting your application to use version-specific features.

We guarantee feature backward compatibility inside of a major version. When making backward incompatible changes between major releases, we usually leave the old features as deprecated in the next release, and only remove them in the release after that. Occasionally, we may remove features between major releases without providing any backward compatibility. It is important that you never rely on any deprecated features and migrate to newer alternatives as soon as possible.

You can follow all of the changes made to the API in the [API changelog](#).

**Further reading**  You now know enough to start working with the Zabbix API, but don’t stop here. For further reading we suggest you have a look at the list of available APIs.

**Method reference**

This section provides an overview of the functions provided by the Zabbix API and will help you find your way around the available classes and methods.

**Monitoring**  The Zabbix API allows you to access history and other data gathered during monitoring.

High availability cluster

Retrieve a list of server nodes and their status.

[High availability cluster API](#)

History

Retrieve historical values gathered by Zabbix monitoring processes for presentation or further processing.

[History API](#)

Trends

Retrieve trend values calculated by Zabbix server for presentation or further processing.

[Trend API](#)

Events

Retrieve events generated by triggers, network discovery and other Zabbix systems for more flexible situation management or third-party tool integration.

[Event API](#)

Problems

Retrieve problems according to the given parameters.

[Problem API](#)

Service monitoring

Create a hierarchy representation of monitored IT infrastructure/business services data.

[Service API](#)

Service Level Agreement

Define Service Level Objectives (SLO), retrieve detailed Service Level Indicators (SLI) information about service performance.

[SLA API](#)

Tasks

Interact with Zabbix server task manager, creating tasks and retrieving response.
Task API

Configuration  The Zabbix API allows you to manage the configuration of your monitoring system.

Hosts and host groups
Manage host groups, hosts and everything related to them, including host interfaces, host macros and maintenance periods.
Host API | Host group API | Host interface API | User macro API | Value map API | Maintenance API

Items
Define items to monitor.
Item API

Triggers
Configure triggers to notify you about problems in your system. Manage trigger dependencies.
Trigger API

Graphs
Edit graphs or separate graph items for better presentation of the gathered data.
Graph API | Graph item API

Templates
Manage templates and link them to hosts or other templates.
Template API | Value map API

Export and import
Export and import Zabbix configuration data for configuration backups, migration or large-scale configuration updates.
Configuration API

Low-level discovery
Configure low-level discovery rules as well as item, trigger and graph prototypes to monitor dynamic entities.
LLD rule API | Item prototype API | Trigger prototype API | Graph prototype API | Host prototype API

Event correlation
Create custom event correlation rules.
Correlation API

Actions and alerts
Define actions and operations to notify users about certain events or automatically execute remote commands. Gain access to information about generated alerts and their receivers.
Action API | Alert API

Services
Manage services for service-level monitoring and retrieve detailed SLA information about any service.
Service API

Dashboards
Manage dashboards and make scheduled reports based on them.
Dashboard API | Template dashboard API | Report API

Maps
Configure maps to create detailed dynamic representations of your IT infrastructure.
Map API

Web monitoring
Configure web scenarios to monitor your web applications and services.
Web scenario API
Network discovery

Manage network-level discovery rules to automatically find and monitor new hosts. Gain full access to information about discovered services and hosts.

Discovery rule API | Discovery check API | Discovered host API | Discovered service API

Administration  With the Zabbix API you can change administration settings of your monitoring system.

Users

Add users that will have access to Zabbix, assign them to user groups and grant permissions. Make roles for granular management of user rights. Track configuration changes each user has done. Configure media types and multiple ways users will receive alerts.

User API | User group API | User role API | Media type API | Audit log API

General

Change certain global configuration options.

Autoregistration API | Icon map API | Image API | User macro API | Settings API | Housekeeping API

Regular expressions

Manage global regular expressions.

Regular expression API

Proxies

Manage the proxies used in your distributed monitoring setup.

Proxy API

Authentication

Change authentication configuration options.

Authentication API

API Tokens

Manage authorization tokens.

Token API

Scripts

Configure and execute scripts to help you with your daily tasks.

Script API

API information  Retrieve the version of the Zabbix API so that your application could use version-specific features.

API info API

Action

This class is designed to work with actions.

Object references:

- Action
- Action condition
- Action operation

Available methods:

- action.create - create new actions
- action.delete - delete actions
- action.get - retrieve actions
- action.update - update actions
**Action object**

The following objects are directly related to the action API.

**Action**

The action object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>actionid</td>
<td>string</td>
<td>(readonly) ID of the action.</td>
</tr>
<tr>
<td>esc_period</td>
<td>string</td>
<td>Default operation step duration. Must be at least 60 seconds. Accepts seconds, time unit with suffix and user macro.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>Note that escalations are supported only for trigger, internal and service actions, and only in normal operations.</td>
</tr>
<tr>
<td>eventsource</td>
<td>integer</td>
<td>(constant) Type of events that the action will handle.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>Refer to the event “source” property for a list of supported event types.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the action.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>Refer to the event “source” property for a list of supported event types.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Whether the action is enabled or disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) enabled; 1 - disabled.</td>
</tr>
<tr>
<td>pause_suppressed</td>
<td>integer</td>
<td>Whether to pause escalation during maintenance periods or not.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - Don't pause escalation; 1 - (default) Pause escalation.</td>
</tr>
<tr>
<td>notify_if_canceled</td>
<td>integer</td>
<td>Whether to notify when escalation is canceled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - Don't notify when escalation is canceled; 1 - (default) Notify when escalation is canceled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note that this parameter is valid for trigger actions only.</td>
</tr>
</tbody>
</table>

**Action operation**

The action operation object defines an operation that will be performed when an action is executed. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operationid</td>
<td>string</td>
<td>(readonly) ID of the action operation.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>operationtype</strong></td>
<td>integer</td>
<td>Type of operation.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - send message;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - global script;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - add host;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - remove host;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - add to host group;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - remove from host group;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 - link to template;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 - unlink from template;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 - enable host;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 - disable host;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 - set host inventory mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note that only types ‘0’ and ‘1’ are supported for trigger and service</td>
</tr>
<tr>
<td></td>
<td></td>
<td>actions, only ‘0’ is supported for internal actions. All types are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>supported for discovery and autoregistration actions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Only types are supported for discovery and autoregistration actions.</td>
</tr>
<tr>
<td>actionid</td>
<td>string</td>
<td>(readonly) ID of the action that the operation belongs to.</td>
</tr>
<tr>
<td>esc_period</td>
<td>string</td>
<td>Duration of an escalation step in seconds. Must be greater than 60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>seconds. Accepts seconds, time unit with suffix and user macro. If set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to 0 or 0s, the default action escalation period will be used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0s.</td>
</tr>
<tr>
<td>esc_step_from</td>
<td>integer</td>
<td>Step to start escalation from.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1.</td>
</tr>
<tr>
<td>esc_step_to</td>
<td>integer</td>
<td>Step to end escalation at.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1.</td>
</tr>
<tr>
<td>evaltype</td>
<td>integer</td>
<td>Operation condition evaluation method.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) AND / OR;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - AND;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - OR.</td>
</tr>
<tr>
<td>opcommand</td>
<td>object</td>
<td>Object containing data on global script run by the operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each object has one following property: scriptid - (string) ID of the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>script.</td>
</tr>
<tr>
<td>opcommand_grp</td>
<td>array</td>
<td>Host groups to run global scripts on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each object has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>opcommand_grpid - (string, readonly) ID of the object;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>operationid - (string, readonly) ID of the operation;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>groupid - (string) ID of the host group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required for global script operations if opcommand_hst is not set.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>opcommand_hst</td>
<td>array</td>
<td>Host to run global scripts on.</td>
</tr>
</tbody>
</table>
|                   |         | Each object has the following properties:  
|                   |         | opcommand_hstid - (string, readonly) ID of the object;  
|                   |         | operationid - (string, readonly) ID of the operation;  
|                   |         | hostid - (string) ID of the host; if set to 0 the command will be run on the current host. |
| opconditions      | array   | Required for global script operations if opcommand_grp is not set.  
|                   |         | Operation conditions used for trigger actions. |
| opgroup           | array   | The operation condition object is described in detail below. |
|                   |         | Each object has the following properties:  
|                   |         | operationid - (string, readonly) ID of the operation;  
|                   |         | grupid - (string) ID of the host group. |
|                   |         | Required for "add to host group" and "remove from host group" operations. |
| opmessage         | object  | The operation message object is described in detail below. |
|                   |         | Each object contains the data about the message sent by the operation. |
| opmessage_grp     | array   | Required for message operations.  
|                   |         | User groups to send messages to. |
|                   |         | Each object has the following properties:  
|                   |         | operationid - (string, readonly) ID of the operation;  
|                   |         | usgrpid - (string) ID of the user group. |
| opmessage_usr     | array   | Required for message operations if opmessage_usr is not set.  
|                   |         | Users to send messages to. |
|                   |         | Each object has the following properties:  
|                   |         | operationid - (string, readonly) ID of the operation;  
|                   |         | userid - (string) ID of the user. |
| optemplate        | array   | Required for message operations if opmessage_grp is not set.  
|                   |         | Templates to link the hosts to. |
|                   |         | Each object has the following properties:  
|                   |         | operationid - (string, readonly) ID of the operation;  
|                   |         | templateid - (string) ID of the template. |
| opinventory        | object  | Required for "link to template" and "unlink from template" operations.  
|                   |         | Inventory mode set host to. |
|                   |         | Object has the following properties:  
|                   |         | operationid - (string, readonly) ID of the operation;  
|                   |         | inventory_mode - (string) Inventory mode. |
|                   |         | Required for "Set host inventory mode" operations. |

Action operation message

The operation message object contains data about the message that will be sent by the operation.
### Property

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default_msg</td>
<td>integer</td>
<td>Whether to use the default action message text and subject. Possible values: 0 - use the data from the operation; 1 - (default) use the data from the media type.</td>
</tr>
<tr>
<td>mediatypeid</td>
<td>string</td>
<td>ID of the media type that will be used to send the message.</td>
</tr>
<tr>
<td>message</td>
<td>string</td>
<td>Operation message text.</td>
</tr>
<tr>
<td>subject</td>
<td>string</td>
<td>Operation message subject.</td>
</tr>
</tbody>
</table>

### Action operation condition

The action operation condition object defines a condition that must be met to perform the current operation. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>opconditionid</td>
<td>string</td>
<td>(readonly) ID of the action operation condition</td>
</tr>
<tr>
<td>conditiontype</td>
<td>integer</td>
<td>Type of condition. Possible values: 14 - event acknowledged.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Value to compare with.</td>
</tr>
<tr>
<td>operationid</td>
<td>string</td>
<td>(readonly) ID of the operation.</td>
</tr>
<tr>
<td>operator</td>
<td>integer</td>
<td>Condition operator. Possible values: 0 - (default) =.</td>
</tr>
</tbody>
</table>

The following operators and values are supported for each operation condition type.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Condition name</th>
<th>Supported operators</th>
<th>Expected value</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Event acknowledged</td>
<td>=</td>
<td>Whether the event is acknowledged.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 - not acknowledged;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - acknowledged.</td>
</tr>
</tbody>
</table>

### Action recovery operation

The action recovery operation object defines an operation that will be performed when a problem is resolved. Recovery operations are possible for trigger, internal and service actions. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operationid</td>
<td>string</td>
<td>(readonly) ID of the action operation.</td>
</tr>
<tr>
<td>operationtype</td>
<td>integer</td>
<td>Type of operation. Possible values for trigger and service actions: 0 - send message; 1 - global script; 11 - notify all involved. Possible values for internal actions: 0 - send message; 11 - notify all involved.</td>
</tr>
<tr>
<td>actionid</td>
<td>string</td>
<td>(readonly) ID of the action that the recovery operation belongs to.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>opcommand</td>
<td>object</td>
<td>Object containing data on global action type script run by the operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each object has one following property: scriptid - (string) ID of the action type script. Required for global script operations.</td>
</tr>
<tr>
<td>opcommand_grp</td>
<td>array</td>
<td>Host group to run global scripts on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each object has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>opcommand_grpid - (string, readonly) ID of the object;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>operationid - (string, readonly) ID of the operation;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>groupid - (string) ID of the host group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required for global script operations if opcommand_hst is not set.</td>
</tr>
<tr>
<td>opcommand_hst</td>
<td>array</td>
<td>Host to run global scripts on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each object has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>opcommand_hstid - (string, readonly) ID of the object;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>operationid - (string, readonly) ID of the operation;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hostid - (string) ID of the host; if set to 0 the command will be run on the current host.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required for global script operations if opcommand_grp is not set.</td>
</tr>
<tr>
<td>opmessage</td>
<td>object</td>
<td>Object containing the data about the message sent by the recovery operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The operation message object is described in detail above.</td>
</tr>
<tr>
<td>opmessage_grp</td>
<td>array</td>
<td>User groups to send messages to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each object has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>operationid - (string, readonly) ID of the operation;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>usrgrpid - (string) ID of the user group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required for message operations if opmessage_usr is not set.</td>
</tr>
<tr>
<td>opmessage_usr</td>
<td>array</td>
<td>Users to send messages to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each object has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>operationid - (string, readonly) ID of the operation;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>userid - (string) ID of the user.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required for message operations if opmessage_grp is not set.</td>
</tr>
</tbody>
</table>

**Action update operation**

The action update operation object defines an operation that will be performed when a problem is updated (commented upon, acknowledged, severity changed, or manually closed). Update operations are possible for trigger and service actions. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operationid</td>
<td>string</td>
<td>(readonly) ID of the action operation.</td>
</tr>
<tr>
<td>operationtype</td>
<td>integer</td>
<td>Type of operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for trigger and service actions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - send message;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - global script;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 - notify all involved.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>opcommand</td>
<td>object</td>
<td>Object containing data on global action type script run by the operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each object has one following property: scriptid - (string) ID of the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>action type script. Required for global script operations.</td>
</tr>
<tr>
<td>opcommand_grp</td>
<td>array</td>
<td>Host group to run global scripts on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each object has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>groupid - (string) ID of the host group. Required for global script</td>
</tr>
<tr>
<td></td>
<td></td>
<td>operations if opcommand_hst is not set.</td>
</tr>
<tr>
<td>opcommand_hst</td>
<td>array</td>
<td>Host to run global scripts on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each object has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hostid - (string) ID of the host; if set to 0 the command will be run on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the current host. Required for global script operations if opcommand_grp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is not set.</td>
</tr>
<tr>
<td>opmessage</td>
<td>object</td>
<td>The operation message object is described in detail above.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Object containing the data about the message sent by the update operation.</td>
</tr>
<tr>
<td>opmessage_grp</td>
<td>array</td>
<td>User groups to send messages to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each object has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>usrgrpid - (string) ID of the user gruop. Required only for send message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>operations if opmessage_usr is not set. Is ignored for send update message</td>
</tr>
<tr>
<td></td>
<td></td>
<td>operations.</td>
</tr>
<tr>
<td>opmessage_usr</td>
<td>array</td>
<td>Users to send messages to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Each object has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>userid - (string) ID of the user.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required only for send message operations if opmessage_grp is not set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Is ignored for send update message operations.</td>
</tr>
</tbody>
</table>

### Action filter

The action filter object defines a set of conditions that must be met to perform the configured action operations. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conditions</td>
<td>array</td>
<td>Set of filter conditions to use for filtering results.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>evaltype</td>
<td>integer</td>
<td>Filter condition evaluation method.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - and/or;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - and;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - or;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - custom expression.</td>
</tr>
<tr>
<td>eval_formula</td>
<td>string</td>
<td>(readonly) Generated expression that will be used for evaluating filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>conditions. The expression contains IDs that reference specific filter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>conditions by its formulaid. The value of eval_formula is equal to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the value of formulaid for filters with a custom expression.</td>
</tr>
</tbody>
</table>
### Formula

- **Property**: formula
- **Type**: string
- **Description**: User-defined expression to be used for evaluating conditions of filters with a custom expression. The expression must contain IDs that reference specific filter conditions by its formula. The IDs used in the expression must exactly match the ones defined in the filter conditions: no condition can remain unused or omitted.

Required for custom expression filters.

### Action filter condition

The action filter condition object defines a specific condition that must be checked before running the action operations.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conditionid</td>
<td>string</td>
<td>(readonly) ID of the action condition.</td>
</tr>
<tr>
<td>conditiontype</td>
<td>integer</td>
<td>Type of condition. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>trigger actions: 0 - host group; 1 - host; 2 - trigger; 3 - trigger name; 4 - trigger severity;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 - time period; 13 - host template; 16 - problem is suppressed; 25 - event tag; 26 - event tag value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>discovery actions: 7 - host IP; 8 - discovered service type; 9 - discovered service port; 10 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>discovery status; 11 - uptime or downtime duration; 12 - received value; 18 - discovery rule;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19 - discovery check; 20 - proxy; 21 - discovery object. Possible values for autoregistration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>actions: 20 - proxy; 22 - host name; 24 - host metadata. Possible values for internal actions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - host group; 1 - host; 13 - host template; 23 - event type; 25 - event tag; 26 - event tag value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>service actions: 25 - event tag; 26 - event tag value; 27 - service; 28 - service name.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Value to compare with.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>value2</td>
<td>string</td>
<td>Secondary value to compare with. Required for trigger, internal and service actions when condition type is 26.</td>
</tr>
<tr>
<td>actionid</td>
<td>string</td>
<td>(readonly) ID of the action that the condition belongs to.</td>
</tr>
<tr>
<td>formulaid</td>
<td>string</td>
<td>Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.</td>
</tr>
<tr>
<td>operator</td>
<td>integer</td>
<td>Condition operator.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) equals;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - does not equal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - contains;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - does not contain;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - in;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - is greater than or equals;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 - is less than or equals;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 - not in;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 - matches;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 - does not match;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 - Yes;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 - No.</td>
</tr>
</tbody>
</table>

To better understand how to use filters with various types of expressions, see examples on the `action.get` and `action.create` method pages.

The following operators and values are supported for each condition type.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Condition name</th>
<th>Supported operators</th>
<th>Expected value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Host group</td>
<td>equals, does not equal</td>
<td>Host group ID.</td>
</tr>
<tr>
<td>1</td>
<td>Host</td>
<td>equals, does not equal</td>
<td>Host ID.</td>
</tr>
<tr>
<td>2</td>
<td>Trigger</td>
<td>equals, does not equal</td>
<td>Trigger ID.</td>
</tr>
<tr>
<td>3</td>
<td>Trigger name</td>
<td>contains, does not contain</td>
<td>Trigger name.</td>
</tr>
<tr>
<td>4</td>
<td>Trigger severity</td>
<td>equals, does not equal, is greater than or equals, is less than or equals</td>
<td>Trigger severity. Refer to the trigger “severity” property for a list of supported trigger severities.</td>
</tr>
<tr>
<td>5</td>
<td>Trigger value</td>
<td>equals</td>
<td>Trigger value. Refer to the trigger “value” property for a list of supported trigger values.</td>
</tr>
<tr>
<td>6</td>
<td>Time period</td>
<td>in, not in</td>
<td>Time when the event was triggered as a <code>time period</code>.</td>
</tr>
<tr>
<td>7</td>
<td>Host IP</td>
<td>equals, does not equal</td>
<td>One or several IP ranges to check separated by commas. Refer to the <code>network discovery configuration</code> section for more information on supported formats of IP ranges.</td>
</tr>
<tr>
<td>8</td>
<td>Discovered service type</td>
<td>equals, does not equal</td>
<td>Type of discovered service. The type of service matches the type of the discovery check used to detect the service. Refer to the <code>discovery check “type” property</code> for a list of supported types.</td>
</tr>
<tr>
<td>9</td>
<td>Discovered service port</td>
<td>equals, does not equal</td>
<td>One or several port ranges separated by commas.</td>
</tr>
<tr>
<td>Condition</td>
<td>Condition name</td>
<td>Supported operators</td>
<td>Expected value</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
<td>---------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>10</td>
<td>Discovery status</td>
<td>equals</td>
<td>Status of a discovered object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 - host or service up;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - host or service down;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 - host or service discovered;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 - host or service lost.</td>
</tr>
<tr>
<td>11</td>
<td>Uptime or downtime duration</td>
<td>is greater than or equals, is less than or equals</td>
<td>Time indicating how long has the discovered object been in the current status in seconds.</td>
</tr>
<tr>
<td>12</td>
<td>Received values</td>
<td>equals, does not equal, is greater than or equals, is less than or equals, contains, does not contain</td>
<td>Value returned when performing a Zabbix agent, SNMPv1, SNMPv2 or SNMPv3 discovery check.</td>
</tr>
<tr>
<td>13</td>
<td>Host template</td>
<td>equals, does not equal</td>
<td>Linked template ID.</td>
</tr>
<tr>
<td>16</td>
<td>Problem is suppressed</td>
<td>Yes, No</td>
<td>No value required: using the “Yes” operator means that problem must be suppressed, “No” - not suppressed.</td>
</tr>
<tr>
<td>18</td>
<td>Discovery rule</td>
<td>equals, does not equal</td>
<td>ID of the discovery rule.</td>
</tr>
<tr>
<td>19</td>
<td>Discovery check</td>
<td>equals, does not equal</td>
<td>ID of the discovery check.</td>
</tr>
<tr>
<td>20</td>
<td>Proxy</td>
<td>equals, does not equal</td>
<td>ID of the proxy.</td>
</tr>
<tr>
<td>21</td>
<td>Discovery object</td>
<td>equals</td>
<td>Type of object that triggered the discovery event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - discovered host;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 - discovered service.</td>
</tr>
<tr>
<td>22</td>
<td>Host name</td>
<td>contains, does not contain, matches, does not match</td>
<td>Host name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Using a regular expression is supported for operators matches and does not match in autoregistration conditions.</td>
</tr>
<tr>
<td>23</td>
<td>Event type</td>
<td>equals</td>
<td>Specific internal event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 - item in “not supported” state;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 - item in “normal” state;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 - LLD rule in “not supported” state;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 - LLD rule in “normal” state;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 - trigger in “unknown” state;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 - trigger in “normal” state.</td>
</tr>
<tr>
<td>24</td>
<td>Host metadata</td>
<td>contains, does not contain, matches, does not match</td>
<td>Metadata of the auto-registered host.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Using a regular expression is supported for operators matches and does not match.</td>
</tr>
<tr>
<td>25</td>
<td>Tag</td>
<td>equals, does not equal, contains, does not contain</td>
<td>Event tag.</td>
</tr>
<tr>
<td>26</td>
<td>Tag value</td>
<td>equals, does not equal, contains, does not contain</td>
<td>Event tag value.</td>
</tr>
<tr>
<td>27</td>
<td>Service</td>
<td>equals, does not equal</td>
<td>Service ID.</td>
</tr>
</tbody>
</table>
### action.create

**Description**

**object action.create(object/array actions)**

This method allows to create new actions.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

**Parameters**

(object/array) Actions to create.

Additionally to the [standard action properties](#), the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>object</td>
<td>Action filter object for the action.</td>
</tr>
<tr>
<td>operations</td>
<td>array</td>
<td>Action operations to create for the action.</td>
</tr>
<tr>
<td>recovery_operations</td>
<td>array</td>
<td>Action recovery operations to create for the action.</td>
</tr>
<tr>
<td>update_operations</td>
<td>array</td>
<td>Action update operations to create for the action.</td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns an object containing the IDs of the created actions under the actionids property. The order of the returned IDs matches the order of the passed actions.

**Examples**

**Create a trigger action**

Create an action that will be run when a trigger from host "10084" that has the word "memory" in its name goes into problem state. The action must first send a message to all users in user group "7". If the event is not resolved in 4 minutes, it will run script "3" on all hosts in group "2". On trigger recovery it will notify all users who received any messages regarding the problem before. On trigger update, message with custom subject and body will be sent to all who left acknowledgments and comments via all media types.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Trigger action",
        "eventsource": 0,
        "status": 0,
        "esc_period": "2m",
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10084"
                },
                {
                    "conditiontype": 3,
                    "operator": 2,
                    "value": "memory"
                }
            ]
        }
    }
}
```
{"operations": [
   {
      "operationtype": 0,
      "esc_period": "0s",
      "esc_step_from": 1,
      "esc_step_to": 2,
      "evaltype": 0,
      "opmessage_grp": [
         {
            "usrgrpid": "7"
         }
      ],
      "opmessage": {
         "default_msg": 1,
         "mediatypid": "1"
      }
   },
   {
      "operationtype": 1,
      "esc_step_from": 3,
      "esc_step_to": 4,
      "evaltype": 0,
      "opconditions": [
         {
            "conditiontype": 14,
            "operator": 0,
            "value": "0"
         }
      ],
      "opcommand_grp": [
         {
            "groupid": "2"
         }
      ],
      "opcommand": {
         "scriptid": "3"
      }
   }
],
"recovery_operations": [
   {
      "operationtype": "11",
      "opmessage": {
         "default_msg": 1
      }
   }
],
"update_operations": [
   {
      "operationtype": "12",
      "opmessage": {
         "default_msg": 0,
         "message": "Custom update operation message body",
         "subject": "Custom update operation message subject"
      }
   }
],
"pause_suppressed": "0",
"notify_if_canceled": "0"
},
"auth": "038e1d7b1735c6a5436ee9eae095879e"}
Create a discovery action

Create an action that will link discovered hosts to template "10091".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Discovery action",
        "eventsource": 1,
        "status": 0,
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "conditiontype": 21,
                    "operator": 0,
                    "value": "1"
                },
                {
                    "conditiontype": 10,
                    "operator": 0,
                    "value": "2"
                }
            ]
        },
        "operations": [
            {
                "operationtype": 6,
                "optemplate": [
                    {
                        "templateid": "10091"
                    }
                ]
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "18"
        ]
    }
}
```
Using a custom expression filter

Create a trigger action that will use a custom filter condition. The action must send a message for each trigger with severity higher or equal to "Warning" for hosts "10084" and "10106". The formula IDs "A", "B" and "C" have been chosen arbitrarily.

Request:

```json
{  
  "jsonrpc": "2.0",
  "method": "action.create",
  "params": { 
    "name": "Trigger action",
    "eventsource": 0,
    "status": 0,
    "esc_period": "2m",
    "filter": { 
      "evaltype": 3,
      "formula": "A and (B or C)",
      "conditions": [ 
        { 
          "conditiontype": 4,
          "operator": 5,
          "value": "2",
          "formulaid": "A"
        },
        { 
          "conditiontype": 1,
          "operator": 0,
          "value": "10084",
          "formulaid": "B"
        },
        { 
          "conditiontype": 1,
          "operator": 0,
          "value": "10106",
          "formulaid": "C"
        }
      ]
    },
    "operations": [ 
      { 
        "operationtype": 0,
        "esc_period": "0s",
        "esc_step_from": 1,
        "esc_step_to": 2,
        "evaltype": 0,
        "opmessage_grp": [ 
          { 
            "usrgrpid": "7"
          }
        ],
        "opmessage": { 
          "default_msg": 1,
          "mediatypeid": "1"
        }
      }
    ],
    "pause_suppressed": "0",
    "notify_if_canceled": "0"
  }
}
```
Create agent autoregistration rule

Add a host to host group "Linux servers" when hostname contains "SRV" or metadata contains "CentOS".

Request:

```json
{  "jsonrpc": "2.0",  "method": "action.create",  "params": {    "name": "Register Linux servers",    "eventsource": "2",    "status": "0",    "filter": {      "evaltype": "2",      "conditions": [        {          "conditiontype": "22",          "operator": "2",          "value": "SRV"        },        {          "conditiontype": "24",          "operator": "2",          "value": "CentOS"        }      ]    },    "operations": [      {        "operationtype": "4",        "opgroup": [          {            "groupid": "2"          }        ]      }    ]  },  "auth": "038e1d7b1735c6a5436ee9eae096879e",  "id": 1}
```

Response:

```json
{  "jsonrpc": "2.0",  "result": {    "actionids": [18]  },  "id": 1}
```

Response:

```json
{  "jsonrpc": "2.0",  "result": {    "actionids": [19]  },  "id": 1}
```
See also

- Action filter
- Action operation

Source

CAction::create() in ui/include/classes/api/services/CAction.php.

**action.delete**

Description

object action.delete(array actionIds)

This method allows to delete actions.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the actions to delete.

Return values

(object) Returns an object containing the IDs of the deleted actions under the actionids property.

Examples

Delete multiple actions

Delete two actions.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "action.delete",
   "params": [
      "17",
      "18"
   ],
   "auth": "3a57200802b24cda67c4e4010b50c065",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "actionids": [
         "17",
         "18"
      ],
      "id": 1
   }
}
```

Source

CAction::delete() in ui/include/classes/api/services/CAction.php.
**action.get**

**Description**

integer/array  action.get(object parameters)

The method allows to retrieve actions according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>actionids</td>
<td>string/array</td>
<td>Return only actions with the given IDs.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only actions that use the given host groups in action conditions.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only actions that use the given hosts in action conditions.</td>
</tr>
<tr>
<td>triggers</td>
<td>string/array</td>
<td>Return only actions that use the given triggers in action conditions.</td>
</tr>
<tr>
<td>mediatypeids</td>
<td>string/array</td>
<td>Return only actions that use the given mediatypes to send messages.</td>
</tr>
<tr>
<td>usrgrpids</td>
<td>string/array</td>
<td>Return only actions that are configured to send messages to the given user groups.</td>
</tr>
<tr>
<td>userids</td>
<td>string/array</td>
<td>Return only actions that are configured to send messages to the given users.</td>
</tr>
<tr>
<td>scriptids</td>
<td>string/array</td>
<td>Return only actions that are configured to run the given scripts.</td>
</tr>
<tr>
<td>selectFilter</td>
<td>query</td>
<td>Return a filter property with the action condition filter.</td>
</tr>
<tr>
<td>selectOperations</td>
<td>query</td>
<td>Return an operations property with action operations.</td>
</tr>
<tr>
<td>selectRecoveryOperations</td>
<td>query</td>
<td>Return a recovery_operations property with action recovery operations.</td>
</tr>
<tr>
<td>selectUpdateOperations</td>
<td>query</td>
<td>Return an update_operations property with action update operations.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Possible values are: actionid, name and status. These parameters being common for all get methods are described in the reference commentary.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Return values**

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

**Examples**

**Retrieve trigger actions**

Retrieve all configured trigger actions together with action conditions and operations.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "action.get",
    "params": {
```
Retrieve discovery actions

Retrieve all configured discovery actions together with action conditions and operations. The filter uses the "and" evaluation type, so the formula property is empty and eval_formula is generated automatically.

Request:
{
"jsonrpc": "2.0",
"method": "action.get",
"params": {
"output": "extend",
"selectOperations": "extend"
"selectFilter": "extend",
"filter": {
"eventsource": 1
}
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}
Response:

{
"jsonrpc": "2.0",
"result": [
{
"actionid": "2",
"name": "Auto discovery. Linux servers.",
"eventsource": "1",
"status": "1",
"esc_period": "0s",
"pause_suppressed": "1",
"filter": {
"evaltype": "0",
"formula": "",
"conditions": [
{
"conditiontype": "10",
"operator": "0",
"value": "0",
"value2": "",
"formulaid": "B"
},
{
"conditiontype": "8",
"operator": "0",
"value": "9",
"value2": "",
"formulaid": "C"
},
{
"conditiontype": "12",
"operator": "2",
"value": "Linux",
"value2": "",
"formulaid": "A"
}
],
"eval_formula": "A and B and C"
},
"operations": [
{
"operationid": "1",
"actionid": "2",
"operationtype": "6",
"esc_period": "0s",
"esc_step_from": "1",
"esc_step_to": "1",

868


See also
- Action filter
- Action operation

Source
CAction::get() in ui/include/classes/api/services/CAction.php.

action.update

Description
object action.update(object/array actions)

This method allows to update existing actions.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(object/array) Action properties to be updated.

The actionid property must be defined for each action, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard action properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>object</td>
<td>Action filter object to replace the current filter.</td>
</tr>
<tr>
<td>operations</td>
<td>array</td>
<td>Action operations to replace existing operations.</td>
</tr>
<tr>
<td>recovery_operations</td>
<td>array</td>
<td>Action recovery operations to replace existing recovery operations.</td>
</tr>
<tr>
<td>update_operations</td>
<td>array</td>
<td>Action update operations to replace existing update operations.</td>
</tr>
</tbody>
</table>

Return values
Returns an object containing the IDs of the updated actions under the actionids property.

Examples

Disable action

Disable action, that is, set its status to "1".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "action.update",
    "params": {
        "actionid": "2",
        "status": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": ["2"]
    },
    "id": 1
}
```

See also

- Action filter
- Action operation

Source

CAction::update() in ui/include/classes/api/services/CAction.php.

**Alert**

This class is designed to work with alerts.

Object references:

- Alert

Available methods:

- alert.get - retrieve alerts

> Alert object

The following objects are directly related to the alert API.

Alert

Alerts are created by the Zabbix server and cannot be modified via the API.

The alert object contains information about whether certain action operations have been executed successfully. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alertid</td>
<td>string</td>
<td>ID of the alert.</td>
</tr>
<tr>
<td>actionid</td>
<td>string</td>
<td>ID of the action that generated the alert.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>alerttype</td>
<td>integer</td>
<td>Alert type.</td>
</tr>
<tr>
<td>clock</td>
<td>timestamp</td>
<td>Time when the alert was generated.</td>
</tr>
<tr>
<td>error</td>
<td>string</td>
<td>Error text if there are problems sending a message or running a command.</td>
</tr>
<tr>
<td>esc_step</td>
<td>integer</td>
<td>Action escalation step during which the alert was generated.</td>
</tr>
<tr>
<td>eventid</td>
<td>string</td>
<td>ID of the event that triggered the action.</td>
</tr>
<tr>
<td>mediatypeid</td>
<td>string</td>
<td>ID of the media type that was used to send the message.</td>
</tr>
<tr>
<td>message</td>
<td>text</td>
<td>Message text. Used for message alerts.</td>
</tr>
<tr>
<td>retries</td>
<td>integer</td>
<td>Number of times Zabbix tried to send the message.</td>
</tr>
<tr>
<td>sendto</td>
<td>string</td>
<td>Address, user name or other identifier of the recipient. Used for message alerts.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Status indicating whether the action operation has been executed successfully.</td>
</tr>
</tbody>
</table>

Possible values for message alerts:
- 0 - message not sent.
- 1 - message sent.
- 2 - failed after a number of retries.
- 3 - new alert is not yet processed by alert manager.

Possible values for command alerts:
- 0 - command not run.
- 1 - command run.
- 2 - tried to run the command on the Zabbix agent but it was unavailable.

subject string Message subject. Used for message alerts.
userid string ID of the user that the message was sent to.
p_eventid string ID of problem event, which generated the alert.
acknowledgeid string ID of acknowledgment, which generated the alert.

**alert.get**

Description

integer/array alert.get(object parameters)

The method allows to retrieve alerts according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See **User roles** for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alertids</td>
<td>string/array</td>
<td>Return only alerts with the given IDs.</td>
</tr>
<tr>
<td>actionids</td>
<td>string/array</td>
<td>Return only alerts generated by the given actions.</td>
</tr>
<tr>
<td>eventids</td>
<td>string/array</td>
<td>Return only alerts generated by the given events.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only alerts generated by objects from the given host groups.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only alerts generated by objects from the given hosts.</td>
</tr>
<tr>
<td>mediatypeids</td>
<td>string/array</td>
<td>Return only message alerts that used the given media types.</td>
</tr>
<tr>
<td>objectids</td>
<td>string/array</td>
<td>Return only alerts generated by the given objects</td>
</tr>
<tr>
<td>userids</td>
<td>string/array</td>
<td>Return only message alerts that were sent to the given users.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>eventobject</td>
<td>integer</td>
<td>Return only alerts generated by events related to objects of the given type. See event &quot;object&quot; for a list of supported object types.</td>
</tr>
<tr>
<td>eventsource</td>
<td>integer</td>
<td>Return only alerts generated by events of the given type. See event &quot;source&quot; for a list of supported event types.</td>
</tr>
<tr>
<td>time_from</td>
<td>timestamp</td>
<td>Return only alerts that have been generated after the given time.</td>
</tr>
<tr>
<td>time_till</td>
<td>timestamp</td>
<td>Return only alerts that have been generated before the given time.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return a hosts property with data of hosts that triggered the action operation.</td>
</tr>
<tr>
<td>selectMediatypes</td>
<td>query</td>
<td>Return a mediatypes property with an array of the media types that were used for the message alert.</td>
</tr>
<tr>
<td>selectUsers</td>
<td>query</td>
<td>Return a users property with an array of the users that the message was addressed to.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties. Possible values are: alertid, clock, eventid, mediatypeid, sendto and status.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in the reference commentary.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values
(integer/array) Returns either:
- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples
Retrieve alerts by action ID
Retrieve all alerts generated by action "3".

Request:
```json
{
    "jsonrpc": "2.0",
    "method": "alert.get",
    "params": {
        "output": "extend",
        "actionids": "3"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:
See also

- Host
- Media type
- User

Source

CAAlert::get() in ui/include/classes/api/services/CAAlert.php.

**API info**

This class is designed to retrieve meta information about the API.

Available methods:

- **apiinfo.version** - retrieving the version of the Zabbix API

**apiinfo.version**

**Description**

string **apiinfo.version**(array)

This method allows to retrieve the version of the Zabbix API.

This method is only available to unauthenticated users and must be called without the auth parameter in the JSON-RPC request.

**Parameters**

(array) The method accepts an empty array.

**Return values**

(string) Returns the version of the Zabbix API.

Starting from Zabbix 2.0.4 the version of the API matches the version of Zabbix.

**Examples**

Retrieving the version of the API

Retrieve the version of the Zabbix API.
Request:
{
    "jsonrpc": "2.0",
    "method": "apiinfo.version",
    "params": [],
    "id": 1
}

Response:
{
    "jsonrpc": "2.0",
    "result": "4.0.0",
    "id": 1
}

Source
CAPIInfo::version() in ui/include/classes/api/services/CAPIInfo.php.

**Audit log**

This class is designed to work with audit log.

Object references:

- Audit log object

Available methods:

- `auditlog.get` - retrieve audit log records

> **Audit log object**

The following objects are directly related to the audit log API.

Audit log

The audit log object contains information about user actions. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auditid</td>
<td>string</td>
<td>(readonly) ID of audit log entry. Generated using CUID algorithm.</td>
</tr>
<tr>
<td>userid</td>
<td>string</td>
<td>Audit log entry author userid.</td>
</tr>
<tr>
<td>username</td>
<td>string</td>
<td>Audit log entry author username.</td>
</tr>
<tr>
<td>clock</td>
<td>timestamp</td>
<td>Audit log entry creation timestamp.</td>
</tr>
<tr>
<td>ip</td>
<td>string</td>
<td>Audit log entry author IP address.</td>
</tr>
<tr>
<td>action</td>
<td>integer</td>
<td>Audit log entry action.</td>
</tr>
</tbody>
</table>

Possible values are:
0 - Add;
1 - Update;
2 - Delete;
4 - Logout;
7 - Execute;
8 - Login;
9 - Failed login;
10 - History clear;
11 - Config refresh.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourcetype</td>
<td>integer</td>
<td>Audit log entry resource type. Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - User; 3 - Media type; 4 - Host; 5 - Action; 6 - Graph; 11 - User group;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 - Trigger; 14 - Host group; 15 - Item; 16 - Image; 17 - Value map; 18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Service; 19 - Map; 22 - Web scenario; 23 - Discovery rule; 25 - Script;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26 - Proxy; 27 - Maintenance; 28 - Regular expression; 29 - Macro; 30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Template; 31 - Trigger prototype; 32 - Icon mapping; 33 - Dashboard;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>34 - Event correlation; 35 - Graph prototype; 36 - Item prototype; 37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Host prototype; 38 - Autoregistration; 39 - Module; 40 - Settings;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>41 - Housekeeping; 42 - Authentication; 43 - Template dashboard; 44 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User role; 45 - Auth token; 46 - Scheduled report; 47 - High availability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>node; 48 - SLA; 49 - LDAP user directory. 50 - Template group.</td>
</tr>
<tr>
<td>resourceid</td>
<td>string</td>
<td>Audit log entry resource identifier.</td>
</tr>
<tr>
<td>resourcename</td>
<td>string</td>
<td>Audit log entry resource human readable name.</td>
</tr>
<tr>
<td>recordsetid</td>
<td>string</td>
<td>Audit log entry recordset ID. The audit log records created during the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>same operation will have the same recordset ID. Generated using CUID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>algorithm.</td>
</tr>
<tr>
<td>details</td>
<td>text</td>
<td>Audit log entry details. The details are stored as JSON object where each</td>
</tr>
<tr>
<td></td>
<td></td>
<td>property name is a path to property or nested object in which change</td>
</tr>
<tr>
<td></td>
<td></td>
<td>occurred, and each value contain the data about the change of this</td>
</tr>
<tr>
<td></td>
<td></td>
<td>property in array format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible value formats are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[&quot;add&quot;] - Nested object has been added;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[&quot;add&quot;, &quot;&lt;value&gt;&quot;] - The property of added object contain &lt;value&gt;;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[&quot;update&quot;] - Nested object has been updated;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[&quot;update&quot;, &quot;&lt;new value&gt;&quot;, &quot;&lt;old value&gt;&quot; ] - The value of property of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>updated object was changed from &lt;old value&gt; to &lt;new value&gt;;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[&quot;delete&quot;] - Nested object has been deleted.</td>
</tr>
</tbody>
</table>
**auditlog.get**

Description

integer/array auditlog.get(object parameters)

The method allows to retrieve audit log records according to the given parameters.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auditids</td>
<td>string/array</td>
<td>Return only audit log with the given IDs.</td>
</tr>
<tr>
<td>userid</td>
<td>string/array</td>
<td>Return only audit log that were created by the given users.</td>
</tr>
<tr>
<td>time_from</td>
<td>timestamp</td>
<td>Returns only audit log entries that have been created after or at the given time.</td>
</tr>
<tr>
<td>time_till</td>
<td>timestamp</td>
<td>Returns only audit log entries that have been created before or at the given time.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td>Return only results that exactly match the given filter.</td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td>Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in the <a href="#">reference commentary</a>.</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieve audit log

Retrieve two latest audit log records.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "auditlog.get",
    "params": {
        "output": "extend",
        "sortfield": "clock",
        "countOutput": true
    }
}
```
"sortorder": "DESC",
"limit": 2
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": [
    {
      "auditid": "cksstgfa0001yhdcc41y20q2",
      "userid": "1",
      "username": "Admin",
      "clock": "1629975715",
      "ip": "127.0.0.1",
      "action": "1",
      "resourcetype": "0",
      "resourceid": "0",
      "resourcename": "Jim",
      "recordsetid": "cksstgfal0000yhdcso67ondl",
      "details": "{"user.name": ["update\", "Jim\", ","], "user.medias[37]": ["add"], "user.medias[37].mediatypeid": ["add"], "user.medias[37].sendto": ["support123@company.com"]}
    },
    {
      "auditid": "ckssofl0p0001yhdcqxclsg8r",
      "userid": "1",
      "username": "Admin",
      "clock": "1629967278",
      "ip": "127.0.0.1",
      "action": "0",
      "resourcetype": "0",
      "resourceid": "20",
      "resourcename": "John",
      "recordsetid": "ckssofl0p0000yhdcpxyoljgo",
      "details": "{"user.username": ["add", "John"], "user.userid": ["add", "20"], "user.usrgrps[28]": ["add", 7]}
    }
  ],
  "id": 1
}

See also
• Audit log object

Source
CAuditLog::get() in ui/include/classes/api/services/CAuditLog.php.

Authentication

This class is designed to work with authentication settings.

Object references:

• Authentication

Available methods:

• authentication.get - retrieve authentication
• authentication.update - update authentication
> **Authentication object**

The following objects are directly related to the authentication API.

Authentication

The authentication object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>authentication_type</td>
<td>integer</td>
<td>Default authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) Internal; 1 - LDAP.</td>
</tr>
<tr>
<td>http_auth_enabled</td>
<td>integer</td>
<td>Enable HTTP authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) Disable; 1 - Enable.</td>
</tr>
<tr>
<td>http_login_form</td>
<td>integer</td>
<td>Default login form.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) Zabbix login form; 1 - HTTP login form.</td>
</tr>
<tr>
<td>http_strip_domains</td>
<td>string</td>
<td>Remove domain name.</td>
</tr>
<tr>
<td>http_case_sensitive</td>
<td>integer</td>
<td>HTTP case sensitive login.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - Off; 1 - (default) On.</td>
</tr>
<tr>
<td>ldap_configured</td>
<td>integer</td>
<td>Enable LDAP authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) Disable; 1 - Enable.</td>
</tr>
<tr>
<td>ldap_case_sensitive</td>
<td>integer</td>
<td>LDAP case sensitive login.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - Off; 1 - (default) On.</td>
</tr>
<tr>
<td>ldap_userdirectoryid</td>
<td>string</td>
<td>LDAP authentication default user directory for user groups with gui_access set to LDAP or System default.</td>
</tr>
<tr>
<td>saml_auth_enabled</td>
<td>integer</td>
<td>Enable SAML authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) Disable; 1 - Enable.</td>
</tr>
<tr>
<td>saml_idp_entityid</td>
<td>string</td>
<td>SAML IdP entity ID.</td>
</tr>
<tr>
<td>saml_sso_url</td>
<td>string</td>
<td>SAML SSO service URL.</td>
</tr>
<tr>
<td>saml_slo_url</td>
<td>string</td>
<td>SAML SLO service URL.</td>
</tr>
<tr>
<td>saml_username_attribute</td>
<td>string</td>
<td>SAML username attribute.</td>
</tr>
<tr>
<td>saml_sp_entityid</td>
<td>string</td>
<td>SAML SP entity ID.</td>
</tr>
<tr>
<td>saml_nameid_format</td>
<td>string</td>
<td>SAML SP name ID format.</td>
</tr>
<tr>
<td>saml_sign_messages</td>
<td>integer</td>
<td>SAML sign messages.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) Do not sign messages; 1 - Sign messages.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>saml_sign_assertions</td>
<td>integer</td>
<td>SAML sign assertions. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Do not sign assertions; 1 - Sign assertions.</td>
</tr>
<tr>
<td>saml_sign_authn_requests</td>
<td>integer</td>
<td>SAML sign AuthN requests.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) Do not sign AuthN requests; 1 - Sign AuthN requests.</td>
</tr>
<tr>
<td>saml_sign_logout_requests</td>
<td>integer</td>
<td>SAML sign logout requests.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) Do not sign logout requests; 1 - Sign logout requests.</td>
</tr>
<tr>
<td>saml_sign_logout_responses</td>
<td>integer</td>
<td>SAML sign logout responses.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) Do not sign logout responses; 1 - Sign logout responses.</td>
</tr>
<tr>
<td>saml_encrypt_nameid</td>
<td>integer</td>
<td>SAML encrypt name ID. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Do not encrypt name ID; 1 - Encrypt name ID.</td>
</tr>
<tr>
<td>saml_encrypt_assertions</td>
<td>integer</td>
<td>SAML encrypt assertions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) Do not encrypt assertions; 1 - Encrypt assertions.</td>
</tr>
<tr>
<td>saml_case_sensitive</td>
<td>integer</td>
<td>SAML case sensitive login.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - Off; 1 - (default) On.</td>
</tr>
<tr>
<td>passwd_min_length</td>
<td>integer</td>
<td>Password minimal length requirement. Possible range of values: 1-70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 - default</td>
</tr>
<tr>
<td>passwd_check_rules</td>
<td>integer</td>
<td>Password checking rules.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible bitmap values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - check password length;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - check if password uses uppercase and lowercase Latin letters;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - check if password uses digits;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - check if password uses special characters;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 - (default) check if password is not in the list of commonly used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>passwords, does not contain derivations of word “Zabbix” or user’s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>name, last name or username.</td>
</tr>
</tbody>
</table>

**authentication.get**

**Description**

`object authentication.get(object parameters)`

The method allows to retrieve authentication object according to the given parameters.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**
Parameters defining the desired output.

The method supports only one parameter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>query</td>
<td>This parameter being common for all get methods described in the reference commentary.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns authentication object.

Examples

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "authentication.get",
    "params": {
        "output": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "authentication_type": "0",
        "http_auth_enabled": "0",
        "http_login_form": "0",
        "http_strip_domains": "",
        "http_case_sensitive": "1",
        "ldap_configured": "0",
        "ldap_case_sensitive": "1",
        "ldap_userdirectoryid": "0",
        "saml_auth_enabled": "0",
        "saml_idp_entityid": "",
        "saml_sso_url": "",
        "saml_slo_url": "",
        "saml_username_attribute": "",
        "saml_sp_entityid": "",
        "saml_nameid_format": "",
        "saml_sign_messages": "0",
        "saml_sign_assertions": "0",
        "saml_sign_authn_requests": "0",
        "saml_sign_logout_requests": "0",
        "saml_sign_logout_responses": "0",
        "saml_encrypt_nameid": "0",
        "saml_encrypt_assertions": "0",
        "saml_case_sensitive": "0",
        "passwd_min_length": "8",
        "passwd_check_rules": "8"
    },
    "id": 1
}
```

Source

CAuthentication::get() in ui/include/classes/api/services/CAuthentication.php.

[authentication.update](#)
Description

`object authentication.update(object authentication)`

This method allows to update existing authentication settings.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Authentication properties to be updated.

Return values

(array) Returns array with the names of updated parameters.

Examples

Request:

```
{  
    "jsonrpc": "2.0",
    "method": "authentication.update",
    "params": {  
        "http_auth_enabled": 1,
        "http_case_sensitive": 0,
        "http_login_form": 1
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{  
    "jsonrpc": "2.0",
    "result": [  
        "http_auth_enabled",
        "http_case_sensitive",
        "http_login_form"
    ],
    "id": 1
}
```

Source

CAuthentication::update() in ui/include/classes/api/services/CAuthentication.php.

**Autoregistration**

This class is designed to work with autoregistration.

Object references:

- Autoregistration

Available methods:

- autoregistration.get - retrieve autoregistration
- autoregistration.update - update autoregistration

> **Autoregistration object**

The following objects are directly related to the autoregistration API.

Autoregistration

The autoregistration object has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tls_accept</td>
<td>integer</td>
<td>Type of allowed incoming connections for autoregistration.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - allow insecure connections;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - allow TLS with PSK.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - allow both insecure and TLS with PSK connections.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(write-only) PSK identity string.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Do not put sensitive information in the PSK identity, it is transmitted</td>
</tr>
<tr>
<td></td>
<td></td>
<td>unencrypted over the network to inform a receiver which PSK to use.</td>
</tr>
<tr>
<td>tls_psk_identity</td>
<td>string</td>
<td>(write-only) PSK value string (an even number of hexadecimal characters).</td>
</tr>
<tr>
<td>tls_psk</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

**autoregistration.get**

Description

**object autoregistration.get(object parameters)**

The method allows to retrieve autoregistration object according to the given parameters.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports only one parameter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>query</td>
<td>This parameter being common for all get methods described in the reference commentary.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns autoregistration object.

Examples

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "autoregistration.get",
    "params": {
        "output": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "tls_accept": "3"
    },
    "id": 1
}
```

Source

CAutoregistration::get() in ui/include/classes/api/services/CAutoregistration.php.
**autoregistration.update**

Description

`object autoregistration.update(object autoregistration)`

This method allows to update existing autoregistration.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

Parameters

*(object)* Autoregistration properties to be updated.

Return values

*(boolean)* Returns boolean true as result on successful update.

Examples

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "autoregistration.update",
   "params": {
      "tls_accept": "3",
      "tls_psk_identity": "PSK 001",
      "tls_psk": "11111595725ac58dd977beef14b97461a7c1045b9a1c923453302c5473193478"
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": true,
   "id": 1
}
```

Source

`CAutoregistration::update()` in `ui/include/classes/api/services/CAutoregistration.php`

**Configuration**

This class is designed to export and import Zabbix configuration data.

Available methods:

- **configuration.export** - exporting the configuration
- **configuration.import** - importing the configuration

**configuration.export**

Description

`string configuration.export(object parameters)`

This method allows to export configuration data as a serialized string.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

Parameters

*(object)* Parameters defining the objects to be exported and the format to use.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>format</strong></td>
<td>string</td>
<td>String in which the data must be exported.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>Possible values: yml - YAML; xml - XML; json - JSON; raw - unprocessed PHP array.</td>
</tr>
<tr>
<td><strong>prettyprint</strong></td>
<td>boolean</td>
<td>Make the output more human readable by adding indentation.</td>
</tr>
<tr>
<td><strong>options</strong></td>
<td>object</td>
<td>Objects to be exported.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>The options object has the following parameters: host_groups - (array) IDs of host groups to export; hosts - (array) IDs of hosts to export; images - (array) IDs of images to export; maps - (array) IDs of maps to export; mediaTypes - (array) IDs of media types to export; template_groups - (array) IDs of template groups to export; templates - (array) IDs of templates to export.</td>
</tr>
</tbody>
</table>

**Return values**

(string) Returns a serialized string containing the requested configuration data.

**Examples**

Export the configuration of a host as an XML string.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "configuration.export",
    "params": {
        "options": {
            "hosts": [
                "10161"
            ],
            "format": "xml"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": "<?xml version="1.0" encoding="UTF-8"?>\n<zabbix_export><version>5.4</version><date>2020-03-13T15:31:45Z</date><groups><group><uuid>6f6799aa69e844b4b3918f779f2abf0 ... value>2</value><newvalue>Unreachable</newvalue></mapping></mappings></valuemap></valuemaps></host></hosts></zabbix_export>
```

**Source**

CConfiguration::export() in ui/include/classes/api/services/CConfiguration.php.
boolean configuration.import(object parameters)

This method allows to import configuration data from a serialized string.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters containing the data to import and rules how the data should be handled.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>format</strong></td>
<td>string</td>
<td>Format of the serialized string. Possible values: yaml - YAML; xml - XML; json - JSON.</td>
</tr>
<tr>
<td><strong>source</strong></td>
<td>string</td>
<td>Serialized string containing the configuration data.</td>
</tr>
<tr>
<td><strong>rules</strong></td>
<td>object</td>
<td>Rules on how new and existing objects should be imported. The rules parameter is described in detail in the table below.</td>
</tr>
</tbody>
</table>

If no rules are given, the configuration will not be updated.

The rules object supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>discoveryRules</td>
<td>object</td>
<td>Rules on how to import LLD rules. Supported parameters: createMissing - (boolean) if set to true, new LLD rules will be created; default: false; updateExisting - (boolean) if set to true, existing LLD rules will be updated; default: false; deleteMissing - (boolean) if set to true, LLD rules not present in the imported data will be deleted from the database; default: false.</td>
</tr>
<tr>
<td>graphs</td>
<td>object</td>
<td>Rules on how to import graphs. Supported parameters: createMissing - (boolean) if set to true, new graphs will be created; default: false; updateExisting - (boolean) if set to true, existing graphs will be updated; default: false; deleteMissing - (boolean) if set to true, graphs not present in the imported data will be deleted from the database; default: false.</td>
</tr>
<tr>
<td>groups</td>
<td>object</td>
<td>Rules on how to import host groups. Supported parameters: createMissing - (boolean) if set to true, new host groups will be created; default: false; updateExisting - (boolean) if set to true, existing host groups will be updated; default: false.</td>
</tr>
<tr>
<td>hosts</td>
<td>object</td>
<td>Rules on how to import hosts. Supported parameters: createMissing - (boolean) if set to true, new hosts will be created; default: false; updateExisting - (boolean) if set to true, existing hosts will be updated; default: false.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>httptests</td>
<td>object</td>
<td>Rules on how to import web scenarios.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>createMissing - (boolean) if set to true, new web scenarios will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>updateExisting - (boolean) if set to true, existing web scenarios will be updated; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>deleteMissing - (boolean) if set to true, web scenarios not present in the imported data will be deleted from the database; default: false.</td>
</tr>
<tr>
<td>images</td>
<td>object</td>
<td>Rules on how to import images.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>createMissing - (boolean) if set to true, new images will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>updateExisting - (boolean) if set to true, existing images will be updated; default: false.</td>
</tr>
<tr>
<td>items</td>
<td>object</td>
<td>Rules on how to import items.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>createMissing - (boolean) if set to true, new items will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>updateExisting - (boolean) if set to true, existing items will be updated; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>deleteMissing - (boolean) if set to true, items not present in the imported data will be deleted from the database; default: false.</td>
</tr>
<tr>
<td>maps</td>
<td>object</td>
<td>Rules on how to import maps.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>createMissing - (boolean) if set to true, new maps will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>updateExisting - (boolean) if set to true, existing maps will be updated; default: false.</td>
</tr>
<tr>
<td>mediaTypes</td>
<td>object</td>
<td>Rules on how to import media types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>createMissing - (boolean) if set to true, new media types will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>updateExisting - (boolean) if set to true, existing media types will be updated; default: false.</td>
</tr>
<tr>
<td>templateLinkage</td>
<td>object</td>
<td>Rules on how to import template links.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>createMissing - (boolean) if set to true, new links between templates and host will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>deleteMissing - (boolean) if set to true, template links not present in the imported data will be deleted from the database; default: false.</td>
</tr>
<tr>
<td>templates</td>
<td>object</td>
<td>Rules on how to import templates.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>createMissing - (boolean) if set to true, new templates will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>updateExisting - (boolean) if set to true, existing templates will be updated; default: false.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>templateDashboards</td>
<td>object</td>
<td>Rules on how to import template dashboards.</td>
</tr>
<tr>
<td>triggers</td>
<td>object</td>
<td>Rules on how to import triggers.</td>
</tr>
<tr>
<td>valueMaps</td>
<td>object</td>
<td>Rules on how to import host or template value maps.</td>
</tr>
</tbody>
</table>

**return values**

(booleanc) Returns true if importing has been successful.

**Examples**

**Importing hosts and items**

Import the host and items contained in the XML string. If any items in XML are missing, they will be deleted from the database, and everything else will be left unchanged.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "configuration.import",
    "params": {
        "format": "xml",
        "rules": {
            "valueMaps": {
                "createMissing": true,
                "updateExisting": false
            },
            "hosts": {
                "createMissing": true,
                "updateExisting": true
            },
            "items": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            }
        },
        "source": "<?xml version="1.0" encoding="UTF-8"?>
        \n        <zabbix_export><version>5.4</version><date>
```
Source
CConfiguration::import() in ui/include/classes/api/services/CConfiguration.php.

configuration.importcompare

Description
array configuration.importcompare(object parameters)

This method allows to compare import file with current system elements and shows what will be changed if this import file will be imported.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(object) Parameters containing the possible data to import and rules how the data should be handled.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>format</td>
<td>string</td>
<td>Format of the serialized string.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>yaml - YAML;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>xml - XML;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>json - JSON.</td>
</tr>
<tr>
<td>source</td>
<td>string</td>
<td>Serialized string containing the configuration data.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rules</td>
<td>object</td>
<td>Rules on how new and existing objects should be imported.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>The rules parameter is described in detail in the table below.</td>
</tr>
</tbody>
</table>

If no rules are given, there will be nothing to update and result will be empty.

Comparison will be done only for host groups and templates. Triggers and graphs will be compared only for imported templates, any other will be considered as “new”.

The rules object supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>discoveryRules</td>
<td>object</td>
<td>Rules on how to import LLD rules.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>createMissing - (boolean) if set to true, new LLD rules will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>updateExisting - (boolean) if set to true, existing LLD rules will be updated; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>deleteMissing - (boolean) if set to true, LLD rules not present in the imported data will be deleted from the database; default: false.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>graphs</td>
<td>object</td>
<td>Rules on how to import graphs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- createMissing - (boolean) if set to true, new graphs will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- updateExisting - (boolean) if set to true, existing graphs will be updated; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- deleteMissing - (boolean) if set to true, graphs not present in the imported data will be deleted from the database; default: false.</td>
</tr>
<tr>
<td>groups</td>
<td>object</td>
<td>Rules on how to import host groups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- createMissing - (boolean) if set to true, new host groups will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- updateExisting - (boolean) if set to true, existing host groups will be updated; default: false.</td>
</tr>
<tr>
<td>hosts</td>
<td>object</td>
<td>Rules on how to import hosts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- createMissing - (boolean) if set to true, new hosts will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- updateExisting - (boolean) if set to true, existing hosts will be updated; default: false.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This parameter will make no difference to the output. It is allowed only for consistency with configuration.import.</td>
</tr>
<tr>
<td>httptests</td>
<td>object</td>
<td>Rules on how to import web scenarios.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- createMissing - (boolean) if set to true, new web scenarios will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- updateExisting - (boolean) if set to true, existing web scenarios will be updated; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- deleteMissing - (boolean) if set to true, web scenarios not present in the imported data will be deleted from the database; default: false.</td>
</tr>
<tr>
<td>images</td>
<td>object</td>
<td>Rules on how to import images.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- createMissing - (boolean) if set to true, new images will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- updateExisting - (boolean) if set to true, existing images will be updated; default: false.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This parameter will make no difference to the output. It is allowed only for consistency with configuration.import.</td>
</tr>
<tr>
<td>items</td>
<td>object</td>
<td>Rules on how to import items.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported parameters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- createMissing - (boolean) if set to true, new items will be created; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- updateExisting - (boolean) if set to true, existing items will be updated; default: false;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- deleteMissing - (boolean) if set to true, items not present in the imported data will be deleted from the database; default: false.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>maps</td>
<td>object</td>
<td>Rules on how to import maps.</td>
</tr>
<tr>
<td>mediaTypes</td>
<td>object</td>
<td>Rules on how to import media types.</td>
</tr>
<tr>
<td>templateLinkage</td>
<td>object</td>
<td>Rules on how to import template links.</td>
</tr>
<tr>
<td>templates</td>
<td>object</td>
<td>Rules on how to import templates.</td>
</tr>
<tr>
<td>templateDashboards</td>
<td>object</td>
<td>Rules on how to import template dashboards.</td>
</tr>
<tr>
<td>triggers</td>
<td>object</td>
<td>Rules on how to import triggers.</td>
</tr>
</tbody>
</table>

**Supported parameters for maps:**
- `createMissing` - (boolean) if set to true, new maps will be created; default: false;
- `updateExisting` - (boolean) if set to true, existing maps will be updated; default: false.

This parameter will make no difference to the output. It is allowed only for consistency with configuration import.

**Supported parameters for mediaTypes:**
- `createMissing` - (boolean) if set to true, new media types will be created; default: false;
- `updateExisting` - (boolean) if set to true, existing media types will be updated; default: false.

This parameter will make no difference to the output. It is allowed only for consistency with configuration import.

**Supported parameters for templateLinkage:**
- `createMissing` - (boolean) if set to true, new links between templates and host will be created; default: false;
- `deleteMissing` - (boolean) if set to true, template links not present in the imported data will be deleted from the database; default: false.

**Supported parameters for templates:**
- `createMissing` - (boolean) if set to true, new templates will be created; default: false;
- `updateExisting` - (boolean) if set to true, existing templates will be updated; default: false.

**Supported parameters for templateDashboards:**
- `createMissing` - (boolean) if set to true, new template dashboards will be created; default: false;
- `updateExisting` - (boolean) if set to true, existing template dashboards will be updated; default: false;
- `deleteMissing` - (boolean) if set to true, template dashboards not present in the imported data will be deleted from the database; default: false.

**Supported parameters for triggers:**
- `createMissing` - (boolean) if set to true, new triggers will be created; default: false;
- `updateExisting` - (boolean) if set to true, existing triggers will be updated; default: false;
- `deleteMissing` - (boolean) if set to true, triggers not present in the imported data will be deleted from the database; default: false.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>valueMaps</td>
<td>object</td>
<td>Rules on how to import host or template value maps.</td>
</tr>
</tbody>
</table>

Supported parameters:
- `createMissing` - (boolean) if set to true, new value maps will be created; default: false;
- `updateExisting` - (boolean) if set to true, existing value maps will be updated; default: false;
- `deleteMissing` - (boolean) if set to true, value maps not present in the imported data will be deleted from the database; default: false.

Return values

(array) Returns an array with changes in configuration, that will be made.

Examples

Importing hosts and items

Import the template and items contained in the YAML string. If any items in YAML are missing, they will be shown as deleted, and everything else will be left unchanged.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "configuration.import",
    "params": {
        "format": "xml",
        "rules": {
            "groups": {
                "createMissing": true,
                "updateExisting": true
            },
            "templates": {
                "createMissing": true,
                "updateExisting": true
            },
            "items": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "triggers": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "discoveryRules": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "valueMaps": {
                "createMissing": true,
                "updateExisting": false
            }
        },
        "source": "<?xml version="1.0" encoding="UTF-8"?><zabbix_export><version>5.4</version><date>2021-05-27T07:12:07Z</date><groups><group><uuid>6f6799aa69e844b4b3918f779f2ab ... value><newvalue>Unreachable</newvalue></mapping></mappings></valuemap></valuemaps></template></templates></zabbix_export>",
    "auth": "038e1d7b1735c6a5436ee9eae096879e",
    "id": 1
}
```

Response:

891
{  
"jsonrpc":"2.0",  
"result":{  
  "templates":{  
    "updated":{  
      "before":{  
        "uuid":"e1bde9bf2f0544f5929f45b82502e744",  
        "template":"Export template",  
        "name":"Export template"  
      },  
      "after":{  
        "uuid":"e1bde9bf2f0544f5929f45b82502e744",  
        "template":"Export template",  
        "name":"Export template"  
      },  
      "items":{  
        "added":{  
          "after":{  
            "uuid":"3237bc89226e42ed8207574022470e83",  
            "name":"Item",  
            "key":"item.key",  
            "delay":"30s",  
            "valuemap":{  
              "name":"Host status"  
            }  
          },  
          "triggers":{  
            "added":{  
              "after":{  
                "uuid":"bd1ed0089e4b4f35b762c9d6c599c348",  
                "expression":"last(/Export template/item.key)=0",  
                "name":"Trigger"  
              }  
            }  
          }  
        },  
        "removed":{  
          "before":{  
            "uuid":"bd3e7b28b3d544d6a83ed01ddaa65ab6",  
            "name":"Old Item",  
            "key":"ite_old.key",  
            "delay":"30s",  
            "valuemap":{  
              "name":"Host status"  
            }  
          }  
        }  
      },  
      "discovery_rules":{  
        "updated":{  
          "before":{  
            "uuid":"c91616bcf4a44f349539a1b40cb0979d",  
            "name":"Discovery rule",  
            "key":"rule.key"  
          }  
        }  
      }  
    }  
  }  
}
Source
CConfiguration::importcompare() in ui/include/classes/api/services/CConfiguration.php.

Correlation

This class is designed to work with correlations.

Object references:

- Correlation

Available methods:

- correlation.create - creating new correlations
- correlation.delete - deleting correlations
- correlation.get - retrieving correlations
- correlation.update - updating correlations

> Correlation object

The following objects are directly related to the correlation API.

Correlation

The correlation object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlationid</td>
<td>string</td>
<td>(readonly) ID of the correlation.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the correlation.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Description of the correlation.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Whether the correlation is enabled or disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) enabled;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - disabled.</td>
</tr>
</tbody>
</table>

Correlation operation

The correlation operation object defines an operation that will be performed when a correlation is executed. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>type</strong></td>
<td>integer</td>
<td>Type of operation.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - close old events;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - close new event.</td>
</tr>
</tbody>
</table>

Correlation filter

The correlation filter object defines a set of conditions that must be met to perform the configured correlation operations. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>evaltype</strong></td>
<td>integer</td>
<td>Filter condition evaluation method.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - and/or;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - and;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - or;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - custom expression.</td>
</tr>
<tr>
<td><strong>conditions</strong></td>
<td>array</td>
<td>Set of filter conditions to use for filtering results.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>(readonly) Generated expression that will be used for evaluating filter conditions. The expression contains IDs that reference specific filter conditions by its formulaid. The value of eval_formula is equal to the value of formula for filters with a custom expression.</td>
</tr>
<tr>
<td><strong>eval_formula</strong></td>
<td>string</td>
<td>User-defined expression to be used for evaluating conditions of filters with a custom expression. The expression must contain IDs that reference specific filter conditions by its formulaid. The IDs used in the expression must exactly match the ones defined in the filter conditions: no condition can remain unused or omitted. Required for custom expression filters.</td>
</tr>
</tbody>
</table>

Correlation filter condition

The correlation filter condition object defines a specific condition that must be checked before running the correlation operations.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>integer</td>
<td>Type of condition.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - old event tag;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - new event tag;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - new event host group;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - event tag pair;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - old event tag value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - new event tag value.</td>
</tr>
<tr>
<td>tag</td>
<td>string</td>
<td>Event tag (old or new). Required when type of condition is: 0, 1, 4, 5.</td>
</tr>
<tr>
<td>groupid</td>
<td>string</td>
<td>Host group ID. Required when type of condition is: 2.</td>
</tr>
<tr>
<td>oldtag</td>
<td>string</td>
<td>Old event tag. Required when type of condition is: 3.</td>
</tr>
<tr>
<td>newtag</td>
<td>string</td>
<td>Old event tag. Required when type of condition is: 3.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Event tag (old or new) value. Required when type of condition is: 4, 5.</td>
</tr>
<tr>
<td>formulaid</td>
<td>string</td>
<td>Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.</td>
</tr>
<tr>
<td>operator</td>
<td>integer</td>
<td>Condition operator. Required when type of condition is: 2, 4, 5.</td>
</tr>
</tbody>
</table>

To better understand how to use filters with various types of expressions, see examples on the correlation.get and correlation.create method pages.

The following operators and values are supported for each condition type.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Condition name</th>
<th>Supported operators</th>
<th>Expected value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Host group</td>
<td>=, &lt;&gt;</td>
<td>Host group ID.</td>
</tr>
<tr>
<td>4</td>
<td>Old event tag value</td>
<td>=, &lt;&gt;, like, not like</td>
<td>string</td>
</tr>
<tr>
<td>5</td>
<td>New event tag value</td>
<td>=, &lt;&gt;, like, not like</td>
<td>string</td>
</tr>
</tbody>
</table>

**correlation.create**

**Description**

**object** correlation.create(object/array correlations)**

This method allows to create new correlations.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See **User roles** for more information.

**Parameters**

(object/array) Correlations to create.

Additionally to the **standard correlation properties**, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operations</td>
<td>array</td>
<td>Correlation operations to create for the correlation.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td>Correlation filter object for the correlation.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns an object containing the IDs of the created correlations under the correlationids property. The order of the returned IDs matches the order of the passed correlations.

**Examples**
Create a new event tag correlation

Create a correlation using evaluation method AND/OR with one condition and one operation. By default the correlation will be enabled.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "correlation.create",
    "params": {
        "name": "new event tag correlation",
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "type": 1,
                    "tag": "ok"
                }
            ]
        },
        "operations": [
            {
                "type": 0
            }
        ],
        "auth": "343baad4f98b4106b9b5961e77437688",
        "id": 1
    }
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1"
        ],
        "id": 1
    }
}
```

Using a custom expression filter

Create a correlation that will use a custom filter condition. The formula IDs "A" or "B" have been chosen arbitrarily. Condition type will be "Host group" with operator "<>".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "correlation.create",
    "params": {
        "name": "new host group correlation",
        "description": "a custom description",
        "status": 0,
        "filter": {
            "evaltype": 3,
            "formula": "A or B",
            "conditions": [
                {
                    "type": 2,
                    "operator": 1,
                    "formulaid": "A"
                },
                {
```
See also

- Correlation filter
- Correlation operation

Source

CCorrelation::create() in ui/include/classes/api/services/CCorrelation.php.

**correlation.delete**

**Description**

object correlation.delete(array correlationids)

This method allows to delete correlations.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(array) IDs of the correlations to delete.

**Return values**

(object) Returns an object containing the IDs of the deleted correlations under the correlationids property.

**Example**

Delete multiple correlations

Delete two correlations.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "correlation.delete",
    "params": [
        "1",
        "2"
    ],
    "auth": "343baad4f88b4106b9b5961e77437688",
    "id": 1
}
```
Response:
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1",
            "2"
        ]
    },
    "id": 1
}

Source
CCorrelation::delete() in ui/include/classes/api/services/CCorrelation.php.

correlation.get

Description
integer/array correlation.get(object parameters)

The method allows to retrieve correlations according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlationids</td>
<td>string/array</td>
<td>Return only correlations with the given IDs.</td>
</tr>
<tr>
<td>selectFilter</td>
<td>query</td>
<td>Return a filter property with the correlation conditions.</td>
</tr>
<tr>
<td>selectOperations</td>
<td>query</td>
<td>Return an operations property with the correlation operations.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in the</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td>reference commentary.</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values
(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.
Examples

Retrieve correlations

Retrieve all configured correlations together with correlation conditions and operations. The filter uses the "and/or" evaluation type, so the formula property is empty and eval_formula is generated automatically.

Request:
```
{
    "jsonrpc": "2.0",
    "method": "correlation.get",
    "params": {
        "output": "extend",
        "selectOperations": "extend",
        "selectFilter": "extend"
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "id": 1
}
```

Response:
```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "correlationid": "1",
            "name": "Correlation 1",
            "description": "",
            "status": "0",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [
                    {
                        "type": "3",
                        "oldtag": "error",
                        "newtag": "ok",
                        "formulaid": "A"
                    }
                ],
                "eval_formula": "A"
            },
            "operations": [
                {
                    "type": "0"
                }
            ],
            "id": 1
        }
    ]
}
```

See also
- Correlation filter
- Correlation operation

Source

CCorrelation::get() in ui/include/classes/api/services/CCorrelation.php.

**correlation.update**

Description

object correlation.update(object/array correlations)
This method allows to update existing correlations.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Correlation properties to be updated.

The correlationid property must be defined for each correlation, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard correlation properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>object</td>
<td>Correlation filter object to replace the current filter.</td>
</tr>
<tr>
<td>operations</td>
<td>array</td>
<td>Correlation operations to replace existing operations.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated correlations under the correlationids property.

Examples

Disable correlation

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "correlation.update",
   "params": {
      "correlationid": "1",
      "status": "1"
   },
   "auth": "343baad4f88b4106b9b5961e77437688",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "correlationids": ["1"],
   },
   "id": 1
}
```

Replace conditions, but keep the evaluation method

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "correlation.update",
   "params": {
      "correlationid": "1",
      "filter": {
         "conditions": [
            {
               "type": 3,
               "oldtag": "error",
               "newtag": "ok"
            }
         ]
      }
   }
}
```
Response:
{
  "jsonrpc": "2.0",
  "result": {
    "correlationIds": [
      "1"
    ],
    "id": 1
  },
  "id": 1
}

See also
• Correlation filter
• Correlation operation

Source

CCorrelation::update() in ui/include/classes/api/services/CCorrelation.php.

Dashboard

This class is designed to work with dashboards.

Object references:
• Dashboard
• Dashboard page
• Dashboard widget
• Dashboard widget field
• Dashboard user
• Dashboard user group

Available methods:
• dashboard.create - creating new dashboards
• dashboard.delete - deleting dashboards
• dashboard.get - retrieving dashboards
• dashboard.update - updating dashboards

> Dashboard object

The following objects are directly related to the dashboard API.

Dashboard

The dashboard object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dashboardid</td>
<td>string</td>
<td>(readonly) ID of the dashboard.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the dashboard.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>userid</td>
<td>string</td>
<td>Dashboard owner user ID.</td>
</tr>
<tr>
<td>private</td>
<td>integer</td>
<td>Type of dashboard sharing.</td>
</tr>
</tbody>
</table>

Possible values:
0 - public dashboard;
1 - (default) private dashboard.
Dashboard page

The dashboard page object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dashboard_pageid</td>
<td>string</td>
<td>(readonly) ID of the dashboard page.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Dashboard page name.</td>
</tr>
<tr>
<td>display_period</td>
<td>integer</td>
<td>Dashboard page display period (in seconds). Possible values: 0, 10, 30, 60, 120, 600, 1800, 3600. Default: 0 (will use the default page display period).</td>
</tr>
<tr>
<td>widgets</td>
<td>array</td>
<td>Array of the dashboard widget objects.</td>
</tr>
</tbody>
</table>

Dashboard widget

The dashboard widget object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>widgetid</td>
<td>string</td>
<td>(readonly) ID of the dashboard widget.</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>Type of the dashboard widget. Possible values: actionlog - Action log; clock - Clock; dataover - Data overview; discovery - Discovery status; favgraphs - Favorite graphs; favmaps - Favorite maps; graph - Graph (classic); graphprototype - Graph prototype; hostavail - Host availability; item - Item value; map - Map; navtree - Map Navigation Tree; plaintext - Plain text; problemhosts - Problem hosts; problems - Problems; problemsbysv - Problems by severity; slareport - SLA report; svggraph - Graph; systeminfo - System information; tophosts - Top hosts; trigover - Trigger overview; url - URL; web - Web monitoring.</td>
</tr>
</tbody>
</table>

Possible values: 10, 30, 60, 120, 600, 1800, 3600.

Default: 30.

Possible values:
0 - do not auto start slideshow;
1 - (default) auto start slideshow.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Custom widget name.</td>
</tr>
<tr>
<td>x</td>
<td>integer</td>
<td>A horizontal position from the left side of the dashboard.</td>
</tr>
<tr>
<td>y</td>
<td>integer</td>
<td>A vertical position from the top of the dashboard.</td>
</tr>
<tr>
<td>width</td>
<td>integer</td>
<td>The widget width.</td>
</tr>
<tr>
<td>height</td>
<td>integer</td>
<td>The widget height.</td>
</tr>
<tr>
<td>view_mode</td>
<td>integer</td>
<td>The widget view mode.</td>
</tr>
<tr>
<td>fields</td>
<td>array</td>
<td>Array of the dashboard widget field objects.</td>
</tr>
</tbody>
</table>

### Dashboard widget field

The dashboard widget field object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>integer</td>
<td>Type of the widget field.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Integer;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - String;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Host group;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - Host;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - Item;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - Item prototype;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 - Graph;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 - Graph prototype;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 - Map;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 - Service;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 - SLA.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Widget field name.</td>
</tr>
<tr>
<td>value</td>
<td>mixed</td>
<td>Widget field value depending of type.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Dashboard user group

List of dashboard permissions based on user groups. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usrgrpid</td>
<td>string</td>
<td>User group ID.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>permission</td>
<td>integer</td>
<td>Type of permission level.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - read only;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - read-write;</td>
</tr>
</tbody>
</table>

### Dashboard user

List of dashboard permissions based on users. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

903
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userid</td>
<td>string</td>
<td>User ID.</td>
</tr>
<tr>
<td>permission</td>
<td>integer</td>
<td>Type of permission level.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - read only;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - read-write;</td>
</tr>
</tbody>
</table>

**dashboard.create**

Description

**object dashboard.create(object/array dashboards)**

This method allows to create new dashboards.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Dashboards to create.

Additionally to the standard dashboard properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pages</td>
<td>array</td>
<td>Dashboard pages to be created for the dashboard. Dashboard pages will be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ordered in the same order as specified. At least one dashboard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>page object is required for pages property.</td>
</tr>
<tr>
<td>users</td>
<td>array</td>
<td>Dashboard user shares to be created on the dashboard.</td>
</tr>
<tr>
<td>userGroups</td>
<td>array</td>
<td>Dashboard user group shares to be created on the dashboard.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the created dashboards under the dashboard_ids property. The order of the returned IDs matches the order of the passed dashboards.

Examples

Creating a dashboard

Create a dashboard named "My dashboard" with one Problems widget with tags and using two types of sharing (user group and user) on a single dashboard page.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "problems",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                ...
                            }
                        ]
                    }
                ]
            }
        ]
    }
}
```
"type": 1,
"name": "tags.tag.0",
"value": "service"
},
{
"type": 0,
"name": "tags.operator.0",
"value": 1
},
{
"type": 1,
"name": "tags.value.0",
"value": "zabbix_server"
}
]
],
"userGroups": [  
{
"usrgrpid": "7",
"permission": 2
}
],
"users": [  
{
"userid": "4",
"permission": 3
}
],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
"jsonrpc": "2.0",
"result": {  
"dashboardids": [  
"2"
  ]
},
"id": 1
}

See also
- Dashboard page
- Dashboard widget
- Dashboard widget field
- Dashboard user
- Dashboard user group

Source
CDashboard::create() in ui/include/classes/api/services/CDashboard.php.

**dashboard.delete**

Description
object dashboard.delete(array dashboardids)
This method allows to delete dashboards.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(array) IDs of the dashboards to delete.

**Return values**

(object) Returns an object containing the IDs of the deleted dashboards under the dashboardids property.

**Examples**

**Deleting multiple dashboards**

Delete two dashboards.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "dashboard.delete",
    "params": [
        "2",
        "3"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2",
            "3"
        ]
    },
    "id": 1
}
```

Source

CDashboard::delete() in ui/include/classes/api/services/CDashboard.php.

**dashboard.get**

**Description**

integer/array dashboard.get(object parameters)

The method allows to retrieve dashboards according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dashboardids</td>
<td>string/array</td>
<td>Return only dashboards with the given IDs.</td>
</tr>
<tr>
<td>selectPages</td>
<td>query</td>
<td>Return a pages property with dashboard pages, correctly ordered.</td>
</tr>
<tr>
<td>selectUsers</td>
<td>query</td>
<td>Return a users property with users that the dashboard is shared with.</td>
</tr>
<tr>
<td>selectUserGroups</td>
<td>query</td>
<td>Return a userGroups property with user groups that the dashboard is shared with.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary page.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values
(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving a dashboard by ID

Retrieve all data about dashboards "1" and "2".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "dashboard.get",
    "params": {
        "output": "extend",
        "selectPages": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
        "dashboardids": [
            "1",
            "2"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dashboardid": "1",
            "name": "Dashboard",
            "userid": "1",
            "private": "0",
            "display_period": "30",
            "auto_start": "1",
            "users": [],
            "userGroups": [],
            "pages": [
                {
```
"dashboard_pageid": "1",
"name": "",
"display_period": "0",
"widgets": [
  {
    "widgetid": "9",
    "type": "systeminfo",
    "name": "",
    "x": "12",
    "y": "8",
    "width": "12",
    "height": "5",
    "view_mode": "0",
    "fields": []
  },
  {
    "widgetid": "8",
    "type": "problemsbysv",
    "name": "",
    "x": "12",
    "y": "4",
    "width": "12",
    "height": "4",
    "view_mode": "0",
    "fields": []
  },
  {
    "widgetid": "7",
    "type": "problemhosts",
    "name": "",
    "x": "12",
    "y": "0",
    "width": "12",
    "height": "4",
    "view_mode": "0",
    "fields": []
  },
  {
    "widgetid": "6",
    "type": "discovery",
    "name": "",
    "x": "6",
    "y": "9",
    "width": "6",
    "height": "4",
    "view_mode": "0",
    "fields": []
  },
  {
    "widgetid": "5",
    "type": "web",
    "name": "",
    "x": "0",
    "y": "9",
    "width": "6",
    "height": "4",
    "view_mode": "0",
    "fields": []
  },
  {
    "widgetid": "4",
    "type": "problems",
    "name": "",
    "x": "12",
    "y": "8",
    "width": "12",
    "height": "5",
    "view_mode": "0",
    "fields": []
  }
]
"name": "",
"x": "0",
"y": "3",
"width": "12",
"height": "6",
"view_mode": "0",
"fields": []
},
{
"widgetid": "3",
"type": "favmaps",
"name": "",
"x": "8",
"y": "0",
"width": "4",
"height": "3",
"view_mode": "0",
"fields": []
},
{
"widgetid": "1",
"type": "favgraphs",
"name": "",
"x": "0",
"y": "0",
"width": "4",
"height": "3",
"view_mode": "0",
"fields": []
}]}
},
{
"dashboard_pageid": "2",
"name": "",
"display_period": "0",
"widgets": []
},
{
"dashboard_pageid": "3",
"name": "Custom page name",
"display_period": "60",
"widgets": []
}
}
},
{
"dashboardid": "2",
"name": "My dashboard",
"userid": "1",
"private": "1",
"display_period": "60",
"auto_start": "1",
"users": [
{
"userid": "4",
"permission": "3"
}
],
"userGroups": [
{
"usrgrpid": "7",
"permission": "3"
}]
}
See also

- Dashboard page
- Dashboard widget
- Dashboard widget field
- Dashboard user
- Dashboard user group

Source

CDashboard::get() in ui/include/classes/api/services/CDashboard.php.

**dashboard.update**

Description

object dashboard.update(object/array dashboards)

This method allows to update existing dashboards.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Dashboard properties to be updated.

The dashboardid property must be specified for each dashboard, all other properties are optional. Only the specified properties will be updated.

Additionally to the standard dashboard properties, the method accepts the following parameters.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pages</td>
<td>array</td>
<td>Dashboard pages to replace the existing dashboard pages. Dashboard pages are updated by the dashboard_pageid property. New dashboard pages will be created for objects without dashboard_pageid property and the existing dashboard pages will be deleted if not reused. Dashboard pages will be ordered in the same order as specified. Only the specified properties of the dashboard pages will be updated. At least one dashboard page object is required for pages property.</td>
</tr>
<tr>
<td>users</td>
<td>array</td>
<td>Dashboard user shares to replace the existing elements.</td>
</tr>
<tr>
<td>userGroups</td>
<td>array</td>
<td>Dashboard user group shares to replace the existing elements.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated dashboards under the dashboardids property.

Examples

Renaming a dashboard

Rename a dashboard to “SQL server status”.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "dashboard.update",
    "params": {
        "dashboardid": "2",
        "name": "SQL server status"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [ "2"
    ]
},
    "id": 1
}
```

Updating dashboard pages

Rename the first dashboard page, replace widgets on the second dashboard page and add a new page as the third one. Delete all other dashboard pages.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "dashboard.update",
    "params": {
        "dashboardid": "2",
        "pages": [
            {
                "dashboard_pageid": 1,
                "name": 'Renamed Page'
            },
            {
                "dashboard_pageid": 2,
```
"widgets": [
  {
    "type": "clock",
    "x": 0,
    "y": 0,
    "width": 4,
    "height": 3
  }
],
{
  "display_period": 60
}
],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": {
    "dashboardids": [
      "2"
    ],
  },
  "id": 2
}

Change dashboard owner

Available only for admins and super admins.

Request:
{
  "jsonrpc": "2.0",
  "method": "dashboard.update",
  "params": {
    "dashboardid": "2",
    "userid": "1"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 2
}

Response:
{
  "jsonrpc": "2.0",
  "result": {
    "dashboardids": [
      "2"
    ],
  },
  "id": 2
}

See also

- Dashboard page
- Dashboard widget
- Dashboard widget field
- Dashboard user
- Dashboard user group
Discovered host

This class is designed to work with discovered hosts.

Object references:

- Discovered host

Available methods:

- dhost.get - retrieve discovered hosts

> Discovered host object

The following objects are directly related to the dhost API.

Discovered host

Discovered host are created by the Zabbix server and cannot be modified via the API.

The discovered host object contains information about a host discovered by a network discovery rule. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dhostid</td>
<td>string</td>
<td>ID of the discovered host.</td>
</tr>
<tr>
<td>druleid</td>
<td>string</td>
<td>ID of the discovery rule that detected the host.</td>
</tr>
<tr>
<td>lastdown</td>
<td>timestamp</td>
<td>Time when the discovered host last went down.</td>
</tr>
<tr>
<td>lastup</td>
<td>timestamp</td>
<td>Time when the discovered host last went up.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Whether the discovered host is up or down. A host is up if it has at least one active discovered service.</td>
</tr>
</tbody>
</table>

Possible values:
0 - host up;
1 - host down.

dhost.get

Description

integer/array dhost.get(object parameters)

The method allows to retrieve discovered hosts according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dhostids</td>
<td>string/array</td>
<td>Return only discovered hosts with the given iDs.</td>
</tr>
<tr>
<td>druleids</td>
<td>string/array</td>
<td>Return only discovered hosts that have been created by the given discovery rules.</td>
</tr>
<tr>
<td>dserviceids</td>
<td>string/array</td>
<td>Return only discovered hosts that are running the given services.</td>
</tr>
<tr>
<td>selectDRules</td>
<td>query</td>
<td>Return a drules property with an array of the discovery rules that detected the host.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>selectDServices</td>
<td>query</td>
<td>Return a dservices property with the discovered services running on the host.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supports count.</td>
</tr>
<tr>
<td>limitSelects</td>
<td>integer</td>
<td>Limits the number of records returned by subselects.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Applies to the following subselects: selectDServices - results will be sorted by dserviceid. Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Possible values are: dhostid and druleid.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:
- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieve discovered hosts by discovery rule

Retrieve all hosts and the discovered services they are running that have been detected by discovery rule "4".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dhost.get",
    "params": {
        "output": "extend",
        "selectDServices": "extend",
        "druleids": "4"
    },
    "auth": "038e1d7b1735c6a5436ee9eae096879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dservices": [
                {
                    "dserviceid": "1",
                    "dhostid": "1",
                    "type": "4",
                    "key": "",
                    "value": ""
                }
            ]
        }
    ]
}
```
"port": "80",
"status": "0",
"lastup": "1337697227",
"lastdown": "0",
"dcheckid": "5",
"ip": "192.168.1.1",
"dns": "station.company.lan"
}
],
"dhostid": "1",
"druleid": "4",
"status": "0",
"lastup": "1337697227",
"lastdown": "0"
},
{
"dservices": [
{
"dserviceid": "2",
"dhostid": "2",
"type": "4",
"key_": "",
"value": "",
"port": "80",
"status": "0",
"lastup": "1337697234",
"lastdown": "0",
"dcheckid": "5",
"ip": "192.168.1.4",
"dns": "john.company.lan"
}
],
"dhostid": "2",
"druleid": "4",
"status": "0",
"lastup": "1337697234",
"lastdown": "0"
},
{
"dservices": [
{
"dserviceid": "3",
"dhostid": "3",
"type": "4",
"key_": "",
"value": "",
"port": "80",
"status": "0",
"lastup": "1337697234",
"lastdown": "0",
"dcheckid": "5",
"ip": "192.168.1.26",
"dns": "printer.company.lan"
}
],
"dhostid": "3",
"druleid": "4",
"status": "0",
"lastup": "1337697234",
"lastdown": "0"
},
{

915


"dservices": [
    {
        "dserviceid": "4",
        "dhostid": "4",
        "type": "4",
        "key": "",
        "value": "",
        "port": "80",
        "status": "0",
        "lastup": "1337697234",
        "lastdown": "0",
        "dcheckid": "5",
        "ip": "192.168.1.7",
        "dns": "mail.company.lan"
    },
    {
        "dhostid": "4",
        "druleid": "4",
        "status": "0",
        "lastup": "1337697234",
        "lastdown": "0"
    }
],
"id": 1

See also
- Discovered service
- Discovery rule

Source
CDHost::get() in ui/include/classes/api/services/CDHost.php.

**Discovered service**

This class is designed to work with discovered services.

Object references:
- Discovered service

Available methods:
- dservice.get - retrieve discovered services

> **Discovered service object**

The following objects are directly related to the dservice API.

**Discovered service**

Discoveries are created by the Zabbix server and cannot be modified via the API.

The discovered service object contains information about a service discovered by a network discovery rule on a host. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dserviceid</td>
<td>string</td>
<td>ID of the discovered service.</td>
</tr>
<tr>
<td>dcheckid</td>
<td>string</td>
<td>ID of the discovery check used to detect the service.</td>
</tr>
<tr>
<td>dhostid</td>
<td>string</td>
<td>ID of the discovered host running the service.</td>
</tr>
<tr>
<td>dns</td>
<td>string</td>
<td>DNS of the host running the service.</td>
</tr>
<tr>
<td>ip</td>
<td>string</td>
<td>IP address of the host running the service.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>lastdown</td>
<td>timestamp</td>
<td>Time when the discovered service last went down.</td>
</tr>
<tr>
<td>lastup</td>
<td>timestamp</td>
<td>Time when the discovered service last went up.</td>
</tr>
<tr>
<td>port</td>
<td>integer</td>
<td>Service port number.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Status of the service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - service up;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - service down.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Value returned by the service when performing a Zabbix agent,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SNMPv1, SNMPv2 or SNMPv3 discovery check.</td>
</tr>
</tbody>
</table>

**dservice.get**

**Description**

integer/array dservice.get(object parameters)

The method allows to retrieve discovered services according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

**Parameters**

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dserviceids</td>
<td>string/array</td>
<td>Return only discovered services with the given IDs.</td>
</tr>
<tr>
<td>dhostids</td>
<td>string/array</td>
<td>Return only discovered services that belong to the given discovered hosts.</td>
</tr>
<tr>
<td>dcheckids</td>
<td>string/array</td>
<td>Return only discovered services that have been detected by the given discovery checks.</td>
</tr>
<tr>
<td>druleids</td>
<td>string/array</td>
<td>Return only discovered services that have been detected by the given discovery rules.</td>
</tr>
<tr>
<td>selectDRules</td>
<td>query</td>
<td>Return a drules property with an array of the discovery rules that detected the service.</td>
</tr>
<tr>
<td>selectDHosts</td>
<td>query</td>
<td>Return a dhosts property with an array the discovered hosts that the service belongs to.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return a hosts property with the hosts with the same IP address and proxy as the service.</td>
</tr>
<tr>
<td>limitSelects</td>
<td>integer</td>
<td>Limits the number of records returned by subselects.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Applies to the following subselects: selectHosts - result will be sorted by hostid.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are: dserviceid, dhostid and ip.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>
### Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

### Examples

Retrieve services discovered on a host

Retrieve all discovered services detected on discovered host "11".

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "dservice.get",
    "params": {
        "output": "extend",
        "dhostids": "11"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dserviceid": "12",
            "dhostid": "11",
            "value": "",
            "port": "80",
            "status": "1",
            "lastup": "0",
            "lastdown": "1348650607",
            "dcheckid": "5",
            "ip": "192.168.1.134",
            "dns": "john.local"
        },
        {
            "dserviceid": "13",
            "dhostid": "11",
            "value": "",
            "port": "21",
            "status": "1",
            "lastup": "0",
            "lastdown": "1348650610",
            "dcheckid": "6",
            "ip": "192.168.1.134",
            "dns": "john.local"
        }
    ],
    "id": 1
}
```

See also

- Discovered host
- Discovery check
Discovery check

This class is designed to work with discovery checks.

Object references:

- Discovery check

Available methods:

- `dcheck.get` - retrieve discovery checks

> Discovery check object

The following objects are directly related to the `dcheck` API.

Discovery check

The discovery check object defines a specific check performed by a network discovery rule. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dcheckid</code></td>
<td>string</td>
<td>(readonly) ID of the discovery check.</td>
</tr>
<tr>
<td><code>druleid</code></td>
<td>string</td>
<td>(readonly) ID of the discovery rule that the check belongs to.</td>
</tr>
<tr>
<td><code>key_</code></td>
<td>string</td>
<td>The value of this property differs depending on the type of the check: - key to query for Zabbix agent checks, required; - SNMP OID for SNMPv1, SNMPv2 and SNMPv3 checks, required.</td>
</tr>
<tr>
<td><code>ports</code></td>
<td>string</td>
<td>One or several port ranges to check separated by commas. Used for all checks except for ICMP.</td>
</tr>
<tr>
<td><code>snmp_community</code></td>
<td>string</td>
<td>SNMP community.</td>
</tr>
<tr>
<td><code>snmpv3_authpassphrase</code></td>
<td>string</td>
<td>Authentication passphrase used for SNMPv3 agent checks with security level set to authNoPriv or authPriv.</td>
</tr>
<tr>
<td><code>snmpv3_authprotocol</code></td>
<td>integer</td>
<td>Authentication protocol used for SNMPv3 agent checks with security level set to authNoPriv or authPriv. Possible values: 0 - (default) MD5; 1 - SHA1; 2 - SHA224; 3 - SHA256; 4 - SHA384; 5 - SHA512.</td>
</tr>
<tr>
<td><code>snmpv3_contextname</code></td>
<td>string</td>
<td>SNMPv3 context name. Used only by SNMPv3 checks.</td>
</tr>
<tr>
<td><code>snmpv3_privpassphrase</code></td>
<td>string</td>
<td>Privacy passphrase used for SNMPv3 agent checks with security level set to authPriv.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>snmpv3_privprotocol</td>
<td>integer</td>
<td>Privacy protocol used for SNMPv3 agent checks with security level set to authPriv. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) DES; 1 - AES128; 2 - AES192; 3 - AES256; 4 - AES192C; 5 - AES256C.</td>
</tr>
<tr>
<td>snmpv3_securitylevel</td>
<td>string</td>
<td>Security level used for SNMPv3 agent checks. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.</td>
</tr>
<tr>
<td>snmpv3_securityname</td>
<td>string</td>
<td>Security name used for SNMPv3 agent checks. Possible values:</td>
</tr>
<tr>
<td>type (required)</td>
<td>integer</td>
<td>Type of check. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - SSH; 1 - LDAP; 2 - SMTP; 3 - FTP; 4 - HTTP; 5 - POP; 6 - NNTP; 7 - IMAP; 8 - TCP; 9 - Zabbix agent; 10 - SNMPv1 agent; 11 - SNMPv2 agent; 12 - ICMP ping; 13 - SNMPv3 agent; 14 - HTTPS; 15 - Telnet.</td>
</tr>
<tr>
<td>uniq</td>
<td>integer</td>
<td>Whether to use this check as a device uniqueness criteria. Only a single unique check can be configured for a discovery rule. Used for Zabbix agent, SNMPv1, SNMPv2 and SNMPv3 agent checks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) do not use this check as a uniqueness criteria; 1 - use this check as a uniqueness criteria.</td>
</tr>
<tr>
<td>host_source</td>
<td>integer</td>
<td>Source for host name. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) DNS; 2 - IP; 3 - discovery value of this check.</td>
</tr>
<tr>
<td>name_source</td>
<td>integer</td>
<td>Source for visible name. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) not specified; 1 - DNS; 2 - IP; 3 - discovery value of this check.</td>
</tr>
</tbody>
</table>

dcheck.get

Description
integer/array dcheck.get(object parameters)

The method allows to retrieve discovery checks according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcheckids</td>
<td>string/array</td>
<td>Return only discovery checks with the given IDs.</td>
</tr>
<tr>
<td>druleids</td>
<td>string/array</td>
<td>Return only discovery checks that belong to the given discovery rules.</td>
</tr>
<tr>
<td>dserviceids</td>
<td>string/array</td>
<td>Return only discovery checks that have detected the given discovered services.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Possible values are: dcheckid and druleid.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieve discovery checks for a discovery rule

Retrieve all discovery checks used by discovery rule “6”.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "dcheck.get",
  "params": {
    "output": "extend",
    "dcheckids": "6"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": [
    {
      "dcheckid": "6",
```
### Discovery rule

This class is designed to work with network discovery rules.

This API is meant to work with network discovery rules. For the low-level discovery rules see the [LLD rule API](#).

Object references:

- Discovery rule

Available methods:

- `drule.create` - create new discovery rules
- `drule.delete` - delete discovery rules
- `drule.get` - retrieve discovery rules
- `drule.update` - update discovery rules

> Discovery rule object

The following objects are directly related to the `drule` API.

**Discovery rule**

The discovery rule object defines a network discovery rule. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>druleid</code></td>
<td>string</td>
<td>(readonly) ID of the discovery rule.</td>
</tr>
<tr>
<td><code>iprange</code></td>
<td>string</td>
<td>One or several IP ranges to check separated by commas.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>name</code></td>
<td>string</td>
<td>Name of the discovery rule.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>delay</code></td>
<td>string</td>
<td>Execution interval of the discovery rule. Accepts seconds, time unit with suffix and user macro.</td>
</tr>
<tr>
<td><code>nextcheck</code></td>
<td>timestamp</td>
<td>(readonly) Time when the discovery rule will be executed next.</td>
</tr>
<tr>
<td><code>proxy_hostid</code></td>
<td>string</td>
<td>ID of the proxy used for discovery.</td>
</tr>
</tbody>
</table>

Source

CDCheck::get() in ui/include/classes/api/services/CDCheck.php.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>integer</td>
<td>Whether the discovery rule is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) enabled;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - disabled.</td>
</tr>
</tbody>
</table>

### drule.create

**Description**

**object drule.create(object/array discoveryRules)**

This method allows to create new discovery rules.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) Discovery rules to create.

Additionally to the standard discovery rule properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dchecks</td>
<td>array</td>
<td>Discovery checks to create for the discovery rule.</td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns an object containing the IDs of the created discovery rules under the druleids property. The order of the returned IDs matches the order of the passed discovery rules.

**Examples**

Create a discovery rule

Create a discovery rule to find machines running the Zabbix agent in the local network. The rule must use a single Zabbix agent check on port 10050.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "drule.create",
    "params": {
        "name": "Zabbix agent discovery",
        "iprange": "192.168.1.1-255",
        "dchecks": [
            {
                "type": "9",
                "key_": "system.uname",
                "ports": "10050",
                "uniq": "0"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": {
```
See also

- Discovery check

Source

CDRule::create() in ui/include/classes/api/services/CDRule.php.

drule.delete

Description

object drule.delete(array discoveryRuleIds)

This method allows to delete discovery rules.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the discovery rules to delete.

Return values

(object) Returns an object containing the IDs of the deleted discovery rules under the druleids property.

Examples

Delete multiple discovery rules

Delete two discovery rules.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "drule.delete",
    "params": [
        "4",
        "6"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "4",
            "6"
        ]
    },
    "id": 1
}
```

Source

CDRule::delete() in ui/include/classes/api/services/CDRule.php.
**drule.get**

**Description**

`integer/array drule.get(object parameters)`

The method allows to retrieve discovery rules according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

*(object)* Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dhostids</td>
<td>string/array</td>
<td>Return only discovery rules that created the given discovered hosts.</td>
</tr>
<tr>
<td>druleids</td>
<td>string/array</td>
<td>Return only discovery rules with the given IDs.</td>
</tr>
<tr>
<td>dserviceids</td>
<td>string/array</td>
<td>Return only discovery rules that created the given discovered services.</td>
</tr>
<tr>
<td>selectDChecks</td>
<td>query</td>
<td>Return a <code>dchecks</code> property with the discovery checks used by the discovery rule. Supports count.</td>
</tr>
<tr>
<td>selectDHosts</td>
<td>query</td>
<td>Return a <code>dhosts</code> property with the discovered hosts created by the discovery rule. Supports count.</td>
</tr>
<tr>
<td>limitSelects</td>
<td>integer</td>
<td>Limits the number of records returned by subselects.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties. Possible values are: <code>druleid</code> and <code>name</code>.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Return values**

*(integer/array)* Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

**Examples**

Retrieve all discovery rules

Retrieve all configured discovery rules and the discovery checks they use.

**Request:**
{ "jsonrpc": "2.0",  "method": "drule.get",  "params": {  "output": "extend",  "selectDChecks": "extend" },  "auth": "038e1d7b1735c6a5436ee9eae095879e",  "id": 1 }  
Response:  
{ "jsonrpc": "2.0",  "result": [  {  "druleid": "2",  "proxy_hostid": "0",  "name": "Local network",  "iprange": "192.168.3.1-255",  "delay": "5s",  "nextcheck": "1348754327",  "status": "0",  "dchecks": [  {  "dcheckid": "7",  "druleid": "2",  "type": "3",  "key_": "",  "snmp_community": "",  "ports": "21",  "snmpv3_securityname": "",  "snmpv3_securitylevel": "0",  "snmpv3_authpassphrase": "",  "snmpv3_privpassphrase": "",  "uniq": "0",  "snmpv3_authprotocol": "0",  "snmpv3_privprotocol": "0",  "host_source": "1",  "name_source": "0"  },  {  "dcheckid": "9",  "druleid": "2",  "type": "4",  "key_": "",  "snmp_community": "",  "ports": "80",  "snmpv3_securityname": "",  "snmpv3_securitylevel": "0",  "snmpv3_authpassphrase": "",  "snmpv3_privpassphrase": "",  "uniq": "0",  "snmpv3_authprotocol": "0",  "snmpv3_privprotocol": "0",  "host_source": "1",  "name_source": "0"  }  ]  },  {  "druleid": "6",  "..."}  
926
"proxy_hostid": "0",
"name": "Zabbix agent discovery",
"iprange": "192.168.1.1-255",
"delay": "1h",
"nextcheck": "0",
"status": "0",
"dchecks": [
  {
    "dcheckid": "10",
    "druleid": "6",
    "type": "9",
    "key_*": "system.uname",
    "snmp_community": "",
    "ports": "10050",
    "snmpv3_securityname": "",
    "snmpv3_securitylevel": "0",
    "snmpv3_authpassphrase": "",
    "snmpv3_privpassphrase": "",
    "uniq": "0",
    "snmpv3_authprotocol": "0",
    "snmpv3_privprotocol": "0",
    "host_source": "2",
    "name_source": "3"
  }
],
"id": 1
}

See also
- Discovered host
- Discovery check

Source
CDRule::get() in include/classes/api/services/CDRule.php.

**drule.update**

Description
object drule.update(object/array discoveryRules)

This method allows to update existing discovery rules.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(object/array) Discovery rule properties to be updated.

The druleid property must be defined for each discovery rule, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard discovery rule properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dchecks</td>
<td>array</td>
<td>Discovery checks to replace existing checks.</td>
</tr>
</tbody>
</table>

Return values
(object) Returns an object containing the IDs of the updated discovery rules under the druleids property.

Examples
Change the IP range of a discovery rule

Change the IP range of a discovery rule to “192.168.2.1-255”.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "drule.update",
    "params": {
        "druleid": "6",
        "iprange": "192.168.2.1-255"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "6"
        ]
    },
    "id": 1
}
```

See also

- Discovery check

Source

CDRule::update() in ui/include/classes/api/services/CDRule.php.

**Event**

This class is designed to work with events.

Object references:

- Event

Available methods:

- event.get - retrieving events
- event.acknowledge - acknowledging events

> Event object

The following objects are directly related to the event API.

**Event**

Events are created by the Zabbix server and cannot be modified via the API.

The event object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventid</td>
<td>string</td>
<td>ID of the event.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>source</td>
<td>integer</td>
<td>Type of the event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - event created by a trigger;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - event created by a discovery rule;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - event created by active agent autoregistration;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - internal event;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - event created on service status update.</td>
</tr>
<tr>
<td>object</td>
<td>integer</td>
<td>Type of object that is related to the event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for trigger events:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - trigger.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for discovery events:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - discovered host;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - discovered service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for autoregistration events:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - auto-registered host.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for internal events:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - trigger;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - item;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - LLD rule.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for service events:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 - service.</td>
</tr>
<tr>
<td>objectid</td>
<td>string</td>
<td>ID of the related object.</td>
</tr>
<tr>
<td>acknowledged</td>
<td>integer</td>
<td>Whether the event has been acknowledged.</td>
</tr>
<tr>
<td>clock</td>
<td>timestamp</td>
<td>Time when the event was created.</td>
</tr>
<tr>
<td>ns</td>
<td>integer</td>
<td>Nanoseconds when the event was created.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Resolved event name.</td>
</tr>
<tr>
<td>value</td>
<td>integer</td>
<td>State of the related object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for trigger and service events:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - OK;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - problem.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for discovery events:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - host or service up;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - host or service down;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - host or service discovered;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - host or service lost.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for internal events:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - &quot;normal&quot; state;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - &quot;unknown&quot; or &quot;not supported&quot; state.</td>
</tr>
<tr>
<td>severity</td>
<td>integer</td>
<td>Event current severity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - not classified;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - information;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - warning;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - average;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - high;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - disaster.</td>
</tr>
<tr>
<td>r_eventid</td>
<td>string</td>
<td>Recovery event ID</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>c_eventid</td>
<td>string</td>
<td>ID of the event that was used to override (close) current event under global correlation rule. See correlationid to identify exact correlation rule. This parameter is only defined when the event is closed by global correlation rule.</td>
</tr>
<tr>
<td>correlationid</td>
<td>string</td>
<td>ID of the correlation rule that generated closing of the problem. This parameter is only defined when the event is closed by global correlation rule.</td>
</tr>
<tr>
<td>userid</td>
<td>string</td>
<td>User ID if the event was manually closed.</td>
</tr>
<tr>
<td>suppressed</td>
<td>integer</td>
<td>Whether the event is suppressed. Possible values: 0 - event is in normal state; 1 - event is suppressed.</td>
</tr>
<tr>
<td>opdata</td>
<td>string</td>
<td>Operational data with expanded macros.</td>
</tr>
<tr>
<td>urls</td>
<td>array of Media type URLs</td>
<td>Active media types URLs.</td>
</tr>
</tbody>
</table>

Event tag

The event tag object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Event tag name.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Event tag value.</td>
</tr>
</tbody>
</table>

Media type URLs

Object with media type url have the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Media type defined URL name.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>Media type defined URL value.</td>
</tr>
</tbody>
</table>

Results will contain entries only for active media types with enabled event menu entry. Macro used in properties will be expanded, but if one of properties contain non expanded macro both properties will be excluded from results. Supported macros described on page.

**event.acknowledge**

Description

**object event.acknowledge(object/array parameters)**

This method allows to update events. Following update actions can be performed:

- Close event. If event is already resolved, this action will be skipped.
- Acknowledge event. If event is already acknowledged, this action will be skipped.
- Unacknowledge event. If event is not acknowledged, this action will be skipped.
- Add message.
- Change event severity. If event already has same severity, this action will be skipped.
- Suppress event. If event is already suppressed, this action will be skipped.
- Unsuppress event. If event is not suppressed, this action will be skipped.

Only trigger events can be updated.
Only problem events can be updated.
Read/Write rights for trigger are required to close the event or to change event's severity.
To close an event, manual close should be allowed in the trigger.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.
Parameters

(object/array) Parameters containing the IDs of the events and update operations that should be performed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventids</td>
<td>string/object</td>
<td>IDs of the events to acknowledge.</td>
</tr>
<tr>
<td>action</td>
<td>integer</td>
<td>Event update action(s). This is bitmask field, any combination of values is acceptable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 1 - close problem; 2 - acknowledge event; 4 - add message; 8 - change severity; 16 - unacknowledge event; 32 - suppress event; 64 - unsuppress event.</td>
</tr>
<tr>
<td>message</td>
<td>string</td>
<td>Text of the message. Required, if action contains 'add message' flag.</td>
</tr>
<tr>
<td>severity</td>
<td>integer</td>
<td>New severity for events. Required, if action contains 'change severity' flag.</td>
</tr>
<tr>
<td>suppress_until</td>
<td>integer</td>
<td>Unix timestamp until which event must be suppressed. Required, if action contains 'suppress event' flag. Set '0' to make indefinitely suppression.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated events under the eventids property.

Examples

Acknowledging an event

Acknowledge a single event and leave a message.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "event.acknowledge",
    "params": {
        "eventids": "20427",
        "action": 6,
        "message": "Problem resolved."
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": ["20427"
    ]
```
Changing event’s severity

Change severity for multiple events and leave a message.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "event.acknowledge",
    "params": {
        "eventids": ["20427", "20428"],
        "action": 12,
        "message": "Maintenance required to fix it.",
        "severity": 4
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": ["20427", "20428"]
    },
    "id": 1
}
```

Source

CEvent::acknowledge() in ui/include/classes/api/services/CEvent.php.

**event.get**

Description

integer/array  event.get(object parameters)

The method allows to retrieve events according to the given parameters.

This method may return events of a deleted entity if these events have not been removed by the housekeeper yet.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventids</td>
<td>string/array</td>
<td>Return only events with the given IDs.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only events created by objects that belong to the given host groups.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only events created by objects that belong to the given hosts.</td>
</tr>
<tr>
<td>objectids</td>
<td>string/array</td>
<td>Return only events created by the given objects.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>source</td>
<td>integer</td>
<td>Return only events with the given type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to the event object page for a list of supported event types.</td>
</tr>
<tr>
<td>object</td>
<td>integer</td>
<td>Return only events created by objects of the given type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to the event object page for a list of supported object types.</td>
</tr>
<tr>
<td>acknowledged</td>
<td>boolean</td>
<td>If set to true return only acknowledged events.</td>
</tr>
<tr>
<td>suppressed</td>
<td>boolean</td>
<td>true - return only suppressed events; false - return events in the normal state.</td>
</tr>
<tr>
<td>severities</td>
<td>integer/array</td>
<td>Return only events with given event severities. Applies only if object is trigger.</td>
</tr>
<tr>
<td>evaltype</td>
<td>integer</td>
<td>Rules for tag searching.</td>
</tr>
<tr>
<td>tags</td>
<td>array of objects</td>
<td>Return only events with given tags. Exact match by tag and case-insensitive search by value and operator.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Format: [&quot;tag&quot;: &quot;&lt;tag&gt;&quot;, &quot;value&quot;: &quot;&lt;value&gt;&quot;, &quot;operator&quot;: &quot;&lt;operator&gt;&quot;], ...].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>An empty array returns all events.</td>
</tr>
<tr>
<td>eventid_from</td>
<td>string</td>
<td>Return only events with IDs greater or equal to the given ID.</td>
</tr>
<tr>
<td>eventid_till</td>
<td>string</td>
<td>Return only events with IDs less or equal to the given ID.</td>
</tr>
<tr>
<td>time_from</td>
<td>timestamp</td>
<td>Return only events that have been created after or at the given time.</td>
</tr>
<tr>
<td>time_till</td>
<td>timestamp</td>
<td>Return only events that have been created before or at the given time.</td>
</tr>
<tr>
<td>problem_time_from</td>
<td>timestamp</td>
<td>Returns only events that were in the problem state starting with problem_time_from. Applies only if the source is trigger event and object is trigger. Mandatory if problem_time_till is specified.</td>
</tr>
<tr>
<td>problem_time_till</td>
<td>timestamp</td>
<td>Returns only events that were in the problem state until problem_time_till. Applies only if the source is trigger event and object is trigger. Mandatory if problem_time_from is specified.</td>
</tr>
<tr>
<td>value</td>
<td>integer/array</td>
<td>Return only events with the given values.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return a hosts property with hosts containing the object that created the event. Supported only for events generated by triggers, items or LLD rules.</td>
</tr>
<tr>
<td>selectRelatedObject</td>
<td>query</td>
<td>Return a relatedObject property with the object that created the event. The type of object returned depends on the event type.</td>
</tr>
<tr>
<td>select_alerts</td>
<td>query</td>
<td>Return an alerts property with alerts generated by the event. Alerts are sorted in reverse chronological order.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>select_acknowledges</td>
<td>query</td>
<td>Return an acknowledges property with event updates. Event updates are sorted in reverse chronological order.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The event update object has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>acknowledgeid - (string) acknowledgment's ID;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>userid - (string) ID of the user that updated the event;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eventid - (string) ID of the updated event;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>clock - (timestamp) time when the event was updated;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>message - (string) text of the message;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>action - (integer) update action that was performed see event.acknowledge;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>old_severity - (integer) event severity before this update action;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>new_severity - (integer) event severity after this update action;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>username - (string) username of the user that updated the event;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>name - (string) name of the user that updated the event;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>surname - (string) surname of the user that updated the event.</td>
</tr>
<tr>
<td>selectTags</td>
<td>query</td>
<td>Return a tags property with event tags.</td>
</tr>
<tr>
<td>selectSuppressionData</td>
<td>query</td>
<td>Return a suppression_data property with the list of active maintenances and manual suppressions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>maintenanceid - (string) ID of the maintenance;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>userid - (string) ID of user who suppressed the event;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>suppress_until - (integer) time until the event is suppressed.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary page.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values
(integer/array) Returns either:
- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples
Retrieving trigger events
Retrieve the latest events from trigger "13926."

Request:
```
{
  "jsonrpc": "2.0",
  "method": "event.get",
  "params": {
    "output": "extend",
    "select_acknowledges": "extend",
    "selectTags": "extend",
    "selectSuppressionData": "extend",
```
Response:

```json
{
  "jsonrpc": "2.0",
  "result": [
    {
      "eventid": "9695",
      "source": "0",
      "object": "0",
      "objectid": "13926",
      "clock": "1347970410",
      "value": "1",
      "acknowledged": "1",
      "ns": "413316245",
      "name": "MySQL is down",
      "severity": "5",
      "r_eventid": "0",
      "c_eventid": "0",
      "correlationid": "0",
      "userid": "0",
      "opdata": "",
      "acknowledges": [
        {
          "acknowledgeid": "1",
          "userid": "1",
          "eventid": "9695",
          "clock": "1350640590",
          "message": "Problem resolved.\n\r\n----[BULK ACKNOWLEDGE]----",
          "action": "6",
          "old_severity": "0",
          "new_severity": "0",
          "username": "Admin",
          "name": "Zabbix",
          "surname": "Administrator"
        }
      ],
      "suppression_data": [
        {
          "maintenanceid": "15",
          "suppress_until": "1472511600",
          "userid": "0"
        }
      ],
      "suppressed": "1",
      "tags": [
        {
          "tag": "service",
          "value": "mysqld"
        },
        {
          "tag": "error",
          "value": ""
        }
      ]
    }
  ]
}
```
Retrieve events by time period

Retrieve all events that have been created between October 9 and 10, 2012, in reverse chronological order.

Request:

```
{
   "jsonrpc": "2.0",
   "method": "event.get",
   "params": {
      "output": "extend",
      "time_from": "1349797228",
      "time_till": "1350661228",
      "sortfield": ["clock", "eventid"],
      "sortorder": "desc"
   },
   "auth": "038e1d7b1735c6a5436ee9eae096879e",
   "id": 1
}
```

Response:

```
{
   "jsonrpc": "2.0",
   "result": [
      {
         "eventid": "20616",
         "source": "0",
         "object": "0",
         "objectid": "14282",
         "clock": "1350477814",
         "value": "1",
         "acknowledged": "0",
         "ns": "0",
         "name": "Less than 25% free in the history cache",
         "severity": "3",
         "r_eventid": "0",
         "c_eventid": "0",
         "correlationid": "0",
         "userid": "0",
         "acknowledges": [],
         "suppression_data": [],
         "suppressed": "0",
         "tags": []
      }
   ]
}
```
See also

- Alert
- Item
- Host
- LLD rule
- Trigger

Source

CEvent::get() in ui/include/classes/api/services/CEvent.php.

Graph

This class is designed to work with items.

Object references:

- Graph
Available methods:

- `graph.create` - creating new graphs
- `graph.delete` - deleting graphs
- `graph.get` - retrieving graphs
- `graph.update` - updating graphs

> Graph object

The following objects are directly related to the graph API.

Graph

The graph object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>graphid</code></td>
<td>string</td>
<td>(readonly) ID of the graph.</td>
</tr>
<tr>
<td><code>height</code></td>
<td>integer</td>
<td>Height of the graph in pixels.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>string</td>
<td>Name of the graph</td>
</tr>
<tr>
<td><code>width</code></td>
<td>integer</td>
<td>Width of the graph in pixels.</td>
</tr>
<tr>
<td><code>flags</code></td>
<td>integer</td>
<td>(readonly) Origin of the graph.</td>
</tr>
<tr>
<td><code>graphtype</code></td>
<td>integer</td>
<td>Graph's layout type.</td>
</tr>
<tr>
<td><code>percent_left</code></td>
<td>float</td>
<td>Left percentile.</td>
</tr>
<tr>
<td><code>percent_right</code></td>
<td>float</td>
<td>Right percentile.</td>
</tr>
<tr>
<td><code>show_3d</code></td>
<td>integer</td>
<td>Whether to show pie and exploded graphs in 3D.</td>
</tr>
<tr>
<td><code>show_legend</code></td>
<td>integer</td>
<td>Whether to show the legend on the graph.</td>
</tr>
<tr>
<td><code>show_work_period</code></td>
<td>integer</td>
<td>Whether to show the working time on the graph.</td>
</tr>
<tr>
<td><code>show_triggers</code></td>
<td>integer</td>
<td>Whether to show the trigger line on the graph.</td>
</tr>
<tr>
<td><code>templateid</code></td>
<td>string</td>
<td>(readonly) ID of the parent template graph.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>yaxismax</td>
<td>float</td>
<td>The fixed maximum value for the Y axis. Default: 100.</td>
</tr>
<tr>
<td>yaxismin</td>
<td>float</td>
<td>The fixed minimum value for the Y axis. Default: 0.</td>
</tr>
<tr>
<td>ymax_itemid</td>
<td>string</td>
<td>ID of the item that is used as the maximum value for the Y axis. Starting with Zabbix 6.2.1, if user have no access to specified item, the graph is rendered like ymax_type would be set to '0' (calculated).</td>
</tr>
<tr>
<td>ymax_type</td>
<td>integer</td>
<td>Maximum value calculation method for the Y axis. Possible values: 0 - (default) calculated; 1 - fixed; 2 - item.</td>
</tr>
<tr>
<td>ymin_itemid</td>
<td>string</td>
<td>ID of the item that is used as the minimum value for the Y axis. Starting with Zabbix 6.2.1, if user have no access to specified item, the graph is rendered like ymin_type would be set to '0' (calculated).</td>
</tr>
<tr>
<td>ymin_type</td>
<td>integer</td>
<td>Minimum value calculation method for the Y axis. Possible values: 0 - (default) calculated; 1 - fixed; 2 - item.</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Universal unique identifier, used for linking imported graphs to already existing ones. Used only for graphs on templates. Auto-generated, if not given. For update operations this field is readonly.</td>
</tr>
</tbody>
</table>

**graph.create**

Description

object graph.create(object/array graphs)

This method allows to create new graphs.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Graphs to create.

Additionally to the standard graph properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gitems</td>
<td>array</td>
<td>Graph items to be created for the graph. (required)</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the created graphs under the graphids property. The order of the returned IDs matches the order of the passed graphs.

Examples

Creating a graph

Create a graph with two items.

Request:
{  
  "jsonrpc": "2.0",
  "method": "graph.create",
  "params": {
    "name": "MySQL bandwidth",
    "width": 900,
    "height": 200,
    "gitems": [
      {
        "itemid": "22828",
        "color": "00AA00"
      },
      {
        "itemid": "22829",
        "color": "3333FF"
      }
    ],
    "auth": "038e1d7b1735c6a5436ee9eae096879e",
    "id": 1
  }
}

Response:

{
  "jsonrpc": "2.0",
  "result": {
    "graphids": ["652"
  ],
  "id": 1
}
}

See also

- Graph item

Source

CGraph::create() in ui/include/classes/api/services/CGraph.php.

**graph.delete**

Description

object graph.delete(array graphIds)

This method allows to delete graphs.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the graphs to delete.

Return values

(object) Returns an object containing the IDs of the deleted graphs under the graphIds property.

Examples

Deleting multiple graphs

Delete two graphs.

Request:
Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652",
            "653"
        ]
    },
    "id": 1
}
```

Source

`CGraph::delete()` in `ui/include/classes/api/services/CGraph.php`.

**graph.get**

**Description**

`integer/array graph.get(object parameters)`

The method allows to retrieve graphs according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

*(object)* Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphids</td>
<td>string/array</td>
<td>Return only graphs with the given IDs.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only graphs that belong to hosts or templates in the given host groups or template groups.</td>
</tr>
<tr>
<td>templateIds</td>
<td>string/array</td>
<td>Return only graph that belong to the given templates.</td>
</tr>
<tr>
<td>hostIds</td>
<td>string/array</td>
<td>Return only graphs that belong to the given hosts.</td>
</tr>
<tr>
<td>itemIds</td>
<td>string/array</td>
<td>Return only graphs that contain the given items.</td>
</tr>
<tr>
<td>templated</td>
<td>boolean</td>
<td>If set to <code>true</code> return only graphs that belong to templates.</td>
</tr>
<tr>
<td>inherited</td>
<td>boolean</td>
<td>If set to <code>true</code> return only graphs inherited from a template.</td>
</tr>
<tr>
<td>expandName</td>
<td>flag</td>
<td>Expand macros in the graph name.</td>
</tr>
<tr>
<td>selectHostGroups</td>
<td>query</td>
<td>Return a <code>host groups</code> property with the host groups that the graph belongs to.</td>
</tr>
<tr>
<td>selectTemplateGroups</td>
<td>query</td>
<td>Return a <code>template groups</code> property with the template groups that the graph belongs to.</td>
</tr>
<tr>
<td>selectTemplates</td>
<td>query</td>
<td>Return a <code>templates</code> property with the templates that the graph belongs to.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return a <code>hosts</code> property with the hosts that the graph belongs to.</td>
</tr>
<tr>
<td>selectItems</td>
<td>query</td>
<td>Return an <code>items</code> property with the items used in the graph.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>selectGraphDiscovery</td>
<td>query</td>
<td>Return a <code>graphDiscovery</code> property with the graph discovery object. The graph discovery objects links the graph to a graph prototype from which it was created.</td>
</tr>
<tr>
<td>selectGraphItems</td>
<td>query</td>
<td>Return a <code>gitems</code> property with the items used in the graph.</td>
</tr>
<tr>
<td>selectDiscoveryRule</td>
<td>query</td>
<td>Return a <code>discoveryRule</code> property with the low-level discovery rule that created the graph.</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td>Return only those results that exactly match the given filter.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td>Possible values are: <code>graphid</code>, <code>name</code> and <code>graphtype</code>.</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td>These parameters are common for all get methods are described in detail in the reference commentary page.</td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td>This parameter is deprecated, please use <code>selectHostGroups</code> or <code>selectTemplateGroups</code> instead.</td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>selectGroups</td>
<td>query</td>
<td>Return a <code>groups</code> property with the host groups and template groups that the graph belongs to.</td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving graphs from hosts

Retrieve all graphs from host "10107" and sort them by name.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "graph.get",
    "params": {
        "output": "extend",
        "hostids": 10107,
        "sortfield": "name"
    },
    "auth": "038e1d7b1735c6a5436ee9eae096879e"
}
```
Response:
{
    "jsonrpc": "2.0",
    "result": [
        {
            "graphid": "612",
            "name": "CPU jumps",
            "width": "900",
            "height": "200",
            "yaxismin": "0",
            "yaxismax": "100",
            "templateid": "439",
            "show_work_period": "1",
            "show_triggers": "1",
            "graphtype": "0",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0",
            "percent_right": "0",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "613",
            "name": "CPU load",
            "width": "900",
            "height": "200",
            "yaxismin": "0",
            "yaxismax": "100",
            "templateid": "433",
            "show_work_period": "1",
            "show_triggers": "1",
            "graphtype": "0",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0",
            "percent_right": "0",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "614",
            "name": "CPU utilization",
            "width": "900",
            "height": "200",
            "yaxismin": "0",
            "yaxismax": "100",
            "templateid": "387",
            "show_work_period": "1",
            "show_triggers": "1",
            "graphtype": "1",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0",
            "percent_right": "0",
            "ymin_type": "1",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        }
    ]
}
See also

- Discovery rule
- Graph item
- Item
- Host
- Host group
- Template
- Template group

Source

CGraph::get() in ui/include/classes/api/services/CGraph.php.

**graph.update**

**Description**

object graph.update(object/array graphs)

This method allows to update existing graphs.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) Graph properties to be updated.

The graphid property must be defined for each graph, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard graph properties the method accepts the following parameters.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gitems</td>
<td>array</td>
<td>Graph items to replace existing graph items. If a graph item has the gitemid property defined it will be updated, otherwise a new graph item will be created.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated graphs under the graphids property.

Examples

Setting the maximum for the Y scale

Set the maximum of the Y scale to a fixed value of 100.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "graph.update",
    "params": {
        "graphid": "439",
        "ymax_type": 1,
        "yaxismax": 100
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "439"
        ]
    },
    "id": 1
}
```

Source

CGraph::update() in ui/include/classes/api/services/CGraph.php.

Graph item

This class is designed to work with hosts.

Object references:

- Graph item

Available methods:

- graphitem.get - retrieving graph items

> Graph item object

The following objects are directly related to the graphitem API.

Graph item

Graph items can only be modified via the graph API.

The graph item object has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gitemid</td>
<td>string</td>
<td>(readonly) ID of the graph item.</td>
</tr>
<tr>
<td>color</td>
<td>string</td>
<td>Graph item’s draw color as a hexadecimal color code.</td>
</tr>
<tr>
<td>itemid</td>
<td>string</td>
<td>ID of the item.</td>
</tr>
<tr>
<td>calc_fnc</td>
<td>integer</td>
<td>Value of the item that will be displayed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - minimum value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - (default) average value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - maximum value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 - all values;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 - last value, used only by pie and exploded graphs.</td>
</tr>
<tr>
<td>drawtype</td>
<td>integer</td>
<td>Draw style of the graph item.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) line;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - filled region;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - bold line;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - dot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - dashed line;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - gradient line.</td>
</tr>
<tr>
<td>graphid</td>
<td>string</td>
<td>ID of the graph that the graph item belongs to.</td>
</tr>
<tr>
<td>sortorder</td>
<td>integer</td>
<td>Position of the item in the graph.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: starts with 0 and increases by one with each entry.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>Type of graph item.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) simple;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - graph sum, used only by pie and exploded graphs.</td>
</tr>
<tr>
<td>yaxisside</td>
<td>integer</td>
<td>Side of the graph where the graph item’s Y scale will be drawn.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) left side;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - right side.</td>
</tr>
</tbody>
</table>

**graphitem.get**

Description

integer/array graphitem.get(object parameters)

The method allows to retrieve graph items according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphids</td>
<td>string/array</td>
<td>Return only graph items that belong to the given graphs.</td>
</tr>
<tr>
<td>itemids</td>
<td>string/array</td>
<td>Return only graph items with the given item IDs.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>Return only graph items with the given type.</td>
</tr>
</tbody>
</table>

Refer to the graph item object page for a list of supported graph item types.
Return a graphs property with an array of graphs that the item belongs to.

Sort the result by the given properties.

Possible values are: gitemid. These parameters being common for all get methods are described in detail in the reference commentary page.

Examples

Retrieving graph items from a graph

Retrieve all graph items used in a graph with additional information about the item and the host.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "graphitem.get",
    "params": {
        "output": "extend",
        "graphids": ["387"]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "gitemid": "1242",
            "graphid": "387",
            "itemid": "22665",
            "drawtype": "1",
            "sortorder": "1",
            "color": "FF5555",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0",
            "key_": "system.cpu.util[,steal][]",
            "hostid": "10001",
            "flags": "0",
            "host": "Linux"
        },
        {
            "gitemid": "1243",
            "graphid": "387",
            "itemid": "22668",
            "drawtype": "1",
            "sortorder": "2",
```

947
Graph prototype

This class is designed to work with graph prototypes.

Object references:

- Graph prototype

Available methods:

- graphprototype.create - creating new graph prototypes
- graphprototype.delete - deleting graph prototypes
- graphprototype.get - retrieving graph prototypes
- graphprototype.update - updating graph prototypes

Graph prototype object

The following objects are directly related to the graphprototype API.

Graph prototype

The graph prototype object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphid</td>
<td>string</td>
<td>(readonly) ID of the graph prototype.</td>
</tr>
<tr>
<td>height</td>
<td>integer</td>
<td>Height of the graph prototype in pixels.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>name</strong></td>
<td>string</td>
<td>Name of the graph prototype.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>width</strong></td>
<td>integer</td>
<td>Width of the graph prototype in pixels.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>graphtype</strong></td>
<td>integer</td>
<td>Graph prototypes's layout type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) normal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - stacked;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - pie;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - exploded.</td>
</tr>
<tr>
<td><strong>percent_left</strong></td>
<td>float</td>
<td>Left percentile.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.</td>
</tr>
<tr>
<td><strong>percent_right</strong></td>
<td>float</td>
<td>Right percentile.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.</td>
</tr>
<tr>
<td><strong>show_3d</strong></td>
<td>integer</td>
<td>Whether to show discovered pie and exploded graphs in 3D.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) show in 2D;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - show in 3D.</td>
</tr>
<tr>
<td><strong>show_legend</strong></td>
<td>integer</td>
<td>Whether to show the legend on the discovered graph.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - hide;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) show.</td>
</tr>
<tr>
<td><strong>show_work_period</strong></td>
<td>integer</td>
<td>Whether to show the working time on the discovered graph.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - hide;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) show.</td>
</tr>
<tr>
<td><strong>templateid</strong></td>
<td>string</td>
<td>(readonly) ID of the parent template graph prototype.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>yaxismax</strong></td>
<td>float</td>
<td>The fixed maximum value for the Y axis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>yaxismin</strong></td>
<td>float</td>
<td>The fixed minimum value for the Y axis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ymax_itemid</strong></td>
<td>string</td>
<td>ID of the item that is used as the maximum value for the Y axis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Starting with Zabbix 6.2.1, if user have no access to specified item, the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>graph is rendered like ymax_type would be set to '0' (calculated).</td>
</tr>
<tr>
<td><strong>ymax_type</strong></td>
<td>integer</td>
<td>Maximum value calculation method for the Y axis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) calculated;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - fixed;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - item.</td>
</tr>
<tr>
<td><strong>ymin_itemid</strong></td>
<td>string</td>
<td>ID of the item that is used as the minimum value for the Y axis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Starting with Zabbix 6.2.1, if user have no access to specified item, the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>graph is rendered like ymin_type would be set to '0' (calculated).</td>
</tr>
<tr>
<td><strong>ymin_type</strong></td>
<td>integer</td>
<td>Minimum value calculation method for the Y axis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) calculated;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - fixed;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - item.</td>
</tr>
<tr>
<td><strong>discover</strong></td>
<td>integer</td>
<td>Graph prototype discovery status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) new graphs will be discovered;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - new graphs will not be discovered and existing graphs will be marked as lost.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Universal unique identifier, used for linking imported graph prototypes to already existing ones. Used only for graph prototypes on templates. Auto-generated, if not given. For update operations this field is readonly.</td>
</tr>
</tbody>
</table>

**graphprototype.create**

**Description**

object graphprototype.create(object/array graphPrototypes)

This method allows to create new graph prototypes.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) Graph prototypes to create.

Additionally to the standard graph prototype properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gitems</td>
<td>array</td>
<td>Graph items to be created for the graph prototypes. Graph items can reference both items and item prototypes, but at least one item prototype must be present.</td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns an object containing the IDs of the created graph prototypes under the graphids property. The order of the returned IDs matches the order of the passed graph prototypes.

**Examples**

Creating a graph prototype

Create a graph prototype with two items.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "graphprototype.create",
    "params": {
        "name": "Disk space usage {#FSNAME}"
        "width": 900,
        "height": 200,
        "gitems": [
            {
                "itemid": "22828",
                "color": "00AA00"
            },
            {
                "itemid": "22829",
                "color": "3333FF"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:
graphprototype.delete

Description

object graphprototype.delete(array graphPrototypeIds)

This method allows to delete graph prototypes.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

[array] IDs of the graph prototypes to delete.

Return values

(object) Returns an object containing the IDs of the deleted graph prototypes under the graphids property.

Examples

Deleting multiple graph prototypes

Delete two graph prototypes.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "graphprototype.delete",
    "params": [
        "652",
        "653"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652",
            "653"
        ],
        "id": 1
    }
}
```

Source

C哲G哲Pr哲o哲totype哲::create() in ui/include/classes/api/services/C哲G哲raphPrototype哲.php.
graphprototype.get

Description

integer/array graphprototype.get(object parameters)

The method allows to retrieve graph prototypes according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>discoveryids</td>
<td>string/array</td>
<td>Return only graph prototypes that belong to the given discovery rules.</td>
</tr>
<tr>
<td>graphids</td>
<td>string/array</td>
<td>Return only graph prototypes with the given IDs.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only graph prototypes that belong to hosts or templates in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>given host groups or template groups.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only graph prototypes that belong to the given hosts.</td>
</tr>
<tr>
<td>inherited</td>
<td>boolean</td>
<td>If set to true return only graph prototypes inherited from a template.</td>
</tr>
<tr>
<td>itemids</td>
<td>string/array</td>
<td>Return only graph prototypes that contain the given item prototypes.</td>
</tr>
<tr>
<td>templated</td>
<td>boolean</td>
<td>If set to true return only graph prototypes that belong to templates.</td>
</tr>
<tr>
<td>templateids</td>
<td>string/array</td>
<td>Return only graph prototypes that belong to the given templates.</td>
</tr>
<tr>
<td>selectDiscoveryRule</td>
<td>query</td>
<td>Return a discoveryRule property with the LLD rule that the graph</td>
</tr>
<tr>
<td>selectGraphItems</td>
<td>query</td>
<td>Return a gitems property with the graph items used in the graph prototype.</td>
</tr>
<tr>
<td>selectHostGroups</td>
<td>query</td>
<td>Return a host groups property with the host groups that the graph prototype</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return a hosts property with the hosts that the graph prototype belongs to.</td>
</tr>
<tr>
<td>selectItems</td>
<td>query</td>
<td>Return an items property with the items and item prototypes used in the</td>
</tr>
<tr>
<td>selectTemplateGroups</td>
<td>query</td>
<td>Return a template groups property with the template groups that the graph</td>
</tr>
<tr>
<td>selectTemplates</td>
<td>query</td>
<td>Return a templates property with the templates that the graph prototype</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td>Return only those results that exactly match the given filter.</td>
</tr>
</tbody>
</table>

Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against.

Supports additional filters:

- host - technical name of the host that the graph prototype belongs to;
- hostid - ID of the host that the graph prototype belongs to.

Sort the result by the given properties.

Possible values are: graphid, name and graphtype. These parameters being common for all get methods are described in detail in the reference commentary.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td>This parameter is deprecated, please use selectHostGroups or selectTemplateGroups instead.</td>
</tr>
<tr>
<td>selectGroups</td>
<td>query</td>
<td>Return a groups property with the host groups and template groups that the graph prototype belongs to.</td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving graph prototypes from a LLD rule

Retrieve all graph prototypes from an LLD rule.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "graphprototype.get",
    "params": {
        "output": "extend",
        "discoverids": "27426"
    },
    "auth": "038e1d7b1735c6a5436ee9eae096879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "graphid": "1017",
            "parent_itemid": "27426",
            "name": "Disk space usage {#FSNAME}"
        },
        {
            "graphid": "1017",
            "parent_itemid": "27426",
            "name": "Disk space usage {#FSNAME}",
            "width": "600",
            "height": "340",
            "yaxismin": "0.0000",
            "yaxismax": "0.0000",
            "templateid": "442",
            "show_work_period": "0",
            "show_triggers": "0",
            "graphtype": "2",
            "show_legend": "1",
            "show_3d": "1",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "discover": "0"
        }
    ],
    "id": 1
}
```

See also

- Discovery rule
Graphitem
Item
Host
Host group
Template
Template group

Source
CGraphPrototype::get() in ui/include/classes/api/services/CGraphPrototype.php.

**graphprototype.update**

**Description**

object graphprototype.update(object/array graphPrototypes)

This method allows to update existing graph prototypes.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) Graph prototype properties to be updated.

The graphid property must be defined for each graph prototype, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard graph prototype properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gitems</td>
<td>array</td>
<td>Graph items to replace existing graph items. If a graph item has the gitemid property defined it will be updated, otherwise a new graph item will be created.</td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns an object containing the IDs of the updated graph prototypes under the graphids property.

**Examples**

Changing the size of a graph prototype

Change the size of a graph prototype to 1100 to 400 pixels.

Request:

```
{
   "jsonrpc": "2.0",
   "method": "graphprototype.update",
   "params": {
      "graphid": "439",
      "width": 1100,
      "height": 400
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```

Response:

```
{
   "jsonrpc": "2.0",
   "result": {
      "graphids": [
         "439"
      ]
   }
}
```
High availability node

This class is designed to work with server nodes that are part of a High availability cluster, or a standalone server instance.

Object references:

- High availability node

Available methods:

- hanode.get - retrieving nodes

> High availability node object

The following object is related to operating a High availability cluster of Zabbix servers.

High availability node

Nodes are created by the Zabbix server and cannot be modified via the API.

The High availability node object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ha_nodeid</td>
<td>string</td>
<td>ID of the node.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name assigned to the node, using the HANodeName configuration entry of zabbix_server.conf. Empty for a server running in standalone mode.</td>
</tr>
<tr>
<td>address</td>
<td>string</td>
<td>IP or DNS name where the node connects from.</td>
</tr>
<tr>
<td>port</td>
<td>integer</td>
<td>Port on which the node is running.</td>
</tr>
<tr>
<td>lastaccess</td>
<td>integer</td>
<td>Heartbeat time, t.i. time of last update from the node. UTC timestamp.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>State of the node. Possible values: 0 - standby; 1 - stopped manually; 2 - unavailable; 3 - active.</td>
</tr>
</tbody>
</table>

hanode.get

Description

integer/array hanode.get(object parameters)

The method allows to retrieve a list of High availability cluster nodes according to the given parameters.

This method is only available to Super admin user types. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ha_nodeids</td>
<td>string/array</td>
<td>Return only nodes with the given node IDs.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td>Return only those results that exactly match the given filter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accepts an array, where the keys are property names, and the values</td>
</tr>
<tr>
<td></td>
<td></td>
<td>are either a single value or an array of values to match against.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>flag</td>
<td>Possible values are: name, lastaccess, status.</td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td>These parameters being common for all get methods are described in</td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td>detail in the reference commentary.</td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Get a list of nodes ordered by status

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "hanode.get",
    "params": {
        "preservekeys": true,
        "sortfield": "status",
        "sortorder": "DESC"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "ckuo7i1nw000h0sajj3l3hh8u": {
            "ha_nodeid": "ckuo7i1nw000h0sajj3l3hh8u",
            "name": "node-active",
            "address": "192.168.1.13",
            "port": "10051",
            "lastaccess": "1635335704",
            "status": "3"
        },
        "ckuo7i1nw000e0sajwfttc1mp": {
            "ha_nodeid": "ckuo7i1nw000e0sajwfttc1mp",
            "name": "node6",
            "address": "192.168.1.10",
            "port": "10053",
            "lastaccess": "1635332902",
            "status": "2"
        },
        "ckuo7i1nv000c0sajz85xcrtt": {
            "ha_nodeid": "ckuo7i1nv000c0sajz85xcrtt",
```
Get a list of specific nodes by their IDs

Request:

```
{  
    "jsonrpc": "2.0",  
    "method": "hanode.get",  
    "params": {  
        "ha_nodeids": ["ckuo7i1nv000e0sajwfttc1mp", "ckuo7i1nv000c0sajz85xcrtt"]  
    },  
    "auth": "3a57200802b24cda67c4e4010b50c065",  
    "id": 1  
}
```

Response:

```
{  
    "jsonrpc": "2.0",  
    "result": [  
        {  
            "ha_nodeid": "ckuo7i1nv000c0sajz85xcrtt",  
            "name": "node4",  
            "address": "192.168.1.8",  
            "port": "10052",  
            "lastaccess": "1635334214",  
            "status": "1"  
        },  
        {  
            "ha_nodeid": "ckuo7i1nw000e0sajwfttc1mp",  
            "name": "node6",  
            "address": "192.168.1.10",  
            "port": "10053",  
            "lastaccess": "1635332902",  
            "status": "2"  
        }  
    ],  
    "id": 1  
}
```

Get a list of stopped nodes

Request:

```
{  
    "jsonrpc": "2.0",  
    "method": "hanode.get",  
    "params": {  
        "output": ["ha_nodeid", "address", "port"]  
    },  
    "auth": "3a57200802b24cda67c4e4010b50c065",  
    "id": 1  
}
```
"filter": {
    "status": 1
},
"auth": "3a57200802b24cda67c4e4010b50c065",
"id": 1
}

Get a count of standby nodes

Request:
{
    "jsonrpc": "2.0",
    "method": "hanode.get",
    "params": {
        "countOutput": true,
        "filter": {
            "status": 0
        }
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}

Response:
{
    "jsonrpc": "2.0",
    "result": "3",
    "id": 1
}

Check status of nodes at specific IP addresses

Request:
{
    "jsonrpc": "2.0",
    "method": "hanode.get",
    "params": {
        "output": ["name", "status"],
        "filter": {
            "ha_nodeid": "ckuo7i1nw000g0sajjsjre7e3",
            "address": "192.168.1.12",
            "port": "10051"
        },
        "ha_nodeid": "ckuo7i1nv000c0sajz85xcrtt",
        "address": "192.168.1.8",
        "port": "10052"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}

Response:
{
    "jsonrpc": "2.0",
    "result": ["ha_nodeid": "ckuo7i1nv000d0sajd95y1b6x",
                "address": "192.168.1.9",
                "port": "10053"],
    "id": 1
}
"address": ["192.168.1.7", "192.168.1.13"],
"auth": "3a57200802b24cda67c4e4010b50c065",
"id": 1
}
}

Response:
{
"jsonrpc": "2.0",
"result": [
{
"name": "node3",
"status": "0"
},
{
"name": "node-active",
"status": "3"
}
],
"id": 1
}

Source
CHaNode::get() in ui/include/classes/api/services/CHaNode.php.

History

This class is designed to work with history data.

Object references:

- History

Available methods:

- history.get - retrieving history data.

> History object

The following objects are directly related to the history API.

History objects differ depending on the item's type of information. They are created by the Zabbix server and cannot be modified via the API.

Float history

The float history object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock</td>
<td>timestamp</td>
<td>Time when that value was received.</td>
</tr>
<tr>
<td>itemid</td>
<td>string</td>
<td>ID of the related item.</td>
</tr>
<tr>
<td>ns</td>
<td>integer</td>
<td>Nanoseconds when the value was received.</td>
</tr>
<tr>
<td>value</td>
<td>float</td>
<td>Received value.</td>
</tr>
</tbody>
</table>

Integer history

The integer history object has the following properties.
Property | Type | Description
--- | --- | ---
clock | timestamp | Time when that value was received.
itemid | string | ID of the related item.
ns | integer | Nanoseconds when the value was received.
value | integer | Received value.

Text history

The text history object has the following properties.

Property | Type | Description
--- | --- | ---
id | string | ID of the history entry.
clock | timestamp | Time when that value was received.
itemid | string | ID of the related item.
ns | integer | Nanoseconds when the value was received.
value | text | Received value.

Log history

The log history object has the following properties.

Property | Type | Description
--- | --- | ---
id | string | ID of the history entry.
clock | timestamp | Time when that value was received.
itemid | string | ID of the related item.
logeventid | integer | Windows event log entry ID.
ns | integer | Nanoseconds when the value was received.
severity | integer | Windows event log entry level.
source | string | Windows event log entry source.
timestamp | timestamp | Windows event log entry time.
value | text | Received value.

**history.clear**

Description

**object history.clear(array itemids)**

This method allows to clear item history.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See **User roles** for more information.

Parameters

(array) IDs of items to clear.

Return values

(object) Returns an object containing the IDs of the cleared items under the itemids property.

Examples
Clear history

Request:
{
   "jsonrpc": "2.0",
   "method": "history.clear",
   "params": [
      "10325",
      "13205"
   ],
   "auth": "038e1d7b1735c6a5436ee9eae096879e",
   "id": 1
}

Response:
{
   "jsonrpc": "2.0",
   "result": {
      "itemids": [
         "10325",
         "13205"
      ]
   },
   "id": 1
}

Source
CHistory::clear() in ui/include/classes/api/services/CHistory.php.

**history.get**

Description
integer/array history.get(object parameters)

The method allows to retrieve history data according to the given parameters.

See also: known issues

This method may return historical data of a deleted entity if this data has not been removed by the housekeeper yet.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>history</td>
<td>integer</td>
<td>History object types to return.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - numeric float;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - character;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - log;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - numeric unsigned;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - text.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 3.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only history from the given hosts.</td>
</tr>
<tr>
<td>itemids</td>
<td>string/array</td>
<td>Return only history from the given items.</td>
</tr>
<tr>
<td>time_from</td>
<td>timestamp</td>
<td>Return only values that have been received after or at the given time.</td>
</tr>
<tr>
<td>time_till</td>
<td>timestamp</td>
<td>Return only values that have been received before or at the given time.</td>
</tr>
</tbody>
</table>
Parameter | Type | Description
--- | --- | ---
sortfield | string/array | Sort the result by the given properties.

Possible values are: itemid and clock.

countOutput | boolean | These parameters being common for all get methods are described in detail in the reference commentary page.

editable | boolean | excludeSearch | boolean
filter | object | limit | integer
output | query | search | object
searchByAny | boolean | searchWildcardsEnabled | boolean
sortorder | string/array | startSearch | boolean

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving item history data

Return 10 latest values received from a numeric(float) item.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "history.get",
   "params": {
      "output": "extend",
      "history": 0,
      "itemids": "23296",
      "sortfield": "clock",
      "sortorder": "DESC",
      "limit": 10
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": [
      {
         "itemid": "23296",
         "clock": "1351090996",
         "value": "0.085",
         "ns": "563157632"
      },
      {
         "itemid": "23296",
         "clock": "1351090936",
         "value": "0.16",
         "ns": "549216402"
      },
      {
         "itemid": "23296",
      }
   ]
}
```
"clock": "1351090876",
"value": "0.18",
"ns": "537418114"
},
{
  "itemid": "23296",
  "clock": "1351090816",
  "value": "0.21",
  "ns": "522659528"
},
{
  "itemid": "23296",
  "clock": "1351090756",
  "value": "0.215",
  "ns": "507809457"
},
{
  "itemid": "23296",
  "clock": "1351090696",
  "value": "0.255",
  "ns": "495509699"
},
{
  "itemid": "23296",
  "clock": "1351090636",
  "value": "0.36",
  "ns": "477708209"
},
{
  "itemid": "23296",
  "clock": "1351090576",
  "value": "0.375",
  "ns": "463251343"
},
{
  "itemid": "23296",
  "clock": "1351090516",
  "value": "0.315",
  "ns": "447947017"
},
{
  "itemid": "23296",
  "clock": "1351090456",
  "value": "0.275",
  "ns": "435307141"
}
],
"id": 1
}

Source

CHistory::get() in ui/include/classes/api/services/CHistory.php.

Host

This class is designed to work with hosts.

Object references:

- Host
- Host inventory
Available methods:

- `host.create` - creating new hosts
- `host.delete` - deleting hosts
- `host.get` - retrieving hosts
- `host.massadd` - adding related objects to hosts
- `host.massremove` - removing related objects from hosts
- `host.massupdate` - replacing or removing related objects from hosts
- `host.update` - updating hosts

> Host object

The following objects are directly related to the host API.

Host

The host object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostid</td>
<td>string</td>
<td>(readonly) ID of the host.</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>Technical name of the host.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>text</td>
<td>Description of the host.</td>
</tr>
<tr>
<td>flags</td>
<td>integer</td>
<td>(readonly) Origin of the host.</td>
</tr>
<tr>
<td>inventory_mode</td>
<td>integer</td>
<td>Host inventory population mode.</td>
</tr>
<tr>
<td>ipmi_authtype</td>
<td>integer</td>
<td>IPMI authentication algorithm.</td>
</tr>
<tr>
<td>ipmi_password</td>
<td>string</td>
<td>IPMI password.</td>
</tr>
<tr>
<td>ipmi_privilege</td>
<td>integer</td>
<td>IPMI privilege level.</td>
</tr>
<tr>
<td>maintenance_from</td>
<td>timestamp</td>
<td>(readonly) Starting time of the effective maintenance.</td>
</tr>
<tr>
<td>maintenance_status</td>
<td>integer</td>
<td>(readonly) Effective maintenance status.</td>
</tr>
</tbody>
</table>

Possible values:

- `hostid`:
  - `(readonly)`
- `host`:
  - Technical name of the host
- `description`:
  - Description of the host
- `flags`:
  - Possible values:
    - 0 - a plain host;
    - 4 - a discovered host
- `inventory_mode`:
  - Possible values are:
    - -1 - (default) disabled;
    - 0 - manual;
    - 1 - automatic
- `ipmi_authtype`:
  - Possible values are:
    - -1 - (default) default;
    - 0 - none;
    - 1 - MD2;
    - 2 - MD5
    - 4 - straight;
    - 5 - OEM;
    - 6 - RMCP+
- `ipmi_password`:
- `ipmi_privilege`:
  - Possible values are:
    - 1 - callback;
    - 2 - (default) user;
    - 3 - operator;
    - 4 - admin;
    - 5 - OEM
- `ipmi_username`:
- `maintenance_from`:
- `maintenance_status`:
  - Possible values are:
    - 0 - (default) no maintenance;
    - 1 - maintenance in effect
### Property | Type | Description
--- | --- | ---
maintenance_type | integer | (readonly) Effective maintenance type.

Possible values are:
0 - (default) maintenance with data collection;
1 - maintenance without data collection.

maintenanceid | string | (readonly) ID of the maintenance that is currently in effect on the host.

name | string | Visible name of the host.

Default: host property value.

proxy_hostid | string | ID of the proxy that is used to monitor the host.

status | integer | Status and function of the host.

Possible values are:
0 - (default) monitored host;
1 - unmonitored host.

tls_connect | integer | Connections to host.

Possible values are:
1 - (default) No encryption;
2 - PSK;
4 - certificate.

tls_accept | integer | Connections from host.

Possible bitmap values are:
1 - (default) No encryption;
2 - PSK;
4 - certificate.

tlsIssuer | string | Certificate issuer.

tls_subject | string | Certificate subject.

tls_psk_identity | string | (write-only) PSK identity. Required if either tls_connect or tls_accept has PSK enabled.

Do not put sensitive information in the PSK identity, it is transmitted unencrypted over the network to inform a receiver which PSK to use.

tls_psk | string | (write-only) The preshared key, at least 32 hex digits. Required if either tls_connect or tls_accept has PSK enabled.

active_available | integer | (readonly) Host active interface availability status.

Possible values are:
0 - interface status is unknown;
1 - interface is available;
2 - interface is not available.

---

**Host inventory**

The host inventory object has the following properties.

Each property has it's own unique ID number, which is used to associate host inventory fields with items.

<table>
<thead>
<tr>
<th>ID</th>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>alias</td>
<td>string</td>
<td>Alias.</td>
</tr>
<tr>
<td>11</td>
<td>asset_tag</td>
<td>string</td>
<td>Asset tag.</td>
</tr>
<tr>
<td>28</td>
<td>chassis</td>
<td>string</td>
<td>Chassis.</td>
</tr>
<tr>
<td>23</td>
<td>contact</td>
<td>string</td>
<td>Contact person.</td>
</tr>
<tr>
<td>32</td>
<td>contract_number</td>
<td>string</td>
<td>Contract number.</td>
</tr>
<tr>
<td>47</td>
<td>date_hw_decomm</td>
<td>string</td>
<td>HW decommissioning date.</td>
</tr>
<tr>
<td>46</td>
<td>date_hw_expiry</td>
<td>string</td>
<td>HW maintenance expiry date.</td>
</tr>
<tr>
<td>45</td>
<td>date_hw_install</td>
<td>string</td>
<td>HW installation date.</td>
</tr>
<tr>
<td>44</td>
<td>date_hw_purchase</td>
<td>string</td>
<td>HW purchase date.</td>
</tr>
<tr>
<td>34</td>
<td>deployment_status</td>
<td>string</td>
<td>Deployment status.</td>
</tr>
<tr>
<td>14</td>
<td>hardware</td>
<td>string</td>
<td>Hardware.</td>
</tr>
<tr>
<td>15</td>
<td>hardware_full</td>
<td>string</td>
<td>Detailed hardware.</td>
</tr>
<tr>
<td>ID</td>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-------------------</td>
<td>------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>39</td>
<td>host_netmask</td>
<td>string</td>
<td>Host subnet mask.</td>
</tr>
<tr>
<td>38</td>
<td>host_networks</td>
<td>string</td>
<td>Host networks.</td>
</tr>
<tr>
<td>40</td>
<td>host_router</td>
<td>string</td>
<td>Host router.</td>
</tr>
<tr>
<td>30</td>
<td>hw_arch</td>
<td>string</td>
<td>HW architecture.</td>
</tr>
<tr>
<td>33</td>
<td>installer_name</td>
<td>string</td>
<td>Installer name.</td>
</tr>
<tr>
<td>24</td>
<td>location</td>
<td>string</td>
<td>Location.</td>
</tr>
<tr>
<td>25</td>
<td>location_lat</td>
<td>string</td>
<td>Location latitude.</td>
</tr>
<tr>
<td>26</td>
<td>location_lon</td>
<td>string</td>
<td>Location longitude.</td>
</tr>
<tr>
<td>12</td>
<td>macaddress_a</td>
<td>string</td>
<td>MAC address A.</td>
</tr>
<tr>
<td>13</td>
<td>macaddress_b</td>
<td>string</td>
<td>MAC address B.</td>
</tr>
<tr>
<td>29</td>
<td>model</td>
<td>string</td>
<td>Model.</td>
</tr>
<tr>
<td>3</td>
<td>name</td>
<td>string</td>
<td>Name.</td>
</tr>
<tr>
<td>27</td>
<td>notes</td>
<td>string</td>
<td>Notes.</td>
</tr>
<tr>
<td>41</td>
<td>oob_ip</td>
<td>string</td>
<td>OOB IP address.</td>
</tr>
<tr>
<td>42</td>
<td>oob_netmask</td>
<td>string</td>
<td>OOB host subnet mask.</td>
</tr>
<tr>
<td>43</td>
<td>oob_router</td>
<td>string</td>
<td>OOB router.</td>
</tr>
<tr>
<td>5</td>
<td>os</td>
<td>string</td>
<td>OS name.</td>
</tr>
<tr>
<td>6</td>
<td>os_full</td>
<td>string</td>
<td>Detailed OS name.</td>
</tr>
<tr>
<td>7</td>
<td>os_short</td>
<td>string</td>
<td>Short OS name.</td>
</tr>
<tr>
<td>61</td>
<td>poc_1_cell</td>
<td>string</td>
<td>Primary POC mobile number.</td>
</tr>
<tr>
<td>58</td>
<td>poc_1_email</td>
<td>string</td>
<td>Primary email.</td>
</tr>
<tr>
<td>57</td>
<td>poc_1_name</td>
<td>string</td>
<td>Primary POC name.</td>
</tr>
<tr>
<td>63</td>
<td>poc_1_notes</td>
<td>string</td>
<td>Primary POC notes.</td>
</tr>
<tr>
<td>59</td>
<td>poc_1_phone_a</td>
<td>string</td>
<td>Primary POC phone A.</td>
</tr>
<tr>
<td>60</td>
<td>poc_1_phone_b</td>
<td>string</td>
<td>Primary POC phone B.</td>
</tr>
<tr>
<td>62</td>
<td>poc_1_screen</td>
<td>string</td>
<td>Primary POC screen name.</td>
</tr>
<tr>
<td>68</td>
<td>poc_2_cell</td>
<td>string</td>
<td>Secondary POC mobile number.</td>
</tr>
<tr>
<td>65</td>
<td>poc_2_email</td>
<td>string</td>
<td>Secondary POC email.</td>
</tr>
<tr>
<td>64</td>
<td>poc_2_name</td>
<td>string</td>
<td>Secondary POC name.</td>
</tr>
<tr>
<td>70</td>
<td>poc_2_notes</td>
<td>string</td>
<td>Secondary POC notes.</td>
</tr>
<tr>
<td>66</td>
<td>poc_2_phone_a</td>
<td>string</td>
<td>Secondary POC phone A.</td>
</tr>
<tr>
<td>67</td>
<td>poc_2_phone_b</td>
<td>string</td>
<td>Secondary POC phone B.</td>
</tr>
<tr>
<td>69</td>
<td>poc_2_screen</td>
<td>string</td>
<td>Secondary POC screen name.</td>
</tr>
<tr>
<td>8</td>
<td>serialno_a</td>
<td>string</td>
<td>Serial number A.</td>
</tr>
<tr>
<td>9</td>
<td>serialno_b</td>
<td>string</td>
<td>Serial number B.</td>
</tr>
<tr>
<td>48</td>
<td>site_address_a</td>
<td>string</td>
<td>Site address A.</td>
</tr>
<tr>
<td>49</td>
<td>site_address_b</td>
<td>string</td>
<td>Site address B.</td>
</tr>
<tr>
<td>50</td>
<td>site_address_c</td>
<td>string</td>
<td>Site address C.</td>
</tr>
<tr>
<td>51</td>
<td>site_city</td>
<td>string</td>
<td>Site city.</td>
</tr>
<tr>
<td>53</td>
<td>site_country</td>
<td>string</td>
<td>Site country.</td>
</tr>
<tr>
<td>56</td>
<td>site_notes</td>
<td>string</td>
<td>Site notes.</td>
</tr>
<tr>
<td>55</td>
<td>site_rack</td>
<td>string</td>
<td>Site rack location.</td>
</tr>
<tr>
<td>52</td>
<td>site_state</td>
<td>string</td>
<td>Site state.</td>
</tr>
<tr>
<td>54</td>
<td>site_zip</td>
<td>string</td>
<td>Site ZIP/postal code.</td>
</tr>
<tr>
<td>16</td>
<td>software</td>
<td>string</td>
<td>Software.</td>
</tr>
<tr>
<td>18</td>
<td>software_app_a</td>
<td>string</td>
<td>Software application A.</td>
</tr>
<tr>
<td>19</td>
<td>software_app_b</td>
<td>string</td>
<td>Software application B.</td>
</tr>
<tr>
<td>20</td>
<td>software_app_c</td>
<td>string</td>
<td>Software application C.</td>
</tr>
<tr>
<td>21</td>
<td>software_app_d</td>
<td>string</td>
<td>Software application D.</td>
</tr>
<tr>
<td>22</td>
<td>software_app_e</td>
<td>string</td>
<td>Software application E.</td>
</tr>
<tr>
<td>17</td>
<td>software_full</td>
<td>string</td>
<td>Software details.</td>
</tr>
<tr>
<td>10</td>
<td>tag</td>
<td>string</td>
<td>Tag.</td>
</tr>
<tr>
<td>1</td>
<td>type</td>
<td>string</td>
<td>Type.</td>
</tr>
<tr>
<td>2</td>
<td>type_full</td>
<td>string</td>
<td>Type details.</td>
</tr>
<tr>
<td>35</td>
<td>url_a</td>
<td>string</td>
<td>URL A.</td>
</tr>
<tr>
<td>36</td>
<td>url_b</td>
<td>string</td>
<td>URL B.</td>
</tr>
<tr>
<td>37</td>
<td>url_c</td>
<td>string</td>
<td>URL C.</td>
</tr>
<tr>
<td>31</td>
<td>vendor</td>
<td>string</td>
<td>Vendor.</td>
</tr>
</tbody>
</table>

Host tag
The host tag object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Host tag name.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Host tag value.</td>
</tr>
<tr>
<td>automatic</td>
<td>integer</td>
<td>Type of host tag.</td>
</tr>
</tbody>
</table>

Possible values are:
0 - (default) manual (tag created by user);
1 - automatic (tag created by low-level discovery)

host.create

Description

object host.create(object/array hosts)

This method allows to create new hosts.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Hosts to create.

Additionally to the standard host properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Host groups to add the host to.</td>
</tr>
<tr>
<td>interfaces</td>
<td>object/array</td>
<td>Interfaces to be created for the host.</td>
</tr>
<tr>
<td>tags</td>
<td>object/array</td>
<td>Host tags.</td>
</tr>
<tr>
<td>templates</td>
<td>object/array</td>
<td>Templates to be linked to the host.</td>
</tr>
<tr>
<td>macros</td>
<td>object/array</td>
<td>The templates must have the templateid property defined.</td>
</tr>
<tr>
<td>inventory</td>
<td>object</td>
<td>User macros to be created for the host.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the created hosts under the hostids property. The order of the returned IDs matches the order of the passed hosts.

Examples

Creating a host

Create a host called “Linux server” with an IP interface and tags, add it to a group, link a template to it and set the MAC addresses in the host inventory.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "Linux server",
        "interfaces": [
            {
                "type": 1,
                "main": 1,
                "useip": 1,
                "ip": "192.168.3.1"
            }
        ],
        "groups": [
            "group1"
        ]
    }
}
```
"dns": "",
"port": "10050"
],
"groups": [
{
"groupid": "50"
}
],
"tags": [
{
"tag": "Host name",
"value": "Linux server"
}
],
"templates": [
{
"templateid": "20045"
}
],
"macros": [
{
"macro": "${USER_ID}"
},
{
"macro": "${USER_LOCATION}"
}
],
"inventory_mode": 0,
"inventory": {
"macaddress_a": "01234",
"macaddress_b": "56768"
}
],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1}
}

Response:
{
"jsonrpc": "2.0",
"result": {
"hostids": [
"107819"
]
},
"id": 1
}

Creating a host with SNMP interface

Create a host called "SNMP host" with an SNMPv3 interface with details.

Request:
{
"jsonrpc": "2.0",
"method": "host.create",
"params": {
"host": "SNMP host",
"interfaces": [


```json
{
    "type": 2,
    "main": 1,
    "useip": 1,
    "ip": "127.0.0.1",
    "dns": "",
    "port": "161",
    "details": {
        "version": 3,
        "bulk": 0,
        "securityname": "mysecurityname",
        "contextname": "",
        "securitylevel": 1
    }
}

[
    {
        "groupid": "4"
    }
]

"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10658"
        ]
    },
    "id": 1
}

See also
- Host group
- Template
- User macro
- Host interface
- Host inventory
- Host tag

Source
CHost::create() in ui/include/classes/api/services/CHost.php.

**host.delete**

Description

object host.delete(array hosts)

This method allows to delete hosts.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of hosts to delete.

Return values


(object) Returns an object containing the IDs of the deleted hosts under the hostids property.

Examples

Deleting multiple hosts

Delete two hosts.

Request:

```
{
   "jsonrpc": "2.0",
   "method": "host.delete",
   "params": [
      "13",
      "32"
   ],
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```

Response:

```
{
   "jsonrpc": "2.0",
   "result": {
      "hostids": [
         "13",
         "32"
      ]
   },
   "id": 1
}
```

Source

CHost::delete() in ui/include/classes/api/services/CHost.php.

**host.get**

Description

integer/array host.get(object parameters)

The method allows to retrieve hosts according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only hosts that belong to the given groups.</td>
</tr>
<tr>
<td>dserviceis</td>
<td>string/array</td>
<td>Return only hosts that are related to the given discovered services.</td>
</tr>
<tr>
<td>graphids</td>
<td>string/array</td>
<td>Return only hosts that have the given graphs.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only hosts with the given host IDs.</td>
</tr>
<tr>
<td>httptestids</td>
<td>string/array</td>
<td>Return only hosts that have the given web checks.</td>
</tr>
<tr>
<td>interfaceids</td>
<td>string/array</td>
<td>Return only hosts that use the given interfaces.</td>
</tr>
<tr>
<td>itemids</td>
<td>string/array</td>
<td>Return only hosts that have the given items.</td>
</tr>
<tr>
<td>maintenanceids</td>
<td>string/array</td>
<td>Return only hosts that are affected by the given maintenances.</td>
</tr>
<tr>
<td>monitored_hosts</td>
<td>flag</td>
<td>Return only monitored hosts.</td>
</tr>
<tr>
<td>proxy_hosts</td>
<td>flag</td>
<td>Return only proxies.</td>
</tr>
<tr>
<td>proxyis</td>
<td>string/array</td>
<td>Return only hosts that are monitored by the given proxies.</td>
</tr>
<tr>
<td>templatedHosts</td>
<td>flag</td>
<td>Return both hosts and templates.</td>
</tr>
<tr>
<td>templateids</td>
<td>string/array</td>
<td>Return only hosts that are linked to the given templates.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>triggerids</td>
<td>string/array</td>
<td>Return only hosts that have the given triggers.</td>
</tr>
<tr>
<td>with_items</td>
<td>flag</td>
<td>Return only hosts that have items.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overrides the with_monitored_items and with_simple_graph_items parameters.</td>
</tr>
<tr>
<td>with_item_prototypes</td>
<td>flag</td>
<td>Return only hosts that have item prototypes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overrides the with_simple_graph_item_prototypes parameter.</td>
</tr>
<tr>
<td>with_graphs</td>
<td>flag</td>
<td>Return only hosts that have graphs.</td>
</tr>
<tr>
<td>with_graph_prototypes</td>
<td>flag</td>
<td>Return only hosts that have graph prototypes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return only hosts that have web checks.</td>
</tr>
<tr>
<td>with_monitored_graphs</td>
<td>flag</td>
<td>Overrides the with_monitored_http_tests parameter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return only hosts that have enabled web checks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return only hosts that have enabled items.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overrides the with_simple_graph_items parameter.</td>
</tr>
<tr>
<td>with_monitored_triggers</td>
<td>flag</td>
<td>Return only hosts that have enabled triggers. All of the items used in the trigger must also be enabled.</td>
</tr>
<tr>
<td>with_simple_graph_items</td>
<td>flag</td>
<td>Return only hosts that have items with numeric type of information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return only hosts that have triggers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overrides the with_monitored_triggers parameter.</td>
</tr>
<tr>
<td>withProblemsSuppressed</td>
<td>boolean</td>
<td>Return hosts that have suppressed problems.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: null - (default) all hosts; true - only hosts with suppressed problems; false - only hosts with unsuppressed problems.</td>
</tr>
<tr>
<td>evaltype</td>
<td>integer</td>
<td>Rules for tag searching.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) And/Or; 2 - Or.</td>
</tr>
<tr>
<td>severities</td>
<td>integer/array</td>
<td>Return hosts that have only problems with given severities. Applies only if problem object is trigger.</td>
</tr>
<tr>
<td>tags</td>
<td>array/object</td>
<td>Return only hosts with given tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Format: [{&quot;tag&quot;: &quot;&lt;tag&gt;&quot;, &quot;value&quot;: &quot;&lt;value&gt;&quot;, &quot;operator&quot;: &quot;&lt;operator&gt;&quot;}, ...]. An empty array returns all hosts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible operator values: 0 - (default) Contains; 1 - Equals; 2 - Not like; 3 - Not equal; 4 - Exists; 5 - Not exists.</td>
</tr>
<tr>
<td>inheritedTags</td>
<td>boolean</td>
<td>Return hosts that have given tags also in all of their linked templates. Default:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: true - linked templates must also have given tags; false - (default) linked template tags are ignored.</td>
</tr>
<tr>
<td>selectDiscoveries</td>
<td>query</td>
<td>Return a discoveries property with host low-level discovery rules. Supports count.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>selectDiscoveryRule</td>
<td>query</td>
<td>Return a <code>discoveryRule</code> property with the low-level discovery rule that created the host (from host prototype in VMware monitoring).</td>
</tr>
<tr>
<td>selectGraphs</td>
<td>query</td>
<td>Return a <code>graphs</code> property with host graphs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supports count.</td>
</tr>
<tr>
<td>selectHostDiscovery</td>
<td>query</td>
<td>Return a <code>hostDiscovery</code> property with host discovery object data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The host discovery object links a discovered host to a host prototype or a host prototypes to an LLD rule and has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>host - (string) host of the host prototype;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hostid - (string) ID of the discovered host or host prototype;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parent_hostid - (string) ID of the host prototype from which the host has been created;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parent_itemid - (string) ID of the LLD rule that created the discovered host;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lastcheck - (timestamp) time when the host was last discovered;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ts_delete - (timestamp) time when a host that is no longer discovered will be deleted.</td>
</tr>
<tr>
<td>selectHostGroups</td>
<td>query</td>
<td>Return a <code>host groups</code> property with host groups data that the host belongs to.</td>
</tr>
<tr>
<td>selectHttpTests</td>
<td>query</td>
<td>Return an <code>httpTests</code> property with host web scenarios.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supports count.</td>
</tr>
<tr>
<td>selectInterfaces</td>
<td>query</td>
<td>Return an <code>interfaces</code> property with host interfaces.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supports count.</td>
</tr>
<tr>
<td>selectInventory</td>
<td>query</td>
<td>Return an <code>inventory</code> property with host inventory data.</td>
</tr>
<tr>
<td>selectItems</td>
<td>query</td>
<td>Return an <code>items</code> property with host items.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supports count.</td>
</tr>
<tr>
<td>selectMacros</td>
<td>query</td>
<td>Return a <code>macros</code> property with host macros.</td>
</tr>
<tr>
<td>selectParentTemplates</td>
<td>query</td>
<td>Return a <code>parentTemplates</code> property with templates that the host is linked to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In addition to Template object fields, it contains <code>link_type</code> - (integer) the way that the template is linked to host.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) manually linked; 1 - automatically linked by LLD.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supports count.</td>
</tr>
<tr>
<td>selectDashboards</td>
<td>query</td>
<td>Return a <code>dashboards</code> property.</td>
</tr>
<tr>
<td>selectTags</td>
<td>query</td>
<td>Return a <code>tags</code> property with host tags.</td>
</tr>
<tr>
<td>selectInheritedTags</td>
<td>query</td>
<td>Return an <code>inheritedTags</code> property with tags that are on all templates which are linked to host.</td>
</tr>
<tr>
<td>selectTriggers</td>
<td>query</td>
<td>Return a <code>triggers</code> property with host triggers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supports count.</td>
</tr>
<tr>
<td>selectValueMaps</td>
<td>query</td>
<td>Return a <code>valuemaps</code> property with host value maps.</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td>Return only those results that exactly match the given filter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allows filtering by interface properties.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>limitSelects</td>
<td>integer</td>
<td>Limits the number of records returned by subselects.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Applies to the following subselects:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>selectParentTemplates - results will be sorted by host;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>selectInterfaces;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>selectItems - sorted by name;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>selectDiscoveries - sorted by name;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>selectTriggers - sorted by description;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>selectGraphs - sorted by name;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>selectDashboards - sorted by name.</td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td>Return results that match the given wildcard search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accepts an array, where the keys are property names, and the values are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>strings to search for. If no additional options are given, this will</td>
</tr>
<tr>
<td></td>
<td></td>
<td>perform a LIKE &quot;%...%&quot; search.</td>
</tr>
<tr>
<td>searchInventory</td>
<td>object</td>
<td>Allows searching by interface properties. Works only with text fields.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return only hosts that have inventory data matching the given wildcard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>search.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Possible values are: hostid, host, name, status.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td>the reference commentary.</td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>selectGroups</td>
<td>query</td>
<td>This parameter is deprecated, please use selectHostGroups instead.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return a groups property with host groups data that the host belongs to.</td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving data by name

Retrieve all data about two hosts named “Zabbix server” and “Linux server”.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "filter": {
            "host": ["Zabbix server", "Linux server"]
        }
    }
}  ```
Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10160",
            "proxy_hostid": "0",
            "host": "Zabbix server",
            "status": "0",
            "lastaccess": "0",
            "ipmi_authtype": "-1",
            "ipmi_privilege": "2",
            "ipmi_username": "",
            "ipmi_password": "",
            "maintenanceid": "0",
            "maintenance_status": "0",
            "maintenance_type": "0",
            "maintenance_from": "0",
            "name": "Zabbix server",
            "flags": "0",
            "description": "The Zabbix monitoring server.",
            "tls_connect": "1",
            "tls_accept": "1",
            "tlsIssuer": "",
            "tlsSubject": "",
            "inventory_mode": "1",
            "active_available": "1"
        },
        {
            "hostid": "10167",
            "proxy_hostid": "0",
            "host": "Linux server",
            "status": "0",
            "ipmi_authtype": "-1",
            "ipmi_privilege": "2",
            "ipmi_username": "",
            "ipmi_password": "",
            "maintenanceid": "0",
            "maintenance_status": "0",
            "maintenance_type": "0",
            "maintenance_from": "0",
            "name": "Linux server",
            "flags": "0",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tlsIssuer": "",
            "tlsSubject": "",
            "inventory_mode": "1",
            "active_available": "1"
        }
    ],
    "id": 1
}
```

Retrieving host groups
Retrieve names of the groups host "Zabbix server" is member of, but no host details themselves.

Request:
```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectHostGroups": "extend",
        "filter": {"host": ["Zabbix server"]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095079e",
    "id": 2
}
```

Response:
```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10085",
            "groups": [
                {
                    "groupid": "2",
                    "name": "Linux servers",
                    "internal": "0",
                    "flags": "0"
                },
                {
                    "groupid": "4",
                    "name": "Zabbix servers",
                    "internal": "0",
                    "flags": "0"
                }
            ]
        },
        "id": 2
    }
```

Retrieving linked templates

Retrieve the IDs and names of templates linked to host "10084".

Request:
```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectParentTemplates": [
            "templateid",
            "name"
        ],
        "hostids": "10084"
    },
    "id": 1,
    "auth": "70785d2b494a7302309b48afcdb3a401"
}
```
Searching by host inventory data

Retrieve hosts that contain "Linux" in the host inventory "OS" field.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["host"],
        "selectInventory": [{"os"}, {"searchInventory": {"os": "Linux"}}],
        "id": 2,
        "auth": "7f9e00124c75e8f25facd5c093f3e9a0"
    }
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "host": "Zabbix server",
            "inventory": {
                "os": "Linux Ubuntu"
            }
        },
        {
            "hostid": "10107",
            "host": "Linux server",
            "inventory": {
                "os": "Linux Mint"
            }
        }
    ],
    "id": 1
}
```
Searching by host tags

Retrieve hosts that have tag "Host name" equal to "Linux server".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectTags": "extend",
        "evaltype": 0,
        "tags": [
            {
                "tag": "Host name",
                "value": "Linux server",
                "operator": 1
            }
        ],
        "inheritedTags": true
    },
    "auth": "7f9e00124c75e8f25facd5c093f3e9a0",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10085",
            "tags": [
                {
                    "tag": "Host name",
                    "value": "Linux server"
                },
                {
                    "tag": "OS",
                    "value": "RHEL 7"
                }
            ]
        }
    ],
    "id": 1
}
```

Retrieve hosts that have these tags not only on host level but also in their linked parent templates.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "tags": [
            {
                "tag": "A",
                "value": "1",
                "operator": 1
            }
        ],
        "inheritedTags": true
    }
}
```
Searching host with tags and template tags

Retrieve a host with tags and all tags that are linked to parent templates.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "hostids": [10502],
        "selectTags": ["tag", "value"],
        "selectInheritedTags": ["tag", "value"]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10502",
            "name": "Desktop",
            "tags": [
                {
                    "tag": "A",
                    "value": "1"
                }
            ],
            "inheritedTags": [
                {
                    "tag": "B",
                    "value": "2"
                }
            ]
        }
    ],
    "id": 1
}
```
Searching hosts by problem severity

Retrieve hosts that have "Disaster" problems.

Request:

```json
{  
  "jsonrpc": "2.0",
  "method": "host.get",
  "params": {
    "output": ["name"],
    "severities": 5
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```json
{  
  "jsonrpc": "2.0",
  "result": [
    
  ],
  "id": 1
}
```

Retrieve hosts that have "Average" and "High" problems.

Request:

```json
{  
  "jsonrpc": "2.0",
  "method": "host.get",
  "params": {
    "output": ["name"],
    "severities": [3, 4]
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```json
{  
  "jsonrpc": "2.0",
  "result": [
    
  ],
  "id": 1
}
```

See also

- Host group
- Template
- User macro
- Host interface
host.massadd

Description

object host.massadd(object parameters)

This method allows to simultaneously add multiple related objects to all the given hosts.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters containing the IDs of the hosts to update and the objects to add to all the hosts.

The method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hosts</td>
<td>object/array</td>
<td>Hosts to be updated. The hosts must have the hostid property defined.</td>
</tr>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Host groups to add to the given hosts.</td>
</tr>
<tr>
<td>interfaces</td>
<td>object/array</td>
<td>The host groups must have the groupid property defined.</td>
</tr>
<tr>
<td>macros</td>
<td>object/array</td>
<td>User macros to be created for the given hosts.</td>
</tr>
<tr>
<td>templates</td>
<td>object/array</td>
<td>Templates to link to the given hosts.</td>
</tr>
</tbody>
</table>

The templates must have the templateid property defined.

Return values

(object) Returns an object containing the IDs of the updated hosts under the hostids property.

Examples

Adding macros

Add two new macros to two hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.massadd",
    "params": {
        "hosts": [
            {
                "hostid": "10160"
            },
            {
                "hostid": "10167"
            }
        ],
        "macros": [
            {
                "macro": "{$TEST1}"
            },
            {
                "macro": "{$TEST2}"
            },
            {
                "macro": "MACROTEST1"
            },
            {
                "macro": "MACROTEST2"
            },
            {
                "description": "Test description"
            }
        ]
    }
}
```
Response:
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10160",
            "10167"
        ],
    },
    "id": 1
}

See also
- host.update
- Host group
- Template
- User macro
- Host interface

Source
CHost::massAdd() in ui/include/classes/api/services/CHost.php.

host.massremove

Description

object host.massremove(object parameters)

This method allows to remove related objects from multiple hosts.

This method is only available to Admin and Superadmin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters containing the IDs of the hosts to update and the objects that should be removed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>IDs of the hosts to be updated.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Host groups to remove the given hosts from.</td>
</tr>
<tr>
<td>interfaces</td>
<td>object/array</td>
<td>Host interfaces to remove from the given hosts.</td>
</tr>
<tr>
<td>macros</td>
<td>string/array</td>
<td>User macros to delete from the given hosts.</td>
</tr>
<tr>
<td>templateids</td>
<td>string/array</td>
<td>Templates to unlink from the given hosts.</td>
</tr>
<tr>
<td>templateids_clear</td>
<td>string/array</td>
<td>Templates to unlink and clear from the given hosts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated hosts under the hostids property.

Examples

Unlinking templates

Unlink a template from two hosts and delete all of the templated entities.
Request:
{
    "jsonrpc": "2.0",
    "method": "host.massremove",
    "params": {
        "hostids": ["69665", "69666"],
        "templateids_clear": "325"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}

Response:
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": ["69665", "69666"],
    },
    "id": 1
}

See also
- host.update
- Usermacro
- Hostinterface

Source
CHost::massRemove() in ui/include/classes/api/services/CHost.php.

host.massupdate

Description
object host.massupdate(object parameters)

This method allows to simultaneously replace or remove related objects and update properties on multiple hosts.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(object) Parameters containing the IDs of the hosts to update and the properties that should be updated.

Additionally to the standard host properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hosts</td>
<td>object/array</td>
<td>Hosts to be updated.</td>
</tr>
<tr>
<td>groups</td>
<td>object/array</td>
<td>The hosts must have the hostid property defined. Host groups to replace the current host groups the hosts belong to.</td>
</tr>
<tr>
<td>interfaces</td>
<td>object/array</td>
<td>The host groups must have the groupid property defined. Host interfaces to replace the current host interfaces on the given hosts.</td>
</tr>
<tr>
<td>inventory</td>
<td>object</td>
<td>Host inventory properties.</td>
</tr>
<tr>
<td>macros</td>
<td>object/array</td>
<td>Host inventory mode cannot be updated using the inventory parameter, use inventory_mode instead. User macros to replace the current user macros on the given hosts.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>templates</td>
<td>object/array</td>
<td>Templates to replace the currently linked templates on the given hosts.</td>
</tr>
<tr>
<td>templates_clear</td>
<td>object/array</td>
<td>The templates must have the templateid property defined.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated hosts under the hostids property.

Examples

Enabling multiple hosts

Enable monitoring of two hosts, i.e., set their status to 0.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.massupdate",
    "params": {
        "hosts": [
            {
                "hostid": "69665"
            },
            {
                "hostid": "69666"
            }
        ],
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "69665",
            "69666"
        ]
    },
    "id": 1
}
```

See also

- host.update
- host.massadd
- host.massremove
- Host group
- Template
- User macro
- Host interface

Source

CHost::massUpdate() in ui/include/classes/api/services/CHost.php.

**host.update**
Description

object host.update(object/array hosts)

This method allows to update existing hosts.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Host properties to be updated.

The hostid property must be defined for each host, all other properties are optional. Only the given properties will be updated, all others will remain unchanged.

Note, however, that updating the host technical name will also update the host’s visible name (if not given or empty) by the host’s technical name value.

Additionally to the standard host properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Host groups to replace the current host groups the host belongs to. The host groups must have the groupid property defined. All host groups that are not listed in the request will be unlinked.</td>
</tr>
<tr>
<td>interfaces</td>
<td>object/array</td>
<td>Host interfaces to replace the current host interfaces.</td>
</tr>
<tr>
<td>tags</td>
<td>object/array</td>
<td>All interfaces that are not listed in the request will be removed. Host tags to replace the current host tags.</td>
</tr>
<tr>
<td>inventory</td>
<td>object</td>
<td>All tags that are not listed in the request will be removed. Host inventory properties.</td>
</tr>
<tr>
<td>macros</td>
<td>object/array</td>
<td>User macros to replace the current user macros.</td>
</tr>
<tr>
<td>templates</td>
<td>object/array</td>
<td>Templates to replace the currently linked templates. All templates that are not listed in the request will be only unlinked.</td>
</tr>
<tr>
<td>templates_clear</td>
<td>object/array</td>
<td>The templates must have the templateid property defined. Templates to unlink and clear from the host. The templates must have the templateid property defined.</td>
</tr>
</tbody>
</table>

As opposed to the Zabbix frontend, when name (visible host name) is the same as host (technical host name), updating host via API will not automatically update name. Both properties need to be updated explicitly.

Return values

(object) Returns an object containing the IDs of the updated hosts under the hostids property.

Examples

Enabling a host

Enable host monitoring, i.e. set its status to 0.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10126",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```
Unlinking templates

Unlink and clear two templates from host.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "host.update",
  "params": {
    "hostid": "10126",
    "templates_clear": [
      {
        "templateid": "10124"
      },
      {
        "templateid": "10125"
      }
    ]
  },
  "auth": "038e1d7b1735c6a5436ee9ea0e0879e",
  "id": 1
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": {
    "hostids": ["10126"]
  },
  "id": 1
}
```

Updating host macros

Replace all host macros with two new ones.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "host.update",
  "params": {
    "hostid": "10126",
    "macros": [
      {
        "macro": "{$PASS}",
        "value": "password"
      },
      {
        "macro": "{$DISC}",
        "value": "sda",
        "description": "Updated description"
      }
    ]
  },
  "id": 1
}
```
Response:

```json
{
  "jsonrpc": "2.0",
  "result": {
    "hostids": [
      "10126"
    ],
  },
  "id": 1
}
```

Updating host inventory

**Change inventory mode and add location**

**Request:**

```json
{
  "jsonrpc": "2.0",
  "method": "host.update",
  "params": {
    "hostid": "10387",
    "inventory_mode": 0,
    "inventory": {
      "location": "Latvia, Riga"
    }
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": {
    "hostids": [
      "10387"
    ],
  },
  "id": 1
}
```

Updating host tags

**Replace all host tags with a new one.**

**Request:**

```json
{
  "jsonrpc": "2.0",
  "method": "host.update",
  "params": {
    "hostid": "10387",
    "tags": {
      "tag": "OS",
      "value": "CentOS 7"
    }
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```
Updating discovered host macros

Convert discovery rule created "automatic" macro to "manual" and change its value to "new-value".

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "host.update",
   "params": {
      "hostid": "10387",
      "macros": {
         "hostmacroid": "5541",
         "value": "new-value",
         "automatic": "0"
      }
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "hostids": [
         "10387"
      ]
   },
   "id": 1
}
```

See also

- host.massadd
- host.massupdate
- host.massremove
- Host group
- Template
- User macro
- Host interface
- Host inventory
- Host tag

Source

CHost::update() in ui/include/classes/api/services/CHost.php.

**Host group**

This class is designed to work with host groups.
Object references:

- Host group

Available methods:

- hostgroup.create - creating new host groups
- hostgroup.delete - deleting host groups
- hostgroup.get - retrieving host groups
- hostgroup.massadd - adding related objects to host groups
- hostgroup.massremove - removing related objects from host groups
- hostgroup.massupdate - replacing or removing related objects from host groups
- hostgroup.propagate - propagating permissions and tag filters to host groups' subgroups
- hostgroup.update - updating host groups

> Host group object

The following objects are directly related to the hostgroup API.

Host group

The host group object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupid</td>
<td>string</td>
<td>(readonly) ID of the host group.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the host group.</td>
</tr>
<tr>
<td>flags</td>
<td>integer</td>
<td>Origin of the host group. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - a plain host group;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - a discovered host group.</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Universal unique identifier, used for linking imported host groups to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>already existing ones. Auto-generated, if not given.</td>
</tr>
</tbody>
</table>

For update operations this field is readonly.

hostgroup.create

Description

object hostgroup.create(object/array hostGroups)

This method allows to create new host groups.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Host groups to create. The method accepts host groups with the standard host group properties.

Return values

(object) Returns an object containing the IDs of the created host groups under the groupIds property. The order of the returned IDs matches the order of the passed host groups.

Examples

Creating a host group

Create a host group called "Linux servers".

Request:
hostgroup.delete

Description

object hostgroup.delete(array hostGroupIds)

This method allows to delete host groups.

A host group can not be deleted if:

- it contains hosts that belong to this group only;
- it is marked as internal;
- it is used by a host prototype;
- it is used in a global script;
- it is used in a correlation condition.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the host groups to delete.

Return values

(object) Returns an object containing the IDs of the deleted host groups under the groupids property.

Examples

Deleting multiple host groups

Delete two host groups.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "hostgroup.delete",
    "params": [
        "107824",
        "107825"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```
Response:
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107824",
            "107825"
        ],
        "with_graphs": false
    },
    "id": 1
}

Source
CHostGroup::delete() in ui/include/classes/api/services/CHostGroup.php.

hostgroup.get

Description

integer/array hostgroup.get(object parameters)

The method allows to retrieve host groups according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphids</td>
<td>string/array</td>
<td>Return only host groups that contain hosts with the given graphs.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only host groups with the given host group IDs.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only host groups that contain the given host groups.</td>
</tr>
<tr>
<td>maintenanceids</td>
<td>string/array</td>
<td>Return only host groups that are affected by the given maintenances.</td>
</tr>
<tr>
<td>triggerids</td>
<td>string/array</td>
<td>Return only host groups that contain hosts with the given triggers.</td>
</tr>
<tr>
<td>with_graphs</td>
<td>flag</td>
<td>Return only host groups that contain hosts with graphs.</td>
</tr>
<tr>
<td>with_graph_prototypes</td>
<td>flag</td>
<td>Return only host groups that contain hosts with graph prototypes.</td>
</tr>
<tr>
<td>with_hosts</td>
<td>flag</td>
<td>Return only host groups that contain hosts.</td>
</tr>
<tr>
<td>with_httptests</td>
<td>flag</td>
<td>Return only host groups that contain hosts with web checks.</td>
</tr>
<tr>
<td>with_items</td>
<td>flag</td>
<td>Overrides the with_monitored_httptests parameter. Return only host groups that contain hosts with items.</td>
</tr>
<tr>
<td>with_item_prototypes</td>
<td>flag</td>
<td>Overrides the with_monitored_items and with_simple_graph_items parameters. Return only host groups that contain hosts with item prototypes.</td>
</tr>
<tr>
<td>with_simple_graph_item_prototypes</td>
<td>flag</td>
<td>Overrides the with_simple_graph_item_prototypes parameter. Return only host groups that contain hosts with item prototypes, which are enabled for creation and have numeric type of information.</td>
</tr>
<tr>
<td>with_monitored_httptests</td>
<td>flag</td>
<td>Return only host groups that contain hosts with enabled web checks.</td>
</tr>
<tr>
<td>with_monitored_hosts</td>
<td>flag</td>
<td>Return only host groups that contain monitored hosts.</td>
</tr>
<tr>
<td>with_monitored_items</td>
<td>flag</td>
<td>Return only host groups that contain hosts with enabled items.</td>
</tr>
<tr>
<td>with_monitored_triggers</td>
<td>flag</td>
<td>Return only host groups that contain hosts with enabled triggers. All of the items used in the trigger must also be enabled.</td>
</tr>
<tr>
<td>with_simple_graph_items</td>
<td>flag</td>
<td>Return only host groups that contain hosts with numeric items.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>with_triggers</td>
<td>flag</td>
<td>Return only host groups that contain hosts with triggers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overrides the with_monitored_triggers parameter.</td>
</tr>
<tr>
<td>selectDiscoveryRule</td>
<td>query</td>
<td>Return a discoveryRule property with the LLD rule that created the host group.</td>
</tr>
<tr>
<td>selectGroupDiscovery</td>
<td>query</td>
<td>Return a groupDiscovery property with the host group discovery object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The host group discovery object links a discovered host group to a host group prototype and has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- groupid - (string) ID of the discovered host group;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- lastcheck - (timestamp) time when the host group was last discovered;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- name - (string) name of the host group prototype;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- parent_group_prototypeid - (string) ID of the host group prototype from which the host group has been created;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ts_delete - (timestamp) time when a host group that is no longer discovered will be deleted.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return a hosts property with the hosts that belong to the host group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supports count.</td>
</tr>
<tr>
<td>limitSelects</td>
<td>integer</td>
<td>Limits the number of records returned by subselects.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Possible values are: groupid, name.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary page.</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>monitored_hosts</td>
<td>flag</td>
<td>This parameter is deprecated, please use with_monitored_hosts instead.</td>
</tr>
<tr>
<td>real_hosts</td>
<td>flag</td>
<td>This parameter is deprecated, please use with_hosts instead.</td>
</tr>
</tbody>
</table>

Return values
(integer/array) Returns either:
- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples
Retrieving data by name
Retrieve all data about two host groups named “Zabbix servers” and “Linux servers”.

Request:
```json
{
    "jsonrpc": "2.0",
```
"method": "hostgroup.get",
"params": {
    "output": "extend",
    "filter": {
        "name": [
            "Zabbix servers",
            "Linux servers"
        ]
    }
},
"auth": "6f38cddc44cfbb6c1bd186f9a220b5a0",
"id": 1
}

Response:

{
    "jsonrpc": "2.0",
    "result": [
        {
            "groupid": "2",
            "name": "Linux servers",
            "internal": "0"
        },
        {
            "groupid": "4",
            "name": "Zabbix servers",
            "internal": "0"
        }
    ],
    "id": 1
}

See also

- Host

Source

CHostGroup::get() in ui/include/classes/api/services/CHostGroup.php.

hostgroup.massadd

Description

object hostgroup.massadd(object parameters)

This method allows to simultaneously add multiple related objects to all the given host groups.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters containing the IDs of the host groups to update and the objects to add to all the host groups.

The method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Host groups to be updated.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>The host groups must have the groupid property defined.</td>
</tr>
<tr>
<td>hosts</td>
<td>object/array</td>
<td>Hosts to add to all host groups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The hosts must have the hostid property defined.</td>
</tr>
</tbody>
</table>

Return values
Returns an object containing the IDs of the updated host groups under the groupids property.

Examples

Adding hosts to host groups

Add two hosts to host groups with IDs 5 and 6.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.massadd",
    "params": {
        "groups": [
            {
                "groupid": "5"
            },
            {
                "groupid": "6"
            }
        ],
        "hosts": [
            {
                "hostid": "30050"
            },
            {
                "hostid": "30001"
            }
        ]
    },
    "auth": "f223adf833b2bf2ff38574a67bba6372",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "5",
            "6"
        ],
        "id": 1
    }
}
```

See also

- Host

Source

CHostGroup::massAdd() in ui/include/classes/api/services/CHostGroup.php.

**hostgroup.massremove**

**Description**

object hostgroup.massremove(object parameters)

This method allows to remove related objects from multiple host groups.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object) Parameters containing the IDs of the host groups to update and the objects that should be removed.
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>groupids</code></td>
<td>string/array</td>
<td>IDs of the host groups to be updated.</td>
</tr>
<tr>
<td><code>hostids</code></td>
<td>string/array</td>
<td>Hosts to remove from all host groups.</td>
</tr>
</tbody>
</table>

### Return values

*(object)* Returns an object containing the IDs of the updated host groups under the `groupids` property.

### Examples

**Removing hosts from host groups**

Remove two hosts from the given host groups.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "hostgroup.massremove",
    "params": {
        "groupids": ["5", "6"],
        "hostids": ["30050", "30001"]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": ["5", "6"],
    },
    "id": 1
}
```

### Source

`CHostGroup::massRemove()` in `ui/include/classes/api/services/CHostGroup.php`.

### `hostgroup.massupdate`

**Description**

`object hostgroup.massupdate(object parameters)`

This method allows to replace hosts and templates with the specified ones in multiple host groups.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

**Parameters**

*(object)* Parameters containing the IDs of the host groups to update and the objects that should be updated.
### Parameter Descriptions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Host groups to be updated. The host groups must have the <code>groupid</code> property defined.</td>
</tr>
<tr>
<td>hosts</td>
<td>object/array</td>
<td>Hosts to replace the current hosts on the given host groups. All other hosts, except the ones mentioned, will be excluded from host groups. Discovered hosts will not be affected. The hosts must have the <code>hostid</code> property defined.</td>
</tr>
</tbody>
</table>

### Return values

(object) Returns an object containing the IDs of the updated host groups under the `groupids` property.

### Examples

#### Replacing hosts in a host group

Replace all hosts in a host group to ones mentioned host.

**Request:**

```json
{
  "jsonrpc": "2.0",
  "method": "hostgroup.massupdate",
  "params": {
    "groups": [
      {
        "groupid": "6"
      }
    ],
    "hosts": [
      {
        "hostid": "30050"
      }
    ]
  },
  "auth": "f223adf833b2bf2ff38574a67bba6372",
  "id": 1
}
```

**Response:**

```json
{
  "jsonrpc": "2.0",
  "result": {
    "groupids": [
      "6",
    ]
  },
  "id": 1
}
```

### See also

- `hostgroup.update`
- `hostgroup.massadd`

### Source

CHostGroup::massUpdate() in `ui/include/classes/api/services/CHostGroup.php`.

### hostgroup.propagate

Description
object hostgroup.propagate(object parameters)

This method allows to apply permissions and tag filters to all hosts groups’ subgroups.

This method is only available to Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Host groups to propagate.</td>
</tr>
<tr>
<td></td>
<td>(required)</td>
<td>The host groups must have the groupid property defined.</td>
</tr>
<tr>
<td>permissions</td>
<td>boolean</td>
<td>Set true if need to propagate permissions.</td>
</tr>
<tr>
<td>tag_filters</td>
<td>boolean</td>
<td>Set true if need to propagate tag filters.</td>
</tr>
</tbody>
</table>

At least one parameter permissions or tag_filters is required.

Return values

(object) Returns an object containing the IDs of the propagated host groups under the groupids property.

Examples

Propagate host group permissions and tag filters to its subgroups.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "hostgroup.propagate",
   "params": {
      "groups": [
         {
            "groupid": "6"
         }
      ],
      "permissions": true,
      "tag_filters": true
   },
   "auth": "f223adf833b2bf2ff38574a67bba6372",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "groupids": [
         "6",
         ...
      ],
   },
   "id": 1
}
```

See also

- hostgroup.update
- hostgroup.massadd
- Host

Source

CHostGroup::propagate() in ui/include/classes/api/services/CHostGroup.php.
**hostgroup.update**

Description

object hostgroup.update(object/array hostGroups)

This method allows to update existing host groups.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Host group properties to be updated.

The groupid property must be defined for each host group, all other properties are optional. Only the given properties will be updated, all others will remain unchanged.

Return values

(object) Returns an object containing the IDs of the updated host groups under the groupids property.

Examples

Renaming a host group

Rename a host group to “Linux hosts.”

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "hostgroup.update",
   "params": {
      "groupid": "7",
      "name": "Linux hosts"
   },
   "auth": "700ca65537074963db7efabda78259",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "groupids": ["7"]
   },
   "id": 1
}
```

Source

CHostGroup::update() in ui/include/classes/api/services/CHostGroup.php.

**Host interface**

This functionality is deprecated and will be removed in upcoming versions.

This class is designed to work with host interfaces.

Object references:

- Host interface

Available methods:

- hostinterface.create - creating new host interfaces
- hostinterface.delete - deleting host interfaces
• hostinterface.get - retrieving host interfaces
• hostinterface.massadd - adding host interfaces to hosts
• hostinterface.massremove - removing host interfaces from hosts
• hostinterface.replacehostinterfaces - replacing host interfaces on a host
• hostinterface.update - updating host interfaces

> Host interface object

This functionality is deprecated and will be removed in upcoming versions.

The following objects are directly related to the hostinterface API.

Host interface

The host interface object has the following properties.

Note that both IP and DNS are required. If you do not want to use DNS, set it to an empty string.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>available</td>
<td>integer</td>
<td>(readonly) Availability of host interface.</td>
</tr>
<tr>
<td>details</td>
<td>array</td>
<td>Additional object for interface. Required if interface 'type' is SNMP.</td>
</tr>
<tr>
<td>disable_until</td>
<td>timestamp</td>
<td>(readonly) The next polling time of an unavailable host interface.</td>
</tr>
<tr>
<td>dns</td>
<td>string</td>
<td>DNS name used by the interface.</td>
</tr>
<tr>
<td>error</td>
<td>string</td>
<td>(readonly) Error text if host interface is unavailable.</td>
</tr>
<tr>
<td>errors_from</td>
<td>timestamp</td>
<td>(readonly) Time when host interface became unavailable.</td>
</tr>
<tr>
<td>hostid</td>
<td>string</td>
<td>ID of the host the interface belongs to.</td>
</tr>
<tr>
<td>interfaceid</td>
<td>string</td>
<td>(readonly) ID of the interface.</td>
</tr>
<tr>
<td>ip</td>
<td>string</td>
<td>IP address used by the interface.</td>
</tr>
<tr>
<td>main</td>
<td>integer</td>
<td>Whether the interface is used as default on the host. Only one interface of some type can be set as default on a host.</td>
</tr>
<tr>
<td>port</td>
<td>string</td>
<td>Port number used by the interface. Can contain user macros.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>Interface type.</td>
</tr>
<tr>
<td>useip</td>
<td>integer</td>
<td>Whether the connection should be made via IP.</td>
</tr>
</tbody>
</table>

Details tag

The details object has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>integer</td>
<td>SNMP interface version.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>Possible values are: 1 - SNMPv1; 2 - SNMPv2c; 3 - SNMPv3</td>
</tr>
<tr>
<td>bulk</td>
<td>integer</td>
<td>Whether to use bulk SNMP requests.</td>
</tr>
<tr>
<td>community</td>
<td>string</td>
<td>SNMP community (required). Used only by SNMPv1 and SNMPv2 interfaces.</td>
</tr>
<tr>
<td>securityname</td>
<td>string</td>
<td>SNMPv3 security name. Used only by SNMPv3 interfaces.</td>
</tr>
<tr>
<td>securitylevel</td>
<td>integer</td>
<td>SNMPv3 security level. Used only by SNMPv3 interfaces.</td>
</tr>
<tr>
<td>authpassphrase</td>
<td>string</td>
<td>SNMPv3 authentication passphrase. Used only by SNMPv3 interfaces.</td>
</tr>
<tr>
<td>privpassphrase</td>
<td>string</td>
<td>SNMPv3 privacy passphrase. Used only by SNMPv3 interfaces.</td>
</tr>
<tr>
<td>authprotocol</td>
<td>integer</td>
<td>SNMPv3 authentication protocol. Used only by SNMPv3 interfaces.</td>
</tr>
<tr>
<td>privprotocol</td>
<td>integer</td>
<td>SNMPv3 privacy protocol. Used only by SNMPv3 interfaces.</td>
</tr>
<tr>
<td>contextname</td>
<td>string</td>
<td>SNMPv3 context name. Used only by SNMPv3 interfaces.</td>
</tr>
</tbody>
</table>

### hostinterface.create

This functionality is deprecated and will be removed in upcoming versions.

**Description**

object hostinterface.create(object/array hostInterfaces)

This method allows to create new host interfaces.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) Host interfaces to create. The method accepts host interfaces with the standard host interface properties.

**Return values**

(object) Returns an object containing the IDs of the created host interfaces under the interfaceids property. The order of the returned IDs matches the order of the passed host interfaces.

**Examples**
Create a new interface

Create a secondary IP agent interface on host “30052.”

Request:

```
{  
  "jsonrpc": "2.0",
  "method": "hostinterface.create",
  "params": {
    "hostid": "30052",
    "main": "0",
    "type": "1",
    "useip": "1",
    "ip": "127.0.0.1",
    "dns": "",
    "port": "10050",
  },  
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{  
  "jsonrpc": "2.0",
  "result": {
    "interfaceids": [  
      "30062"
    ]
  },  
  "id": 1
}
```

Create an interface with SNMP details

Request:

```
{  
  "jsonrpc": "2.0",
  "method": "hostinterface.create",
  "params": {
    "hostid": "10456",
    "main": "0",
    "type": "2",
    "useip": "1",
    "ip": "127.0.0.1",
    "dns": "",
    "port": "1601",
    "details": {
      "version": "2",
      "bulk": "1",
      "community": "{$SNMP_COMMUNITY}"  
    },  
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{  
  "jsonrpc": "2.0",
  "result": {
    "interfaceids": [  
      "30063"
    ]
  }
}
```
hostinterface.delete

This functionality is deprecated and will be removed in upcoming versions.

Description

object hostinterface.delete(array hostInterfaceIds)

This method allows to delete host interfaces.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the host interfaces to delete.

Return values

(object) Returns an object containing the IDs of the deleted host interfaces under the interfaceids property.

Examples

Delete a host interface

Delete the host interface with ID 30062.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.delete",
    "params": [
        "30062"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30062"
        ]
    },
    "id": 1
}
```

See also

• hostinterface.massremove
• host.massremove

Source

CHostInterface::delete() in ui/include/classes/api/services/CHostInterface.php.
**hostinterface.get**

This functionality is deprecated and will be removed in upcoming versions.

**Description**

`integer/array hostinterface.get(object parameters)`

The method allows to retrieve host interfaces according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only host interfaces used by the given hosts.</td>
</tr>
<tr>
<td>interfaceIds</td>
<td>string/array</td>
<td>Return only host interfaces with the given IDs.</td>
</tr>
<tr>
<td>itemIds</td>
<td>string/array</td>
<td>Return only host interfaces used by the given items.</td>
</tr>
<tr>
<td>triggerIds</td>
<td>string/array</td>
<td>Return only host interfaces used by items in the given triggers.</td>
</tr>
<tr>
<td>selectItems</td>
<td>query</td>
<td>Return an items property with the items that use the interface.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return a hosts property with an array of hosts that use the interface.</td>
</tr>
<tr>
<td>limitSelects</td>
<td>integer</td>
<td>Limits the number of records returned by subselects.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary page.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>nodeIds</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Return values**

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

**Examples**

Retrieve host interfaces

Retrieve all data about the interfaces used by host “30057.”

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.get",
```
"params": {
    "output": "extend",
    "hostids": "30057"
},
"auth": "038e1d7b1735c6a5a436ee9eae095879e",
"id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": [
    {
      "interfaceid": "50039",
      "hostid": "30057",
      "main": "1",
      "type": "1",
      "useip": "1",
      "ip": "::1",
      "dns": "",
      "port": "10050",
      "available": "0",
      "error": "",
      "errors_from": "0",
      "disable_until": "0",
      "details": []
    },
    {
      "interfaceid": "55082",
      "hostid": "30057",
      "main": "0",
      "type": "1",
      "useip": "1",
      "ip": "127.0.0.1",
      "dns": "",
      "port": "10051",
      "available": "0",
      "error": "",
      "errors_from": "0",
      "disable_until": "0",
      "details": {
        "version": "2",
        "bulk": "0",
        "community": "{$SNMP_COMMUNITY}"
      }
    }
  ],
  "id": 1
}

See also
• Host
• Item

Source
CHostInterface::get() in ui/include/classes/api/services/CHostInterface.php.

hostinterface.massadd

This functionality is deprecated and will be removed in upcoming versions.

Description
object hostinterface.massadd(object parameters)

This method allows to simultaneously add host interfaces to multiple hosts.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters containing the host interfaces to be created on the given hosts.

The method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hosts</td>
<td>object/array</td>
<td>Hosts to be updated. The hosts must have the hostid property defined.</td>
</tr>
<tr>
<td>interfaces</td>
<td>object/array</td>
<td>Host interfaces to create on the given hosts.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the created host interfaces under the interfaceids property.

Examples

Creating interfaces

Create an interface on two hosts.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "hostinterface.massadd",
    "params": {
        "hosts": [
            {
                "hostid": "30050"
            },
            {
                "hostid": "30052"
            }
        ],
        "interfaces": {
            "dns": ",",
            "ip": "127.0.0.1",
            "main": 0,
            "port": "10050",
            "type": 1,
            "useip": 1
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30069",
            "30070"
        ]
    },
    "id": 1
}
```
See also

- `hostinterface.create`
- `host.massadd`
- `Host`

**Source**

`CHostInterface::massAdd()` in `ui/include/classes/api/services/CHostInterface.php`.

### `hostinterface.massremove`

This functionality is deprecated and will be removed in upcoming versions.

**Description**

```object hostinterface.massremove(object parameters)```

This method allows to remove host interfaces from the given hosts.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See **User roles** for more information.

**Parameters**

(object) Parameters containing the IDs of the hosts to be updated and the interfaces to be removed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>hostids</strong> (required)</td>
<td>string/array</td>
<td>IDs of the hosts to be updated.</td>
</tr>
<tr>
<td><strong>interfaces</strong> (required)</td>
<td>object/array</td>
<td>Host interfaces to remove from the given hosts. The host interface object must have the ip, dns and port properties defined</td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns an object containing the IDs of the deleted host interfaces under the `interfaceids` property.

**Examples**

**Removing interfaces**

Remove the “127.0.0.1” SNMP interface from two hosts.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "hostinterface.massremove",
    "params": {
        "hostids": [
            "30050",
            "30052"
        ],
        "interfaces": {
            "dns": "",
            "ip": "127.0.0.1",
            "port": "161"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

**Response:**

```json
"id": 1
}
```
hostinterface.replacehostinterfaces

This functionality is deprecated and will be removed in upcoming versions.

Description

object hostinterface.replacehostinterfaces(object parameters)

This method allows to replace all host interfaces on a given host.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters containing the ID of the host to be updated and the new host interfaces.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostid</td>
<td>string</td>
<td>ID of the host to be updated.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>interfaces</td>
<td>object/array</td>
<td>Host interfaces to replace the current host interfaces with.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the created host interfaces under the interfaceids property.

Examples

Replacing host interfaces

Replace all host interfaces with a single agent interface.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "hostinterface.replacehostinterfaces",
    "params": {
        "hostid": "30052",
        "interfaces": {
            "dns": "",
            "ip": "127.0.0.1",
            "main": 1,
            "port": "10050",
            "type": 1,
            "useip": 1
        }
    }
}
```
See also

- host.update
- host.massupdate

Source

CHostInterface::replaceHostInterfaces() in ui/include/classes/api/services/CHostInterface.php.

**hostinterface.update**

This functionality is deprecated and will be removed in upcoming versions.

**Description**

do ubject hostinterface.update(object/array hostInterfaces)

This method allows to update existing host interfaces.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) **Host interface properties** to be updated.

The interfaceid property must be defined for each host interface, all other properties are optional. Only the given properties will be updated, all others will remain unchanged.

**Return values**

(object) Returns an object containing the IDs of the updated host interfaces under the interfaceids property.

**Examples**

Changing a host interface port

Change the port of a host interface.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "hostinterface.update",
    "params": {
        "interfaceid": "30048",
        "port": "30050"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

**Response:**
Host prototype

This class is designed to work with host prototypes.

Object references:

- Host prototype
- Host prototype inventory
- Group link
- Group prototype

Available methods:

- `hostprototype.create` - creating new host prototypes
- `hostprototype.delete` - deleting host prototypes
- `hostprototype.get` - retrieving host prototypes
- `hostprototype.update` - updating host prototypes

> Host prototype object

The following objects are directly related to the hostprototype API.

Host prototype

The host prototype object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostid</td>
<td>string</td>
<td>(readonly) ID of the host prototype.</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>Technical name of the host prototype.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Visible name of the host prototype.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Status of the host prototype.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) monitored host;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - unmonitored host.</td>
</tr>
<tr>
<td>inventory_mode</td>
<td>integer</td>
<td>Host inventory population mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1 - (default) disabled;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0  - manual;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1  - automatic.</td>
</tr>
<tr>
<td>templateid</td>
<td>string</td>
<td>(readonly) ID of the parent template host prototype.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>discover</td>
<td>integer</td>
<td>Host prototype discovery status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) new hosts will be discovered;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - new hosts will not be discovered and existing hosts will be marked</td>
</tr>
<tr>
<td></td>
<td></td>
<td>as lost.</td>
</tr>
<tr>
<td>custom_interfaces</td>
<td>integer</td>
<td>Source of interfaces for hosts created by the host prototype.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) inherit interfaces from parent host;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - use host prototypes custom interfaces.</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Universal unique identifier, used for linking imported host prototypes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to already existing ones. Used only for host prototypes on templates.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Auto-generated, if not given.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For update operations this field is readonly.</td>
</tr>
</tbody>
</table>

Group link

The group link object links a host prototype with a host group and has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group_prototypeid</td>
<td>string</td>
<td>(readonly) ID of the group link.</td>
</tr>
<tr>
<td>groupid</td>
<td>string</td>
<td>ID of the host group.</td>
</tr>
<tr>
<td>hostid</td>
<td>string</td>
<td>(readonly) ID of the host prototype</td>
</tr>
<tr>
<td>templateid</td>
<td>string</td>
<td>(readonly) ID of the parent template group link.</td>
</tr>
</tbody>
</table>

Group prototype

The group prototype object defines a group that will be created for a discovered host and has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group_prototypeid</td>
<td>string</td>
<td>(readonly) ID of the group prototype.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the group prototype.</td>
</tr>
<tr>
<td>hostid</td>
<td>string</td>
<td>(readonly) ID of the host prototype</td>
</tr>
<tr>
<td>templateid</td>
<td>string</td>
<td>(readonly) ID of the parent template group prototype.</td>
</tr>
</tbody>
</table>

Host prototype tag

The host prototype tag object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Host prototype tag name.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Host prototype tag value.</td>
</tr>
</tbody>
</table>

Custom interface

The custom interface object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dns</td>
<td>string</td>
<td>DNS name used by the interface.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required if the connection is made via DNS. Can contain macros.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>ip</td>
<td>string</td>
<td>IP address used by the interface.</td>
</tr>
<tr>
<td>main</td>
<td>integer</td>
<td>Whether the interface is used as default on the host. Only one interface of some type can be set as default on a host. Possible values are: 0 - not default; 1 - default.</td>
</tr>
<tr>
<td>port</td>
<td>string</td>
<td>Port number used by the interface. Can contain user and LLD macros.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>Interface type. Possible values are: 1 - agent; 2 - SNMP; 3 - IPMI; 4 - JMX.</td>
</tr>
<tr>
<td>useip</td>
<td>integer</td>
<td>Whether the connection should be made via IP. Possible values are: 0 - connect using host DNS name; 1 - connect using host IP address for this host interface.</td>
</tr>
<tr>
<td>details</td>
<td>array</td>
<td>Additional object for interface. Required if interface ‘type’ is SNMP.</td>
</tr>
</tbody>
</table>

Custom interface details

The details object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>integer</td>
<td>SNMP interface version. Possible values are: 1 - SNMPv1; 2 - SNMPv2c; 3 - SNMPv3</td>
</tr>
<tr>
<td>bulk</td>
<td>integer</td>
<td>Whether to use bulk SNMP requests. Possible values are: 0 - don’t use bulk requests; 1 - (default) - use bulk requests.</td>
</tr>
<tr>
<td>community</td>
<td>string</td>
<td>SNMP community. Used only by SNMPv1 and SNMPv2 interfaces.</td>
</tr>
<tr>
<td>securityname</td>
<td>string</td>
<td>SNMPv3 security name. Used only by SNMPv3 interfaces.</td>
</tr>
<tr>
<td>securitylevel</td>
<td>integer</td>
<td>SNMPv3 security level. Used only by SNMPv3 interfaces. Possible values are: 0 - (default) - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.</td>
</tr>
<tr>
<td>authpassphrase</td>
<td>string</td>
<td>SNMPv3 authentication passphrase. Used only by SNMPv3 interfaces.</td>
</tr>
<tr>
<td>privpassphrase</td>
<td>string</td>
<td>SNMPv3 privacy passphrase. Used only by SNMPv3 interfaces.</td>
</tr>
</tbody>
</table>
### hostprototype.create

**Description**

object hostprototype.create(object/array hostPrototypes)

This method allows to create new host prototypes.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) Host prototypes to create.

Additionally to the standard host prototype properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupLinks</td>
<td>array</td>
<td>Group links to be created for the host prototype.</td>
</tr>
<tr>
<td>ruleid</td>
<td>string</td>
<td>ID of the LLD rule that the host prototype belongs to.</td>
</tr>
<tr>
<td>groupPrototypes</td>
<td>array</td>
<td>Group prototypes to be created for the host prototype.</td>
</tr>
<tr>
<td>macros</td>
<td>object/array</td>
<td>User macros to be created for the host prototype.</td>
</tr>
<tr>
<td>tags</td>
<td>object/array</td>
<td>Host prototype tags.</td>
</tr>
<tr>
<td>interfaces</td>
<td>object/array</td>
<td>Host prototype custom interfaces.</td>
</tr>
<tr>
<td>templates</td>
<td>object/array</td>
<td>Templates to be linked to the host prototype.</td>
</tr>
</tbody>
</table>

The templates must have the templateid property defined.

**Return values**

(object) Returns an object containing the IDs of the created host prototypes under the hostIds property. The order of the returned IDs matches the order of the passed host prototypes.

**Examples**

**Creating a host prototype**

Create a host prototype "(#VM.NAME)" on LLD rule "23542" with a group prototype "(#HV.NAME)", tag pair "Datacenter": "(#DATACENTER.NAME)" and custom SNMPv2 interface 127.0.0.1:161 with community {$SNMP_COMMUNITY}. Link it to host group "2".

**Request:**

```plaintext
1011
```
See also

- Group link
- Group prototype
- Host prototype tag
- Custom interface
- User macro

Source
hostprototype.delete

Description

object hostprototype.delete(array hostPrototypeIds)

This method allows to delete host prototypes.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the host prototypes to delete.

Return values

(object) Returns an object containing the IDs of the deleted host prototypes under the hostids property.

Examples

Deleting multiple host prototypes

Delete two host prototypes.

Request:

```
{
   "jsonrpc": "2.0",
   "method": "hostprototype.delete",
   "params": [
      "10103",
      "10105"
   ],
   "auth": "3a57200802b24cda67c4e4010b50c065",
   "id": 1
}
```

Response:

```
{
   "jsonrpc": "2.0",
   "result": {
      "hostids": [
         "10103",
         "10105"
      ],
      "id": 1
   }
}
```

Source

CHostPrototype::create() in ui/include/classes/api/services/CHostPrototype.php.

hostprototype.get

Description

integer/array hostprototype.get(object parameters)

The method allows to retrieve host prototypes according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.
The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only host prototypes with the given IDs.</td>
</tr>
<tr>
<td>discoveryids</td>
<td>string/array</td>
<td>Return only host prototype that belong to the given LLD rules.</td>
</tr>
<tr>
<td>inherited</td>
<td>boolean</td>
<td>If set to true return only items inherited from a template.</td>
</tr>
<tr>
<td>selectDiscoveryRule</td>
<td>query</td>
<td>Return a discoveryRule property with the LLD rule that the host prototype belongs to.</td>
</tr>
<tr>
<td>selectInterfaces</td>
<td>query</td>
<td>Return an interfaces property with host prototype custom interfaces.</td>
</tr>
<tr>
<td>selectGroupLinks</td>
<td>query</td>
<td>Return a groupLinks property with the group links of the host prototype.</td>
</tr>
<tr>
<td>selectGroupPrototypes</td>
<td>query</td>
<td>Return a groupPrototypes property with the group prototypes of the host prototype.</td>
</tr>
<tr>
<td>selectMacros</td>
<td>query</td>
<td>Return a macros property with host prototype macros.</td>
</tr>
<tr>
<td>selectParentHost</td>
<td>query</td>
<td>Return a parentHost property with the host that the host prototype belongs to.</td>
</tr>
<tr>
<td>selectTags</td>
<td>query</td>
<td>Return a tags property with host prototype tags.</td>
</tr>
<tr>
<td>selectTemplates</td>
<td>query</td>
<td>Return a templates property with the templates linked to the host prototype.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail on the Generic Zabbix API Information page.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td>Supports count.</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td>Possible values are: hostid, host, name and status.</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values
(integer/array) Returns either:
- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving host prototypes from an LLD rule

Retrieve all host prototypes, their group links, group prototypes and tags from an LLD rule.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "hostprototype.get",
    "params": {
        "output": "extend",
        "selectInterfaces": "extend",
        "selectGroupLinks": "extend",
        "selectGroupPrototypes": "extend",
        "selectTags": "extend",
        "discoveryids": "23554"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e"
}
```
Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "hostid": "10092",
      "host": "{#HV.UUID}",
      "name": "{#HV.UUID}",
      "status": "0",
      "templateid": "0",
      "discover": "0",
      "custom_interfaces": "1",
      "inventory_mode": "-1",
      "groupLinks": [
        {
          "group_prototypeid": "4",
          "hostid": "10092",
          "groupid": "7",
          "templateid": "0"
        }
      ],
      "groupPrototypes": [
        {
          "group_prototypeid": "7",
          "hostid": "10092",
          "name": "{#CLUSTER.NAME}",
          "templateid": "0"
        }
      ],
      "tags": [
        {
          "tag": "Datacenter",
          "value": "{#DATACENTER.NAME}"
        },
        {
          "tag": "Instance type",
          "value": "{#INSTANCE_TYPE}"}
      ],
      "interfaces": [
        {
          "main": "1",
          "type": "2",
          "useip": "1",
          "ip": "127.0.0.1",
          "dns": "",
          "port": "161",
          "details": {
            "version": "2",
            "bulk": "1",
            "community": "{SNMP_COMMUNITY}"}
        }
      ]
    }
  ]
}
```
• Group link
• Group prototype
• User macro

Source

CHostPrototype::get() in ui/include/classes/api/services/CHostPrototype.php.

**hostprototype.update**

Description

`object hostprototype.update(object/array hostPrototypes)`

This method allows to update existing host prototypes.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See *User roles* for more information.

Parameters

(object/array) Host prototype properties to be updated.

The `hostid` property must be defined for each host prototype, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the **standard host prototype properties**, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupLinks</td>
<td>array</td>
<td>Group links to replace the current group links on the host prototype.</td>
</tr>
<tr>
<td>groupPrototypes</td>
<td>array</td>
<td>Group prototypes to replace the existing group prototypes on the host prototype.</td>
</tr>
<tr>
<td>macros</td>
<td>object/array</td>
<td>User macros to replace the current user macros.</td>
</tr>
<tr>
<td>tags</td>
<td>object/array</td>
<td>All macros that are not listed in the request will be removed. Host prototype tags to replace the current tags.</td>
</tr>
<tr>
<td>interfaces</td>
<td>object/array</td>
<td>Host prototype custom interfaces to replace the current interfaces.</td>
</tr>
<tr>
<td>templates</td>
<td>object/array</td>
<td>Custom interface object should contain all its parameters. All interfaces that are not listed in the request will be removed. Templates to replace the currently linked templates.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated host prototypes under the `hostids` property.

Examples

Disabling a host prototype

Disable a host prototype, that is, set its status to 1.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "hostprototype.update",
    "params": {
        "hostid": "10092",
        "status": 1
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```
Response:

```
{
   "jsonrpc": "2.0",
   "result": {
      "hostids": [
         "10092"
      ],
   },
   "id": 1
}
```

Updating host prototype tags

Replace host prototype tags with new ones.

Request:

```
{
   "jsonrpc": "2.0",
   "method": "hostprototype.update",
   "params": {
      "hostid": "10092",
      "tags": [
         {
            "tag": "Datacenter",
            "value": "#{DATACENTER.NAME}"
         },
         {
            "tag": "Instance type",
            "value": "#{INSTANCE_TYPE}"
         }
      ]
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```

Response:

```
{
   "jsonrpc": "2.0",
   "result": {
      "hostids": [
         "10092"
      ],
   },
   "id": 1
}
```

Updating host prototype custom interfaces

Replace inherited interfaces with host prototype custom interfaces.

Request:

```
{
   "jsonrpc": "2.0",
   "method": "hostprototype.update",
   "params": {
      "hostid": "10092",
      "custom_interfaces": "1",
      "interfaces": [
         {
            "main": "1",
            "type": "2",
            "useip": "1",
            "ip": "127.0.0.1",
```
"dns": "",
"port": "161",
"details": {
   "version": "2",
   "bulk": "1",
   "community": "{$SNMP_COMMUNITY}"
}
}
,"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
"jsonrpc": "2.0",
"result": {
   "hostids": [10092]
},
"id": 1
}

See also
- Group link
- Group prototype
- Host prototype tag
- Custom interface
- User macro

Source
CHostPrototype::update() in ui/include/classes/api/services/CHostPrototype.php.

Housekeeping

This class is designed to work with housekeeping.

Object references:

- Housekeeping

Available methods:

- housekeeping.get - retrieve housekeeping
- housekeeping.update - update housekeeping

Housekeeping object

The following objects are directly related to the housekeeping API.

Housekeeping

The settings object has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hk_events_mode</td>
<td>integer</td>
<td>Enable internal housekeeping for events and alerts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Disable;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) Enable.</td>
</tr>
<tr>
<td>hk_events_trigger</td>
<td>string</td>
<td>Trigger data storage period. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 365d.</td>
</tr>
<tr>
<td>hk_events_service</td>
<td>string</td>
<td>Service data storage period. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1d.</td>
</tr>
<tr>
<td>hk_events_internal</td>
<td>string</td>
<td>Internal data storage period. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1d.</td>
</tr>
<tr>
<td>hk_events_discovery</td>
<td>string</td>
<td>Network discovery data storage period. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1d.</td>
</tr>
<tr>
<td>hk_events_autoreg</td>
<td>string</td>
<td>Autoregistration data storage period. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1d.</td>
</tr>
<tr>
<td>hk_services_mode</td>
<td>integer</td>
<td>Enable internal housekeeping for services.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Disable;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) Enable.</td>
</tr>
<tr>
<td>hk_services</td>
<td>string</td>
<td>Services data storage period. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 365d.</td>
</tr>
<tr>
<td>hk_audit_mode</td>
<td>integer</td>
<td>Enable internal housekeeping for audit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Disable;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) Enable.</td>
</tr>
<tr>
<td>hk_audit</td>
<td>string</td>
<td>Audit data storage period. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 365d.</td>
</tr>
<tr>
<td>hk_sessions_mode</td>
<td>integer</td>
<td>Enable internal housekeeping for sessions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Disable;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) Enable.</td>
</tr>
<tr>
<td>hk_sessions</td>
<td>string</td>
<td>Sessions data storage period. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 365d.</td>
</tr>
<tr>
<td>hk_history_mode</td>
<td>integer</td>
<td>Enable internal housekeeping for history.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Disable;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) Enable.</td>
</tr>
<tr>
<td>hk_history_global</td>
<td>integer</td>
<td>Override item history period.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Do not override;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) Override.</td>
</tr>
<tr>
<td>hk_history</td>
<td>string</td>
<td>History data storage period. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 90d.</td>
</tr>
</tbody>
</table>
### housekeeping.get

**Description**

object housekeeping.get(object parameters)

The method allows to retrieve housekeeping object according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object) Parameters defining the desired output.

The method supports only one parameter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>query</td>
<td>This parameter being common for all get methods described in the reference commentary.</td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns housekeeping object.

**Examples**

**Request:**

```json
{
  "jsonrpc": "2.0",
  "method": "housekeeping.get",
  "params": {}
}
```
"output": "extend"
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": {
    "hk_events_mode": "1",
    "hk_events_trigger": "365d",
    "hk_events_service": "1d",
    "hk_events_internal": "id",
    "hk_events_discovery": "1d",
    "hk_events_autoreg": "1d",
    "hk_services_mode": "1",
    "hk_services": "365d",
    "hk_audit_mode": "1",
    "hk_audit": "365d",
    "hk_sessions_mode": "1",
    "hk_sessions": "365d",
    "hk_history_mode": "1",
    "hk_history": "90d",
    "hk_trends_mode": "1",
    "hk_trends": "365d",
    "db_extension": "",
    "compression_status": "0",
    "compress_older": "7d"
  },
  "id": 1
}

Source
CHousekeeping ::get() in ui/include/classes/api/services/CHousekeeping.php.

**housekeeping.update**

**Description**

object housekeeping.update(object housekeeping)

This method allows to update existing housekeeping settings.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

**Parameters**

(object) Housekeeping properties to be updated.

**Return values**

(array) Returns array with the names of updated parameters.

**Examples**

**Request:**

{
  "jsonrpc": "2.0",
  "method": "housekeeping.update",
  "params": {
    "hk_events_mode": "1",
    "hk_events_trigger": "200d",
  }
}
"hk_events_internal": "2d",
"hk_events_discovery": "2d"
},
"auth": "038e1d7b1735c6a5436ee9eae096879e",
"id": 1
}

Response:
{
"jsonrpc": "2.0",
"result": [
"hk_events_mode",
"hk_events_trigger",
"hk_events_internal",
"hk_events_discovery"
],
"id": 1
}

Source
CHousekeeping::update() in ui/include/classes/api/services/CHousekeeping.php.

Icon map

This class is designed to work with icon maps.

Object references:

- Icon map
- Icon mapping

Available methods:

- iconmap.create - create new icon maps
- iconmap.delete - delete icon maps
- iconmap.get - retrieve icon maps
- iconmap.update - update icon maps

> Icon map object

The following objects are directly related to the iconmap API.

Icon map

The icon map object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iconmapid</td>
<td>string</td>
<td>(readonly) ID of the icon map.</td>
</tr>
<tr>
<td>default_iconid</td>
<td>string</td>
<td>ID of the default icon.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the icon map.</td>
</tr>
</tbody>
</table>

Icon mapping

The icon mapping object defines a specific icon to be used for hosts with a certain inventory field value. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iconmappingid</td>
<td>string</td>
<td>(readonly) ID of the icon map.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>iconid</td>
<td>string</td>
<td>ID of the icon used by the icon mapping.</td>
</tr>
<tr>
<td>expression</td>
<td>string</td>
<td>Expression to match the inventory field against.</td>
</tr>
<tr>
<td>inventory_link</td>
<td>integer</td>
<td>ID of the host inventory field.</td>
</tr>
<tr>
<td>iconmapid</td>
<td>string</td>
<td>(readonly) ID of the icon map that the icon mapping belongs to.</td>
</tr>
<tr>
<td>sortorder</td>
<td>integer</td>
<td>(readonly) Position of the icon mapping in the icon map.</td>
</tr>
</tbody>
</table>

**iconmap.create**

Description

object iconmap.create(object/array iconMaps)

This method allows to create new icon maps.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

Parameters

(object/array) Icon maps to create.

Additionally to the [standard icon map properties](#), the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mappings</td>
<td>array</td>
<td>Icon mappings to be created for the icon map.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the created icon maps under the iconmapids property. The order of the returned IDs matches the order of the passed icon maps.

Examples

Create an icon map

Create an icon map to display hosts of different types.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "iconmap.create",
    "params": {
        "name": "Type icons",
        "default_iconid": 2,
        "mappings": [
            {
                "inventory_link": 1,
                "expression": "server",
                "iconid": 3
            },
            {
                "inventory_link": 1,
                "expression": "switch",
                "iconid": 4
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436e9eae095879e",
```
See also
• icon mapping

Source
CIconMap::create() in ui/include/classes/api/services/CIconMap.php.

**iconmap.delete**

Description

object iconmap.delete(array iconMapIds)

This method allows to delete icon maps.

This method is only available to Superadmin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the icon maps to delete.

Return values

(object) Returns an object containing the IDs of the deleted icon maps under the iconmapids property.

Examples

Delete multiple icon maps

Delete two icon maps.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "iconmap.delete",
    "params": [
        "2",
        "5"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "iconmapids": [
            "2",
            "5"
        ]
    },
    "id": 1
}
```
**iconmap.get**

**Description**

`integer/array iconmap.get(object parameters)`

The method allows to retrieve icon maps according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iconmapids</td>
<td>string/array</td>
<td>Return only icon maps with the given IDs.</td>
</tr>
<tr>
<td>sysmapids</td>
<td>string/array</td>
<td>Return only icon maps that are used in the given maps.</td>
</tr>
<tr>
<td>selectMappings</td>
<td>query</td>
<td>Return a mappings property with the icon mappings used.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Return values**

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

**Examples**

Retrieve an icon map

Retrieve all data about icon map "3".

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "iconmap.get",
    "params": {
        "iconmapids": "3",
        "output": "extend",
        "selectMappings": "extend"
    }
}
```
Response:
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mappings": [
                {
                    "iconmappingid": "3",
                    "iconmapid": "3",
                    "iconid": "6",
                    "inventory_link": "1",
                    "expression": "server",
                    "sortorder": "0"
                },
                {
                    "iconmappingid": "4",
                    "iconmapid": "3",
                    "iconid": "10",
                    "inventory_link": "1",
                    "expression": "switch",
                    "sortorder": "1"
                }
            ],
            "iconmapid": "3",
            "name": "Host type icons",
            "default_iconid": "2"
        },
        "id": 1
    ]
}

See also

- Icon mapping

Source
CIconMap::get() in ui/include/classes/api/services/CIconMap.php.

iconmap.update

Description

docter iconmap.update(object/array iconMaps)

This method allows to update existing icon maps.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Icon map properties to be updated.

The iconmapid property must be defined for each icon map, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard icon map properties, the method accepts the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mappings</td>
<td>array</td>
<td>Icon mappings to replace the existing icon mappings.</td>
</tr>
</tbody>
</table>
Return values

(object) Returns an object containing the IDs of the updated icon maps under the iconmapids property.

Examples

Rename icon map

Rename an icon map to “OS icons”.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "iconmap.update",
    "params": {
        "iconmapid": "1",
        "name": "OS icons"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "iconmapids": [
            "1"
        ],
    },
    "id": 1
}
```

See also

- Icon mapping

Source

CIconMap::update() in ui/include/classes/api/services/CIconMap.php.

Image

This class is designed to work with images.

Object references:

- Image

Available methods:

- image.create - create new images
- image.delete - delete images
- image.get - retrieve images
- image.update - update images

> Image object

The following objects are directly related to the image API.

Image

The image object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imageid</td>
<td>string</td>
<td>(readonly) ID of the image.</td>
</tr>
</tbody>
</table>
### image.create

**Description**

`object image.create(object/array images)`

This method allows to create new images. This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

**Parameters**

`(object/array)` Images to create.

Additionally to the standard image properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name</strong></td>
<td>string</td>
<td>Name of the image.</td>
</tr>
<tr>
<td><strong>imagetype</strong></td>
<td>integer</td>
<td>Type of image.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) icon;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - background image.</td>
</tr>
</tbody>
</table>

| **image** | string | Base64 encoded image. The maximum size of the encoded image is 1 MB. Maximum size can be adjusted by changing `ZBX_MAX_IMAGE_SIZE` constant value. Supported image formats are: PNG, JPEG, GIF. |

**Return values**

`(object)` Returns an object containing the IDs of the created images under the `imageids` property. The order of the returned IDs matches the order of the passed images.

**Examples**

Create an image

Create a cloud icon.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "image.create",
    "params": {
        "imagetype": 1,
        "name": "Cloud_(24)",
        "image": "iVBORw0KGgoAAAANSUhEUgAAABgAAAANCAYAAACzbK7QAAAAALAAAAAABHNCVQICAgIgAhkiAAAAA1wSF1zAAACnMwAAACmAAAApgBNtNH3wAAABl0RVh0U29mdHdhcmUAd3d3L...2lvvW/V2EBssnxlSGmWsxljw0znV9XfPLjTCW84r+cn7Jc8c2eWrbM6Wbe6/aTJbhJ/TNkWc9/xXW592Xb9iPkKnUfH8BKdLgFy0lDyQAAAAASUVORK5CYII="
    },
    "auth": "038e1d7b1735c6a5436ee9eae096879e",
    "id": 1
}
```

**Response:**

1028
image.delete

Description

`object image.delete(array imageIds)`

This method allows to delete images.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See `User roles` for more information.

Parameters

(array) IDs of the images to delete.

Return values

(object) Returns an object containing the IDs of the deleted images under the `imageids` property.

Examples

Delete multiple images

Delete two images.

Request:

```
{
   "jsonrpc": "2.0",
   "method": "image.delete",
   "params": [
      "188",
      "192"
   ],
   "auth": "3a57200802b24cda67c4e4010b50c065",
   "id": 1
}
```

Response:

```
{
   "jsonrpc": "2.0",
   "result": {
      "imageids": [
         "188",
         "192"
      ]
   },
   "id": 1
}
```
image.get

Description

integer/array image.get(object parameters)

The method allows to retrieve images according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imageids</td>
<td>string/array</td>
<td>Return only images with the given IDs.</td>
</tr>
<tr>
<td>sysmapids</td>
<td>string/array</td>
<td>Return images that are used on the given maps.</td>
</tr>
<tr>
<td>select_image</td>
<td>flag</td>
<td>Return an image property with the Base64 encoded image.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
</tbody>
</table>

Value possible: imageid, name.

countOutput boolean

These parameters being common for all get methods are described in detail in the reference commentary.

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieve an image

Retrieve all data for image with ID "2".

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "image.get",
   "params": {
      "output": "extend",
      "select_image": true,
      "imageids": "2"
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```

Response:
{
  "jsonrpc": "2.0",
  "result": [
    {
      "imageid": "2",
      "imagetype": "1",
      "name": "Cloud_(24)",
      "image": "iVBORw0KGgoAAAANSUhEUgAAAAwFAYAAAAd/2w0kgAAABmJLR0QA/wDQ8h8gAAAGZUlEQVR42mP8/A0w2xunjMi8Ej+99U2Mh2qL...
    }
  ],
  "id": 1
}

Source

CImage::get() in ui/include/classes/api/services/CImage.php.

**image.update**

Description

object image.update(object/array images)

This method allows to update existing images.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Image properties to be updated.

The imageid property must be defined for each image, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard image properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>string</td>
<td>Base64 encoded image. The maximum size of the encoded image is 1 MB. Maximum size can be adjusted by changing ZBX_MAX_IMAGE_SIZE constant value. Supported image formats are: PNG, JPEG, GIF.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated images under the imageids property.

Examples

Rename image

Rename image to “Cloud icon”.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "image.update",
  "params": {
    "imageid": "2",
    "name": "Cloud icon"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:
Item

This class is designed to work with items.

Object references:

- Item

Available methods:

- `item.create` - creating new items
- `item.delete` - deleting items
- `item.get` - retrieving items
- `item.update` - updating items

> Item object

The following objects are directly related to the `item` API.

Item

Web items cannot be directly created, updated or deleted via the Zabbix API.

The item object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>itemid</code></td>
<td>string</td>
<td>(readonly) ID of the item.</td>
</tr>
<tr>
<td><code>delay</code></td>
<td>string</td>
<td>(required) Update interval of the item. Accepts seconds or a time unit with suffix (30s,1m,2h,1d). Optionally one or more custom intervals can be specified either as flexible intervals or scheduling. Multiple intervals are separated by a semicolon. User macros may be used. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. Flexible intervals may be written as two macros separated by a forward slash (e.g. <code>{FLEX_INTERVAL}/{FLEX_PERIOD}</code>). Optional for Zabbix trapper, dependent items and for Zabbix agent (active) with <code>mqtt.get</code> key.</td>
</tr>
<tr>
<td><code>hostid</code></td>
<td>string</td>
<td>(required) ID of the host or template that the item belongs to. For update operations this field is readonly.</td>
</tr>
<tr>
<td><code>interfaceid</code></td>
<td>string</td>
<td>(required) ID of the item’s host interface. Used only for host items. Not required for Zabbix agent (active), Zabbix internal, Zabbix trapper, calculated, dependent, database monitor and script items. Optional for HTTP agent items.</td>
</tr>
<tr>
<td><code>key_</code></td>
<td>string</td>
<td>(required) Item key.</td>
</tr>
</tbody>
</table>

Source

`CImage::update()` in `ui/include/classes/api/services/CImage.php`. 

```json
{
    "jsonrpc": "2.0",
    "result": {
        "imageids": ["2"],
    },
    "id": 1
}
```
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the item.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>Type of the item.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>URL string, required only for HTTP agent item type. Supports user macros:</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>{HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST},</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{HOST.NAME}, {ITEM.ID}, {ITEM.KEY}.</td>
</tr>
<tr>
<td>value_type</td>
<td>integer</td>
<td>Type of information of the item.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>allow_traps</td>
<td>integer</td>
<td>HTTP agent item field. Allow to populate value as in trapper item type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>also.</td>
</tr>
<tr>
<td>authtype</td>
<td>integer</td>
<td>SSH agent authentication method possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) password;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - public key.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HTTP agent authentication method possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) none</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - basic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - NTLM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - Kerberos</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Description of the item.</td>
</tr>
<tr>
<td>error</td>
<td>string</td>
<td>(readonly) Error text if there are problems updating the item.</td>
</tr>
<tr>
<td>flags</td>
<td>integer</td>
<td>(readonly) Origin of the item.</td>
</tr>
<tr>
<td>follow_redirects</td>
<td>integer</td>
<td>HTTP agent item field. Follow response redirects while pooling data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Do not follow redirects.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) Follow redirects.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>headers</td>
<td>object</td>
<td>HTTP agent item field. Object with HTTP(S) request headers, where header name is used as key and header value as value.</td>
</tr>
<tr>
<td>history</td>
<td>string</td>
<td>A time unit of how long the history data should be stored. Also accepts user macro.</td>
</tr>
<tr>
<td>http_proxy</td>
<td>string</td>
<td>HTTP agent item field. HTTP(S) proxy connection string.</td>
</tr>
<tr>
<td>inventory_link</td>
<td>integer</td>
<td>ID of the host inventory field that is populated by the item. Refer to the host inventory page for a list of supported host inventory fields and their IDs.</td>
</tr>
<tr>
<td>ipmi_sensor</td>
<td>string</td>
<td>IPMI sensor. Used only by IPMI items.</td>
</tr>
<tr>
<td>jmx_endpoint</td>
<td>string</td>
<td>JMX agent custom connection string.</td>
</tr>
<tr>
<td>lastclock</td>
<td>timestamp</td>
<td>(readonly) Time when the item was last updated. By default, only values that fall within the last 24 hours are displayed. You can extend this time period by changing the value of Max history display period parameter in the Administration → General menu section.</td>
</tr>
<tr>
<td>lastns</td>
<td>integer</td>
<td>(readonly) Nanoseconds when the item was last updated. By default, only values that fall within the last 24 hours are displayed. You can extend this time period by changing the value of Max history display period parameter in the Administration → General menu section.</td>
</tr>
<tr>
<td>lastvalue</td>
<td>string</td>
<td>(readonly) Last value of the item. By default, only values that fall within the last 24 hours are displayed. You can extend this time period by changing the value of Max history display period parameter in the Administration → General menu section.</td>
</tr>
<tr>
<td>logtimefmt</td>
<td>string</td>
<td>Format of the time in log entries. Used only by log items.</td>
</tr>
<tr>
<td>master_itemid</td>
<td>integer</td>
<td>Master item ID. Recursion up to 3 dependent items and maximum count of dependent items equal to 29999 are allowed.</td>
</tr>
<tr>
<td>output_format</td>
<td>integer</td>
<td>HTTP agent item field. Should response be converted to JSON. 0 - (default) Store raw. 1 - Convert to JSON.</td>
</tr>
<tr>
<td>params</td>
<td>string</td>
<td>Additional parameters depending on the type of the item: executed script for SSH and Telnet items; SQL query for database monitor items; formula for calculated items; the script for script item.</td>
</tr>
<tr>
<td>parameters</td>
<td>array</td>
<td>Additional parameters for script items. Array of objects with ‘name’ and ‘value’ properties, where name must be unique.</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. When used by JMX, username should also be specified together with password or both properties should be left blank.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>post_type</td>
<td>integer</td>
<td>HTTP agent item field. Type of post data body stored in posts property.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Raw data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - JSON data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - XML data.</td>
</tr>
<tr>
<td>posts</td>
<td>string</td>
<td>HTTP agent item field. HTTP(S) request body data. Used with post_type.</td>
</tr>
<tr>
<td>prevvalue</td>
<td>string</td>
<td>(readonly) Previous value of the item.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>By default, only values that fall within the last 24 hours are displayed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You can extend this time period by changing the value of Max history display period parameter in the Administration → General menu section.</td>
</tr>
<tr>
<td>privatekey</td>
<td>string</td>
<td>Name of the private key file.</td>
</tr>
<tr>
<td>publickey</td>
<td>string</td>
<td>Name of the public key file.</td>
</tr>
<tr>
<td>query_fields</td>
<td>array</td>
<td>HTTP agent item field. Query parameters. Array of objects with 'key':'value' pairs, where value can be empty string.</td>
</tr>
<tr>
<td>request_method</td>
<td>integer</td>
<td>HTTP agent item field. Type of request method.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) GET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - POST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - PUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - HEAD</td>
</tr>
<tr>
<td>retrieve_mode</td>
<td>integer</td>
<td>HTTP agent item field. What part of response should be stored.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Headers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Both body and headers will be stored.</td>
</tr>
<tr>
<td>snmp_oid</td>
<td>string</td>
<td>SNMP OID.</td>
</tr>
<tr>
<td>ssl_cert_file</td>
<td>string</td>
<td>HTTP agent item field. Public SSL Key file path.</td>
</tr>
<tr>
<td>ssl_key_file</td>
<td>string</td>
<td>HTTP agent item field. Private SSL Key file path.</td>
</tr>
<tr>
<td>ssl_key_password</td>
<td>string</td>
<td>HTTP agent item field. Password for SSL Key file. (readonly) State of the item.</td>
</tr>
<tr>
<td>state</td>
<td>integer</td>
<td>Status of the item.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) normal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - not supported.</td>
</tr>
<tr>
<td>status_codes</td>
<td>string</td>
<td>HTTP agent item field. Ranges of required HTTP status codes separated by commas. Also supports user macros as part of comma separated list.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: 200,200-{$M},{$M},200-400</td>
</tr>
<tr>
<td>templateid</td>
<td>string</td>
<td>(readonly) ID of the parent template item.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hint: Use the hostid property to specify the template that the item belongs to.</td>
</tr>
<tr>
<td>timeout</td>
<td>string</td>
<td>Item data polling request timeout. Used for HTTP agent and script items. Supports user macros.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>default: 3s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>maximum value: 60s</td>
</tr>
<tr>
<td>trapper_hosts</td>
<td>string</td>
<td>Allowed hosts. Used by trapper items or HTTP agent items.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>trends</td>
<td>string</td>
<td>A time unit of how long the trends data should be stored. Also accepts user macro. Default: 365d. Value units.</td>
</tr>
<tr>
<td>units</td>
<td>string</td>
<td>Value units.</td>
</tr>
<tr>
<td>username</td>
<td>string</td>
<td>Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. Required by SSH and Telnet items. When used by JMX, password should also be specified together with username or both properties should be left blank.</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Universal unique identifier, used for linking imported item to already existing ones. Used only for items on templates. Auto-generated, if not given. For update operations this field is readonly.</td>
</tr>
<tr>
<td>valuemapid</td>
<td>string</td>
<td>ID of the associated value map. HTTP agent item field. Validate host name in URL is in Common Name field or a Subject Alternate Name field of host certificate. 0 - (default) Do not validate. 1 - Validate.</td>
</tr>
<tr>
<td>verify_host</td>
<td>integer</td>
<td>HTTP agent item field. Validate is host certificate authentic. 0 - (default) Do not validate. 1 - Validate.</td>
</tr>
</tbody>
</table>

**Item tag**

The item tag object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Item tag name. (required)</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Item tag value.</td>
</tr>
</tbody>
</table>

**Item preprocessing**

The item preprocessing object has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>integer</td>
<td>The preprocessing option type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Custom multiplier;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Right trim;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - Left trim;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - Trim;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - Regular expression matching;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 - Boolean to decimal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 - Octal to decimal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 - Hexadecimal to decimal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 - Simple change;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 - Change per second;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 - XML XPath;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 - JSONPath;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 - In range;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14 - Matches regular expression;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15 - Does not match regular expression;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16 - Check for error in JSON;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17 - Check for error in XML;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18 - Check for error using regular expression;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19 - Discard unchanged;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 - Discard unchanged with heartbeat;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21 - JavaScript;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22 - Prometheus pattern;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23 - Prometheus to JSON;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24 - CSV to JSON;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25 - Replace;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26 - Check unsupported;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27 - XML to JSON.</td>
</tr>
<tr>
<td>params</td>
<td>string</td>
<td>Additional parameters used by preprocessing option. Multiple parameters are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>separated by LF (\n) character.</td>
</tr>
<tr>
<td>error_handler</td>
<td>integer</td>
<td>Action type used in case of preprocessing step failure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Error message is set by Zabbix server;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Discard value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Set custom value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - Set custom error message.</td>
</tr>
<tr>
<td>error_handler_params</td>
<td>string</td>
<td>Error handler parameters. Used with error_handler.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Must be empty, if error_handler is 0 or 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Can be empty if, error_handler is 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cannot be empty, if error_handler is 3.</td>
</tr>
</tbody>
</table>

The following parameters and error handlers are supported for each preprocessing type.

<table>
<thead>
<tr>
<th>Preprocessing type</th>
<th>Name</th>
<th>Parameter 1</th>
<th>Parameter 2</th>
<th>Parameter 3</th>
<th>Supported error handlers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Custom number(^1),(^6)</td>
<td>mult-</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>iliplier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Right trim</td>
<td>list of</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>characters(^2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Left trim</td>
<td>list of</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>characters(^2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Trim</td>
<td>list of</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>characters(^2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preprocessing type</td>
<td>Name</td>
<td>Parameter 1</td>
<td>Parameter 2</td>
<td>Parameter 3</td>
<td>Supported error handlers</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>5</td>
<td>Regular expression pattern</td>
<td>output</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>6</td>
<td>Boolean to decimal</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>7</td>
<td>Octal to decimal</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>8</td>
<td>Hexadecimal to decimal</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>9</td>
<td>Simple change</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>10</td>
<td>Change per second</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>11</td>
<td>XML path</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>12</td>
<td>JSONPath path</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>13</td>
<td>In range</td>
<td>min&lt;sup&gt;1,6&lt;/sup&gt;</td>
<td>max&lt;sup&gt;1,6&lt;/sup&gt;</td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>14</td>
<td>Matches pattern</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>15</td>
<td>Does not match regular pattern</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>16</td>
<td>Check for error in XML path</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>17</td>
<td>Check for error in JSON path</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>18</td>
<td>Check for error using regular expression pattern output</td>
<td>0, 1, 2, 3</td>
<td>0, 1, 2, 3</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>Preprocessing type</td>
<td>Name</td>
<td>Parameter 1</td>
<td>Parameter 2</td>
<td>Parameter 3</td>
<td>Supported error handlers</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>19</td>
<td>Discard un-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>changed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Discard un-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>changed with heart-beat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>with heartbeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>JavaScript script</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Prometheus pattern</td>
<td>value, label, output</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td></td>
<td>to JSON</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Prometheus pattern</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td></td>
<td>to JSON</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>CSV to JSON</td>
<td>character</td>
<td>character</td>
<td>0, 1</td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Replace search string</td>
<td>replacement</td>
<td></td>
<td></td>
<td>1, 2, 3</td>
</tr>
<tr>
<td></td>
<td>Check</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>unsupported</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>to JSON</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>XML to JSON</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
</tbody>
</table>

1. integer or floating-point number
2. string
3. regular expression
4. JSONPath or XML XPath
5. positive integer (with support of time suffixes, e.g. 30s, 1m, 2h, 1d)
6. user macro
7. Prometheus pattern following the syntax: `<metric name>\{{label name}="<label value>", ...\} == <value>`. Each Prometheus pattern component (metric, label name, label value and metric value) can be user macro.
8. Prometheus output following the syntax: `<label name>` (can be a user macro) if label is selected as the second parameter.
9. One of the aggregation functions: sum, min, max, avg, count if function is selected as the second parameter.

**item.create**

Description

object item.create(object/array items)

This method allows to create new items.

Web items cannot be created via the Zabbix API.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Items to create.

Additionally to the standard item properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>preprocessing</td>
<td>array</td>
<td>Item preprocessing options.</td>
</tr>
<tr>
<td>tags</td>
<td>array</td>
<td>Item tags.</td>
</tr>
</tbody>
</table>

1039
Return values

(object) Returns an object containing the IDs of the created items under the itemids property. The order of the returned IDs matches the order of the passed items.

Examples

Creating an item

Create a numeric Zabbix agent item with 2 item tags to monitor free disk space on host with ID "30074".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Free disk space on /home/joe/",
        "key_": "vfs.fs.size[/home/joe/,free]",
        "hostid": "30074",
        "type": 0,
        "value_type": 3,
        "interfaceid": "30084",
        "tags": [
            { "tag": "Disc usage" },
            { "tag": "Equipment",
              "value": "Workstation"
            }
        ],
        "delay": "30s"
    },
    "auth": "038e1d7b1735c6a5436ee9eae95879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24758"
        ]
    },
    "id": 1
}
```

Creating a host inventory item

Create a Zabbix agent item to populate the host’s “OS” inventory field.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "uname",
        "key_": "system.uname",
        "hostid": "30021",
        "type": 0,
        "interfaceid": "30007",
        "value_type": 1,
        "delay": "10s",
        "inventory_link": 5
    },
}
```

Creating an item with preprocessing

Create an item using custom multiplier.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "item.create",
  "params": {
    "name": "Device uptime",
    "key": "sysUpTime",
    "hostid": "11312",
    "type": 4,
    "snmp_oid": "SNMPv2-MIB::sysUpTime.0",
    "value_type": 1,
    "delay": "60s",
    "units": "uptime",
    "interfaceid": "1156",
    "preprocessing": [
      {
        "type": 1,
        "params": "0.01",
        "error_handler": 1,
        "error_handler_params": ""
      }
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
  }
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": {
    "itemids": [
      "24759"
    ],
  },
  "id": 1
}
```

Creating dependent item

Create a dependent item for the master item with ID 24759. Only dependencies on the same host are allowed, therefore master and the dependent item should have the same hostid.

Request:

```json
{
  "jsonrpc": "2.0",
  "result": {
    "itemids": [
      "44210"
    ],
  },
  "id": 1
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": {
    "itemids": [
      "24759"
    ],
  },
  "id": 1
}
```
Create HTTP agent item

Create POST request method item with JSON response preprocessing.

Request:

```
{  "jsonrpc": "2.0",  "method": "item.create",  "params": {    "hostid": "30074",    "name": "Dependent test item",    "key_": "dependent.item",    "type": 18,    "master_itemid": "24759",    "value_type": 2  },  "auth": "038e1d7b1735c6a5436ee9eae095879e",  "id": 1}
```

Response:

```
{  "jsonrpc": "2.0",  "result": {    "itemids": [      "44211"    ]  },  "id": 1}
```

```
CreateHTTPagentitem
CreatePOSTrequestmethoditemwithJSONresponsepreprocessing.
Request:

{  "jsonrpc": "2.0",  "method": "item.create",  "params": {    "url": "http://127.0.0.1/http.php",    "query_fields": [      {        "mode": "json"      },      {        "min": "10"      },      {        "max": "100"      }    ],    "interfaceid": "1",    "type": 19,    "hostid": "10254",    "delay": "5s",    "key_": "json",    "name": "HTTP agent example JSON",    "value_type": 0,    "output_format": 1,    "preprocessing": [      {        "type": 12,        "params": "$.random",        "error_handler": 0,        "error_handler_params": ""      }    ]  },  "id": 1042
```
Create script item

Create a simple data collection using a script item.

Request:
```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Script example",
        "key_": "custom.script.item",
        "hostid": "12345",
        "type": 21,
        "value_type": 4,
        "params": "var request = new HttpRequest();\n        return request.post("https://postman-echo.com/post\n        ", JSON.parse(value));",
        "parameters": [
            {
                "name": "host",
                "value": "{HOST.CONN}" 
            }
        ],
        "timeout": "6s",
        "delay": "30s"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

Response:
```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 3
}
```

Source
CItem::create() in ui/include/classes/api/services/CItem.php.

**item.delete**

Description
object item.delete(array itemIds)
This method allows to delete items.

Web items cannot be deleted via the Zabbix API.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the items to delete.

Return values

(object) Returns an object containing the IDs of the deleted items under the itemids property.

Examples

Deleting multiple items

Delete two items. Dependent items and item prototypes are removed automatically if master item is deleted.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.delete",
    "params": [
        "22982",
        "22986"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22982",
            "22986"
        ]
    },
    "id": 1
}
```

Source

CItem::delete() in ui/include/classes/api/services/CItem.php.

**item.get**

Description

integer/array item.get(object parameters)

The method allows to retrieve items according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemids</td>
<td>string/array</td>
<td>Return only items with the given IDs.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only items that belong to the hosts from the given groups.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>templateids</td>
<td>string/array</td>
<td>Return only items that belong to the given templates.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only items that belong to the given hosts.</td>
</tr>
<tr>
<td>proxyids</td>
<td>string/array</td>
<td>Return only items that are monitored by the given proxies.</td>
</tr>
<tr>
<td>interfaceids</td>
<td>string/array</td>
<td>Return only items that use the given host interfaces.</td>
</tr>
<tr>
<td>graphids</td>
<td>string/array</td>
<td>Return only items that are used in the given graphs.</td>
</tr>
<tr>
<td>triggerids</td>
<td>string/array</td>
<td>Return only items that are used in the given triggers.</td>
</tr>
<tr>
<td>webitems</td>
<td>flag</td>
<td>Include web items in the result.</td>
</tr>
<tr>
<td>inherited</td>
<td>boolean</td>
<td>If set to true return only items inherited from a template.</td>
</tr>
<tr>
<td>templated</td>
<td>boolean</td>
<td>If set to true return only items that belong to templates.</td>
</tr>
<tr>
<td>monitored</td>
<td>boolean</td>
<td>If set to true return only enabled items that belong to monitored hosts.</td>
</tr>
<tr>
<td>group</td>
<td>string</td>
<td>Return only items that belong to a group with the given name.</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>Return only items that belong to a host with the given name.</td>
</tr>
<tr>
<td>evaltype</td>
<td>integer</td>
<td>Rules for tag searching.</td>
</tr>
</tbody>
</table>

Possible values:
0 - (default) And/Or; 2 - Or.

tags | array of objects | Return only items with given tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value. Format: [{"tag": "<tag>", "value": "<value>", "operator": "<operator>", ...}]. An empty array returns all items.

Possible operator types:
0 - (default) Like; 1 - Equal; 2 - Not like; 3 - Not equal; 4 - Exists; 5 - Not exists.

with_triggers | boolean | If set to true return only items that are used in triggers. |
selectHosts | query | Return a hosts property with an array of hosts that the item belongs to. |
selectInterfaces | query | Return an interfaces property with an array of host interfaces used by the item. |
selectTriggers | query | Return a triggers property with the triggers that the item is used in. |
selectGraphs | query | Return a graphs property with the graphs that contain the item. |
selectDiscoveryRule | query | Return a discoveryRule property with the LLD rule that created the item. |
selectItemDiscovery | query | Return an itemDiscovery property with the item discovery object. The item discovery object links the item to an item prototype from which it was created. |

It has the following properties:
itemdiscoveryid - (string) ID of the item discovery; itemid - (string) ID of the discovered item; parent_itemid - (string) ID of the item prototype from which the item has been created; key - (string) key of the item prototype; lastcheck - (timestamp) time when the item was last discovered; ts_delete - (timestamp) time when an item that is no longer discovered will be deleted.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>selectPreprocessing</td>
<td>query</td>
<td>Return a <code>preprocessing</code> property with item preprocessing options.</td>
</tr>
</tbody>
</table>
| params            | (string)   | Additional parameters used by preprocessing option. Multiple parameters are separated by LF (`
`) character. |
| error_handler     | (string)   | Action type used in case of preprocessing step failure:                    |
| host              |            | Technical name of the host that the item belongs to.                       |
| limitSelects      | integer    | Limits the number of records returned by subselects.                       |
| sortfield         | string/array | Sort the result by the given properties. Possible values are: `itemid`, `name`, `key_`, `delay`, `history`, `trends`, `type` and `status`. |
| countOutput       | boolean    | These parameters being common for all `get` methods are described in detail in the reference commentary page. |
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

### Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

### Examples

#### Finding items by key

Retrieve all items used in triggers for specific host ID that have word “system.cpu” in the item key and sort results by name.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "hostids": ["10084"],
        "with_triggers": true,
        "search": {
            "key_": "system.cpu"
        },
        "sortfield": "name"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": [
    {
        "itemid": "42269",
        "type": "18",
        "snmp_oid": "",
        "hostid": "10084",
        "name": "CPU utilization",
        "key_": "system.cpu.util",
        "delay": "0",
        "history": "7d",
        "trend": "365d",
        "status": "0",
        "value_type": "0",
        "trapper_hosts": "",
        "units": "%",
        "logtimefmt": "",
        "templateid": "42267",
        "valuemapid": "0",
        "params": ""
    }
    ]
}
```
"ipmi_sensor": "",
"authtype": "0",
"username": "",
"password": "",
"publickey": "",
"privatekey": "",
"flags": "0",
"interfaceid": "0",
"description": "CPU utilization in \%.",
"inventory_link": "0",
"evaltype": "0",
"jmx_endpoint": "",
"master_itemid": "42264",
"timeout": "3s",
"url": "",
"query_fields": [],
"posts": "",
"status_codes": "200",
"follow_redirects": "1",
"post_type": "0",
"http_proxy": "",
"headers": [],
"retrieve_mode": "0",
"request_method": "0",
"output_format": "0",
"ssl_cert_file": "",
"ssl_key_file": "",
"ssl_key_password": "",
"verify_peer": "0",
"verify_host": "0",
"allow_traps": "0",
"uuid": "",
"state": "0",
"error": "",
"parameters": [],
"lastclock": "0",
"lastns": "0",
"lastvalue": "0",
"prevvalue": "0"
},
{
"itemid": "42259",
"type": "0",
"snmp_oid": "",
"hostid": "10084",
"name": "Load average (15m avg)",
"key_": "system.cpu.load[all,avg15]",
"delay": "1m",
"history": "7d",
"trends": "365d",
"status": "0",
"value_type": "0",
"trapper_hosts": "",
"units": "",
"logtimefmt": "",
"templateid": "42219",
"valuemapid": "0",
"params": "",
"ipmi_sensor": "",
"authtype": "0",
"username": "",
"password": ""
"publickey": "",
"privatekey": "",
"flags": "0",
"interfaceid": "1",
"description": "",
"inventory_link": "0",
"evaltype": "0",
"jmx_endpoint": "",
"master_itemid": "0",
"timeout": "3s",
"url": "",
"query_fields": [],
"posts": "",
"status_codes": "200",
"follow_redirs": "0",
"post_type": "0",
"http_proxy": "",
"headers": [],
"retrieve_mode": "0",
"request_method": "0",
"output_format": "0",
"ssl_cert_file": "",
"ssl_key_file": "",
"ssl_key_password": "",
"verify_peer": "0",
"verify_host": "0",
"allow_traps": "0",
"uuid": "",
"state": "0",
"error": "",
"parameters": [],
"lastclock": "0",
"lastns": "0",
"lastvalue": "0",
"prevvalue": "0"
},
{
"itemid": "42249",
"type": "0",
"snmp_oid": "",
"hostid": "10084",
"name": "Load average (1m avg)",
"key_": "system.cpu.load[all,avg1]",
"delay": "1m",
"history": "7d",
"trends": "365d",
"status": "0",
"value_type": "0",
"trapper_hosts": "",
"units": "",
"logtimefmt": "",
"templateid": "42209",
"valuemapid": "0",
"params": "",
"ipmi_sensor": "",
"authtype": "0",
"username": "",
"password": "",
"publickey": "",
"privatekey": "",
"flags": "0",
"interfaceid": "1"}
<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>inventory_link</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>evaltype</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>jmx_endpoint</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>master_itemid</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>timeout</td>
<td>&quot;3s&quot;</td>
</tr>
<tr>
<td>url</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>query_fields</td>
<td>[]</td>
</tr>
<tr>
<td>posts</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>status_codes</td>
<td>[&quot;200&quot;]</td>
</tr>
<tr>
<td>follow_redirects</td>
<td>&quot;1&quot;</td>
</tr>
<tr>
<td>post_type</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>http_proxy</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>headers</td>
<td>[]</td>
</tr>
<tr>
<td>retrieve_mode</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>request_method</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>output_format</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>ssl_cert_file</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>ssl_key_file</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>ssl_key_password</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>verify_peer</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>verify_host</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>allow_traps</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>uuid</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>state</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>error</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>parameters</td>
<td>[]</td>
</tr>
<tr>
<td>lastclock</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>lastns</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>lastvalue</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>prevvalue</td>
<td>&quot;0&quot;</td>
</tr>
</tbody>
</table>

```
```
```

```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
"master_itemid": "0",
"timeout": "3s",
"url": "",
"query_fields": [],
"posts": "",
"status_codes": "200",
"follow_redirects": "1",
"post_type": "0",
"http_proxy": "",
"headers": [],
"retrieve_mode": "0",
"request_method": "0",
"output_format": "0",
"ssl_cert_file": "",
"ssl_key_file": "",
"ssl_key_password": "",
"verify_peer": "0",
"verify_host": "0",
"allow_traps": "0",
"uuid": "",
"state": "0",
"error": "",
"parameters": [],
"lastclock": "0",
"lastns": "0",
"lastvalue": "0",
"prevvalue": "0"
},
{
  "itemid": "42260",
  "type": "0",
  "snmp_oid": "",
  "hostid": "10084",
  "name": "Number of CPUs",
  "key_": "system.cpu.num",
  "delay": "1m",
  "history": "7d",
  "trends": "365d",
  "status": "0",
  "value_type": "3",
  "trapper_hosts": "",
  "units": "",
  "logtimefmt": "",
  "templateid": "42220",
  "valuemapid": "0",
  "params": "",
  "ipmi_sensor": "",
  "authtype": "0",
  "username": "",
  "password": "",
  "publickey": "",
  "privatekey": "",
  "flags": "0",
  "interfaceid": "1",
  "description": "",
  "inventory_link": "0",
  "evaltype": "0",
  "jmx_endpoint": "",
  "master_itemid": "0",
  "timeout": "3s",
  "url": "",
  "query_fields": []
}
Finding dependent items by key

Retrieve all dependent items from host with ID "10116" that have the word "apache" in the key.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "item.get",
  "params": {
    "output": "extend",
    "hostids": "10116",
    "search": {
      "key_": "apache"
    },
    "filter": {
      "type": 18
    }
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": [
    {
      "itemid": "25550",
      "type": "18",
      "snmp_oid": "",
      "hostid": "10116",
      "name": "Days",
      "key_": "apache.status.uptime.days",
      "delay": "0",
      "history": "90d"
    }
  ]
}
```
"trends": "365d",
"status": "0",
"value_type": "3",
"trapper_hosts": "",
"units": "",
"logtimefmt": "",
"templateid": "0",
"valuemapid": "0",
"params": "",
"ipmi_sensor": "",
"authtype": "0",
"username": "",
"password": "",
"publickey": "",
"privatekey": "",
"flags": "0",
"interfaceid": "0",
"description": "",
"inventory_link": "0",
"evaltype": "0",
"jmx_endpoint": "",
"master_itemid": "25545",
"timeout": "3s",
"url": "",
"query_fields": [],
"posts": "",
"status_codes": "200",
"follow_redirects": "1",
"post_type": "0",
"http_proxy": "",
"headers": [],
"retrieve_mode": "0",
"request_method": "0",
"output_format": "0",
"ssl_cert_file": "",
"ssl_key_file": "",
"ssl_key_password": "",
"verify_peer": "0",
"verify_host": "0",
"allow_traps": "0",
"uuid": "",
"state": "0",
"error": "",
"parameters": [],
"lastclock": "0",
"lastns": "0",
"lastvalue": "0",
"prevvalue": "0"
},
{
"itemid": "25555",
"type": "18",
"snmp_oid": "",
"hostid": "10116",
"name": "Hours",
"key_": "apache.status.uptime.hours",
"delay": "0",
"history": "90d",
"trends": "365d",
"status": "0",
"value_type": "3",
"trapper_hosts": ""
Find HTTP agent item

Find HTTP agent item with post body type XML for specific host ID.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "hostids": "10255",
        "filter": {
            "type": 19,
            "post_type": 3
        }
    }
}
```

Response:
{
   "jsonrpc": "2.0",
   "result": [
   {
      "itemid": "28252",
      "type": "19",
      "snmp_oid": ",",
      "hostid": "10255",
      "name": "template item",
      "key_": "ti",
      "delay": "30s",
      "history": "90d",
      "trends": "365d",
      "status": "0",
      "value_type": "3",
      "trapper_hosts": ",",
      "units": ",",
      "logtimefmt": ",",
      "templateid": "0",
      "valuemapid": "0",
      "params": "",
      "ipmi_sensor": "",
      "authtype": "0",
      "username": ",",
      "password": ",",
      "publickey": ",",
      "privatekey": ",",
      "flags": "0",
      "interfaceid": "0",
      "description": ",",
      "inventory_link": "0",
      "evaltype": "0",
      "jmx_endpoint": ",",
      "master_itemid": "0",
      "timeout": "3s",
      "url": "localhost",
      "query_fields": [
      {
         "mode": "xml"
      }
      ],
      "posts": "<body><"}[
(CDATA[{$MACRO}<foo</bar>])]<r"n</body>>", "status_codes": "200",
      "follow_redirects": "0",
      "post_type": "3",
      "http_proxy": "",
      "headers": []
   }
   ],
   "posts": "<body><r"n<! [CDATA[{$MACRO}<foo</bar>]]><r"n</body>>", "status_codes": "200",
   "follow_redirects": "0",
   "post_type": "3",
   "http_proxy": "",
   "headers": []
}
Retrieving items with preprocessing rules

Retrieve all items and their preprocessing rules for specific host ID.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": ["itemid", "name", "key_"]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "itemid": "23865",
        "name": "HTTP agent example JSON",
        "key_": "json",
        "preprocessing": [
            {
                "type": "12",
                "params": "$.random",
                "error_handler": "1",
                "error_handler_params": ""
            }
        ],
        "id": 1
    }
}
```

See also

- Discovery rule
- Graph
- Host
- Host interface
- Trigger

Source

CItem::get() in ui/include/classes/api/services/CItem.php.

**item.update**
object item.update(object/array items)
This method allows to update existing items.

Web items cannot be updated via the Zabbix API.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(object/array) Item properties to be updated.
The itemid property must be defined for each item, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard item properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>preprocessing</td>
<td>array</td>
<td>Item preprocessing options to replace the current preprocessing options.</td>
</tr>
<tr>
<td>tags</td>
<td>array</td>
<td>Item tags.</td>
</tr>
</tbody>
</table>

Return values
(object) Returns an object containing the IDs of the updated items under the itemids property.

Examples
Enabling an item
Enable an item, that is, set its status to "0".

Request:
```
{   "jsonrpc": "2.0",   "method": "item.update",   "params": {     "itemid": "10092",     "status": 0   },   "auth": "700ca65537074ec963db7efabda78259",   "id": 1 }
```

Response:
```
{   "jsonrpc": "2.0",   "result": {     "itemids": [       "10092"     ],   }   }   
```

Update dependent item
Update Dependent item name and Master item ID. Only dependencies on same host are allowed, therefore Master and Dependent item should have same hostid.

Request:
```
{   "jsonrpc": "2.0",   "method": "item.update",   "params": {     "name": "Dependent item updated name",     "master_itemid": "25562",   } }
```
"itemid": "189019",
"auth": "700ca65537074ec963db7efabda78259",
"id": 1
}

Response:
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "189019"
        ],
    },
    "id": 1
}

Update HTTP agent item
Enable item value trapping.

Request:
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "23856",
        "allow_traps": 1
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}

Response:
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23856"
        ],
    },
    "id": 1
}

Updating an item with preprocessing
Update an item with item preprocessing rule "In range".

Request:
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "23856",
        "preprocessing": [
            {
                "type": 13,
                "params": "\n100",
                "error_handler": 1,
                "error_handler_params": ""
            }
        ]
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
Updating a script item

Update a script item with a different script and remove unnecessary parameters that were used by previous script.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "item.update",
   "params": {
      "itemid": "23865",
      "parameters": [],
      "script": "Zabbix.log(3, 'Log test');\nreturn 1;"
   },
   "auth": "700ca65537074ec963db7efabda78259",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "itemids": ["23865"
   ],
   "id": 1
}
```

Source

CItem::update() in ui/include/classes/api/services/CItem.php.

**Item prototype**

This class is designed to work with item prototypes.

Object references:

- Item prototype

Available methods:

- itemprototype.create - creating new item prototypes
- itemprototype.delete - deleting item prototypes
- itemprototype.get - retrieving item prototypes
- itemprototype.update - updating item prototypes

> Item prototype object
The following objects are directly related to the itemprototype API.

### Item prototype

The item prototype object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemid</td>
<td>string</td>
<td>(readonly) ID of the item prototype. Accepts seconds or a time unit with suffix (30s,1m,2h,1d). Optionally one or more custom intervals can be specified either as flexible intervals or scheduling. Multiple intervals are separated by a semicolon. User macros and LLD macros may be used. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. Flexible intervals may be written as two macros separated by a forward slash (e.g. <code>$FLEX_INTERVAL</code>/<code>$FLEX_PERIOD</code>).</td>
</tr>
<tr>
<td>delay</td>
<td>string</td>
<td>Update interval of the item prototype. Accepts seconds or a time unit with suffix (30s,1m,2h,1d). Optionally one or more custom intervals can be specified either as flexible intervals or scheduling. Multiple intervals are separated by a semicolon. User macros and LLD macros may be used. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. Flexible intervals may be written as two macros separated by a forward slash (e.g. <code>$FLEX_INTERVAL</code>/<code>$FLEX_PERIOD</code>).</td>
</tr>
<tr>
<td>hostid</td>
<td>string</td>
<td>ID of the host that the item prototype belongs to.</td>
</tr>
<tr>
<td>ruleid</td>
<td>string</td>
<td>ID of the LLD rule that the item belongs to.</td>
</tr>
<tr>
<td>interfaceid</td>
<td>string</td>
<td>ID of the item prototype's host interface. Used only for host item prototypes. Not required for Zabbix agent (active), Zabbix internal, Zabbix trapper, calculated, dependent, database monitor and script item prototypes. Optional for HTTP agent item prototypes.</td>
</tr>
<tr>
<td>key_</td>
<td>string</td>
<td>Item prototype key.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the item prototype.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>Type of the item prototype.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - Zabbix agent; 2 - Zabbix trapper; 3 - simple check; 5 - Zabbix internal; 7 - Zabbix agent (active); 10 - external check; 11 - database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent; 15 - calculated; 16 - JMX agent; 17 - SNMP trap; 18 - Dependent item; 19 - HTTP agent; 20 - SNMP agent; 21 - Script.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>URL string required only for HTTP agent item prototypes. Supports LLD macros, user macros. <code>{HOST.IP}</code>, <code>{HOST.CONN}</code>, <code>{HOST.DNS}</code>, <code>{HOST.HOST}</code>, <code>{HOST.NAME}</code>, <code>{ITEM.ID}</code>, <code>{ITEM.KEY}</code>.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>value_type</td>
<td>integer</td>
<td>Type of information of the item prototype. Possible values: 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text.</td>
</tr>
<tr>
<td>allow_traps</td>
<td>integer</td>
<td>HTTP agent item prototype field. Allow to populate value as in trapper item type also.</td>
</tr>
<tr>
<td>allow_traps</td>
<td>integer</td>
<td>0 - (default) Do not allow to accept incoming data. 1 - Allow to accept incoming data.</td>
</tr>
<tr>
<td>authtype</td>
<td>integer</td>
<td>Used only by SSH agent item prototypes or HTTP agent item prototypes.</td>
</tr>
<tr>
<td>authtype</td>
<td>integer</td>
<td>SSH agent authentication method possible values: 0 - (default) password; 1 - public key.</td>
</tr>
<tr>
<td>authtype</td>
<td>integer</td>
<td>HTTP agent authentication method possible values: 0 - (default) none 1 - basic 2 - NTLM 3 - Kerberos</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Description of the item prototype.</td>
</tr>
<tr>
<td>follow_redirects</td>
<td>integer</td>
<td>HTTP agent item prototype field. Follow response redirects while pooling data.</td>
</tr>
<tr>
<td>follow_redirects</td>
<td>integer</td>
<td>0 - Do not follow redirects. 1 - (default) Follow redirects.</td>
</tr>
<tr>
<td>headers</td>
<td>object</td>
<td>HTTP agent item prototype field. Object with HTTP(S) request headers, where header name is used as key and header value as value.</td>
</tr>
<tr>
<td>history</td>
<td>string</td>
<td>A time unit of how long the history data should be stored. Also accepts user macro and LLD macro.</td>
</tr>
<tr>
<td>history</td>
<td>string</td>
<td>Default: 90d.</td>
</tr>
<tr>
<td>http_proxy</td>
<td>string</td>
<td>HTTP agent item prototype field. HTTP(S) proxy connection string.</td>
</tr>
<tr>
<td>ipmi_sensor</td>
<td>string</td>
<td>IPMI sensor. Used only by IPMI item prototypes.</td>
</tr>
<tr>
<td>jmx_endpoint</td>
<td>string</td>
<td>JMX agent custom connection string.</td>
</tr>
<tr>
<td>logtimefmt</td>
<td>string</td>
<td>Format of the time in log entries. Used only by log item prototypes. Master item ID.</td>
</tr>
<tr>
<td>master_itemid</td>
<td>integer</td>
<td>Recursion up to 3 dependent items and item prototypes and maximum count of dependent items and item prototypes equal to 29999 are allowed.</td>
</tr>
<tr>
<td>output_format</td>
<td>integer</td>
<td>HTTP agent item prototype field. Should response be converted to JSON.</td>
</tr>
<tr>
<td>output_format</td>
<td>integer</td>
<td>0 - (default) Store raw. 1 - Convert to JSON.</td>
</tr>
<tr>
<td>params</td>
<td>string</td>
<td>Additional parameters depending on the type of the item prototype: executed script for SSH and Telnet item prototypes; SQL query for database monitor item prototypes; formula for calculated item prototypes.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>parameters</td>
<td>array</td>
<td>Additional parameters for script item prototypes. Array of objects with</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘name’ and ‘value’ properties, where name must be unique.</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>Password for authentication. Used by simple check, SSH, Telnet, database</td>
</tr>
<tr>
<td></td>
<td></td>
<td>monitor, JMX and HTTP agent item prototypes.</td>
</tr>
<tr>
<td>post_type</td>
<td>integer</td>
<td>HTTP agent item prototype field. Type of post data body stored in posts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>property.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Raw data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - JSON data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - XML data.</td>
</tr>
<tr>
<td>posts</td>
<td>string</td>
<td>HTTP agent item prototype field. HTTP(S) request body data. Used with</td>
</tr>
<tr>
<td></td>
<td></td>
<td>post_type.</td>
</tr>
<tr>
<td>privatekey</td>
<td>string</td>
<td>Name of the private key file.</td>
</tr>
<tr>
<td>publickey</td>
<td>string</td>
<td>Name of the public key file.</td>
</tr>
<tr>
<td>query_fields</td>
<td>array</td>
<td>HTTP agent item prototype field. Query parameters. Array of objects with</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘key’:‘value’ pairs, where value can be empty string.</td>
</tr>
<tr>
<td>request_method</td>
<td>integer</td>
<td>HTTP agent item prototype field. Type of request method.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) GET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - POST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - PUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - HEAD</td>
</tr>
<tr>
<td>retrieve_mode</td>
<td>integer</td>
<td>HTTP agent item prototype field. What part of response should be stored.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Headers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Both body and headers will be stored.</td>
</tr>
<tr>
<td>snmp_oid</td>
<td>string</td>
<td>SNMP OID.</td>
</tr>
<tr>
<td>ssl_cert_file</td>
<td>string</td>
<td>HTTP agent item prototype field. Public SSL Key file path.</td>
</tr>
<tr>
<td>ssl_key_file</td>
<td>string</td>
<td>HTTP agent item prototype field. Private SSL Key file path.</td>
</tr>
<tr>
<td>ssl_key_password</td>
<td>string</td>
<td>HTTP agent item prototype field. Password for SSL Key file.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Status of the item prototype.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) enabled item prototype;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - disabled item prototype;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - unsupported item prototype.</td>
</tr>
<tr>
<td>status_codes</td>
<td>string</td>
<td>HTTP agent item prototype field. Ranges of required HTTP status codes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>separated by commas. Also supports user macros or LLD macros as</td>
</tr>
<tr>
<td></td>
<td></td>
<td>part of comma separated list.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: 200,200-{$M},{$M},200-400</td>
</tr>
<tr>
<td>templateid</td>
<td>string</td>
<td>(readonly) ID of the parent template item prototype.</td>
</tr>
<tr>
<td>timeout</td>
<td>string</td>
<td>Item data polling request timeout. Used for HTTP agent and script item</td>
</tr>
<tr>
<td></td>
<td></td>
<td>prototypes. Supports user macros and LLD macros.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>default: 3s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>maximum value: 60s</td>
</tr>
<tr>
<td>trapper_hosts</td>
<td>string</td>
<td>Allowed hosts. Used by trapper item prototypes or HTTP item prototypes.</td>
</tr>
<tr>
<td>trends</td>
<td>string</td>
<td>A time unit of how long the trends data should be stored. Also accepts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>user macro and LLD macro.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 365d.</td>
</tr>
<tr>
<td>units</td>
<td>string</td>
<td>Value units.</td>
</tr>
<tr>
<td>username</td>
<td>string</td>
<td>Username for authentication. Used by simple check, SSH, Telnet, database</td>
</tr>
<tr>
<td></td>
<td></td>
<td>monitor, JMX and HTTP agent item prototypes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required by SSH and Telnet item prototypes.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Universal unique identifier, used for linking imported item prototypes to already existing ones. Used only for item prototypes on templates. Auto-generated, if not given. For update operations this field is readonly.</td>
</tr>
<tr>
<td>valuemapid</td>
<td>string</td>
<td>ID of the associated value map.</td>
</tr>
<tr>
<td>verify_host</td>
<td>integer</td>
<td>HTTP agent item prototype field. Validate host name in URL is in Common Name field or a Subject Alternate Name field of host certificate. 0 - (default) Do not validate. 1 - Validate.</td>
</tr>
<tr>
<td>verify_peer</td>
<td>integer</td>
<td>HTTP agent item prototype field. Validate host certificate authentic. 0 - (default) Do not validate. 1 - Validate.</td>
</tr>
<tr>
<td>discover</td>
<td>integer</td>
<td>Item prototype discovery status. Possible values: 0 - (default) new items will be discovered; 1 - new items will not be discovered and existing items will be marked as lost.</td>
</tr>
</tbody>
</table>

**Item prototype tag**

The item prototype tag object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Item prototype tag name. (required)</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Item prototype tag value.</td>
</tr>
</tbody>
</table>

**Item prototype preprocessing**

The item prototype preprocessing object has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>type</strong></td>
<td>integer</td>
<td>The preprocessing option type.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Custom multiplier;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Right trim;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - Left trim;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - Trim;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - Regular expression matching;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 - Boolean to decimal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 - Octal to decimal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 - Hexadecimal to decimal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 - Simple change;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 - Change per second;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 - XML XPath;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 - JSONPath;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 - In range;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14 - Matches regular expression;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15 - Does not match regular expression;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16 - Check for error in JSON;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17 - Check for error in XML;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18 - Check for error using regular expression;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19 - Discard unchanged;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 - Discard unchanged with heartbeat;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21 - JavaScript;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22 - Prometheus pattern;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23 - Prometheus to JSON;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24 - CSV to JSON;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25 - Replace;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26 - Check unsupported;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27 - XML to JSON.</td>
</tr>
<tr>
<td><strong>params</strong></td>
<td>string</td>
<td>Additional parameters used by preprocessing option. Multiple parameters are</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>separated by LF (\n) character.</td>
</tr>
<tr>
<td><strong>error_handler</strong></td>
<td>integer</td>
<td>Action type used in case of preprocessing step failure.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Error message is set by Zabbix server;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Discard value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Set custom value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - Set custom error message.</td>
</tr>
<tr>
<td><strong>error_handler_params</strong></td>
<td>string</td>
<td>Error handler parameters. Used with error_handler.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Must be empty, if error_handler is 0 or 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Can be empty if, error_handler is 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cannot be empty, if error_handler is 3.</td>
</tr>
</tbody>
</table>

The following parameters and error handlers are supported for each preprocessing type.

<table>
<thead>
<tr>
<th>Preprocessing type</th>
<th>Name</th>
<th>Parameter 1</th>
<th>Parameter 2</th>
<th>Parameter 3</th>
<th>Supported error handlers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Custom number(^1,^6)</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td></td>
<td>multiplier</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Right</td>
<td>list of</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>trim</td>
<td>characters(^2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Left</td>
<td>list of</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>trim</td>
<td>characters(^2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Trim</td>
<td>list of</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>characters(^2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preprocessing type</td>
<td>Name</td>
<td>Parameter 1</td>
<td>Parameter 2</td>
<td>Parameter 3</td>
<td>Supported error handlers</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------</td>
<td>----------------------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>5</td>
<td>Regular pattern expression</td>
<td>output</td>
<td>2</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Boolean to decimal</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Octal to decimal</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Hexadecimal to decimal</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Simple change</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Change per second</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>XML path</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>JSONPath path</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>In range</td>
<td>min, max</td>
<td>1, 6</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Matches pattern</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Does not match pattern</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Check for error in JSON path</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Check for error in XML path</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Check for error using regular expression</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>Preprocessing type</td>
<td>Name</td>
<td>Parameter 1</td>
<td>Parameter 2</td>
<td>Parameter 3</td>
<td>Supported error handlers</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------</td>
<td>------------------------------------------</td>
<td>------------------</td>
<td>-------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>19</td>
<td>Discard unchanged</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Discard unchanged with heartbeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>JavaScript script</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Prometheus pattern</td>
<td>value, label, function</td>
<td>output</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Prometheus pattern</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>CSV to JSON</td>
<td>character</td>
<td>character</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Replace search string</td>
<td>replacement</td>
<td></td>
<td>1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Check unsupported</td>
<td></td>
<td></td>
<td>1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>XML to JSON</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
</tbody>
</table>

1. integer or floating-point number
2. string
3. regular expression
4. JSONPath or XML XPath
5. positive integer (with support of time suffixes, e.g. 30s, 1m, 2h, 1d)
6. user macro, LLD macro
7. Prometheus pattern following the syntax: `<metric name>{<label name>="<label value>", ...} == <value>`. Each Prometheus pattern component (metric, label name, label value and metric value) can be user macro or LLD macro.
8. Prometheus output following the syntax: `<label name>` (can be a user macro or an LLD macro) if label is selected as the second parameter.
9. One of the aggregation functions: sum, min, max, avg, count if function is selected as the second parameter.

**itemprototype.create**

Description

**object itemprototype.create(object/array itemPrototypes)**

This method allows to create new item prototypes.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See **User roles** for more information.

Parameters

(object/array) Item prototype to create.

Additionally to the **standard item prototype properties**, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ruleid</td>
<td>string</td>
<td>ID of the LLD rule that the item belongs to.</td>
</tr>
<tr>
<td>preprocessing</td>
<td>array</td>
<td>Item prototype preprocessing options.</td>
</tr>
<tr>
<td>tags</td>
<td>array</td>
<td>Item prototype tags.</td>
</tr>
</tbody>
</table>
Return values

(object) Returns an object containing the IDs of the created item prototypes under the `items` property. The order of the returned IDs matches the order of the passed item prototypes.

Examples

Creating an item prototype

Create an item prototype to monitor free disc space on a discovered file system. Discovered items should be numeric Zabbix agent items updated every 30 seconds.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "name": "Free disk space on {#FSNAME}" ,
        "key": "vfs.fs.size[{#FSNAME},free]",
        "hostid": "10197",
        "ruleid": "27665",
        "type": 0,
        "value_type": 3,
        "interfaceid": "112",
        "delay": "30s"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "items": [
            "27666"
        ]
    },
    "id": 1
}
```

Creating an item prototype with preprocessing

Create an item using change per second and a custom multiplier as a second step.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "name": "Incoming network traffic on {#IFNAME}" ,
        "key": "net.if.in[{#IFNAME}]",
        "hostid": "10001",
        "ruleid": "27665",
        "type": 0,
        "value_type": 3,
        "delay": "60s",
        "units": "bps",
        "interfaceid": "1155",
        "preprocessing": [
            {
                "type": 10,
                "params": "",
                "error_handler": 0,
                "error_handler_params": ""
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```
Creating dependent item prototype

Create Dependent item prototype for Master item prototype with ID 44211. Only dependencies on same host (template/discovery rule) are allowed, therefore Master and Dependent item should have same hostid and ruleid.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "hostid": "10001",
        "ruleid": "27665",
        "name": "Dependent test item prototype",
        "key_": "dependent.prototype",
        "type": 18,
        "master_itemid": "44211",
        "value_type": 3
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44212"
        ]
    },
    "id": 1
}
```

Create HTTP agent item prototype

Create item prototype with URL using user macro, query fields and custom headers.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44212"
        ]
    },
    "id": 1
}
```
"params": {
    "type": "19",
    "hostid": "10254",
    "ruleid": "28256",
    "interfaceid": "2",
    "name": "api item prototype example",
    "key_": "api_http_item",
    "value_type": 3,
    "url": "${URL_PROTOTYPE}",
    "query_fields": [
        {
            "min": "10"
        },
        {
            "max": "100"
        }
    ],
    "headers": {
        "X-Source": "api"
    },
    "delay": "35"
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28305"
        ]
    },
    "id": 1
}

Create script item prototype

Create a simple data collection using a script item prototype.

Request:
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "name": "Script example",
        "key_": "custom.script.itemprototype",
        "hostid": "12345",
        "type": 21,
        "value_type": 4,
        "params": "var request = new HttpRequest();\nreturn request.post("https://postman-echo.com/post","${HOST.CONN});",
        "parameters": [
            {
                "name": "host",
                "value": "${HOST.CONN}"
            }
        ],
        "timeout": "6s",
        "delay": "30s"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
itemprototype.delete

Description

object itemprototype.delete(array itemPrototypeIds)

This method allows to delete item prototypes.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the item prototypes to delete.

Return values

(object) Returns an object containing the IDs of the deleted item prototypes under the prototypeIds property.

Examples

Deleting multiple item prototypes

Delete two item prototypes.

Dependent item prototypes are removed automatically if master item or item prototype is deleted.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.delete",
    "params": ["27352", "27356"],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "prototypeIds": ["27352", "27356"],
    },
    "id": 1
}
```
**itemprototype.get**

**Description**

integer/array itemprototype.get(object parameters)

The method allows to retrieve item prototypes according to the given parameters. This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>discoveryids</td>
<td>string/array</td>
<td>Return only item prototypes that belong to the given LLD rules.</td>
</tr>
<tr>
<td>graphids</td>
<td>string/array</td>
<td>Return only item prototypes that are used in the given graph prototypes.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only item prototypes that belong to the given hosts.</td>
</tr>
<tr>
<td>inherited</td>
<td>boolean</td>
<td>If set to true return only item prototypes inherited from a template.</td>
</tr>
<tr>
<td>itemids</td>
<td>string/array</td>
<td>Return only item prototypes with the given IDs.</td>
</tr>
<tr>
<td>monitored</td>
<td>boolean</td>
<td>If set to true return only enabled item prototypes that belong to monitored hosts.</td>
</tr>
<tr>
<td>templated</td>
<td>boolean</td>
<td>If set to true return only item prototypes that belong to templates.</td>
</tr>
<tr>
<td>templateids</td>
<td>string/array</td>
<td>Return only item prototypes that belong to the given templates.</td>
</tr>
<tr>
<td>triggerids</td>
<td>string/array</td>
<td>Return only item prototypes that are used in the given trigger prototypes.</td>
</tr>
<tr>
<td>selectDiscoveryRule</td>
<td>query</td>
<td>Return a discoveryRule property with the low-level discovery rule that the item prototype belongs to.</td>
</tr>
<tr>
<td>selectGraphs</td>
<td>query</td>
<td>Return a graph property with graph prototypes that the item prototype is used in. Supports count.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return a hosts property with an array of hosts that the item prototype belongs to.</td>
</tr>
<tr>
<td>selectTags</td>
<td>query</td>
<td>Return the item prototype tags in tags property.</td>
</tr>
<tr>
<td>selectTriggers</td>
<td>query</td>
<td>Return a triggers property with trigger prototypes that the item prototype is used in. Supports count.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>selectPreprocessing</td>
<td>query</td>
<td>Return a <code>preprocessing</code> property with item preprocessing options. It has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>type</code> - (string) The preprocessing option type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Custom multiplier;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Right trim;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - Left trim;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - Trim;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - Regular expression matching;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 - Boolean to decimal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 - Octal to decimal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 - Hexadecimal to decimal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 - Simple change;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 - Change per second;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 - XML XPath;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 - JSON Path;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 - In range;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14 - Matches regular expression;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15 - Does not match regular expression;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16 - Check for error in JSON;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17 - Check for error in XML;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18 - Check for error using regular expression;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19 - Discard unchanged;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 - Discard unchanged with heartbeat;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21 - JavaScript;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22 - Prometheus pattern;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23 - Prometheus to JSON;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24 - CSV to JSON;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25 - Replace;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26 - Check for not supported value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27 - XML to JSON.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>params</code> - (string) Additional parameters used by preprocessing option. Multiple parameters are separated by LF (<code>\n</code>) character.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>error_handler</code> - (string) Action type used in case of preprocessing step failure:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Error message is set by Zabbix server;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Discard value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Set custom value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - Set custom error message.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>error_handler_params</code> - (string) Error handler parameters.</td>
</tr>
<tr>
<td>selectValueMap</td>
<td>query</td>
<td>Return a <code>valuemap</code> property with item prototype value map.</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td>Return only those results that exactly match the given filter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supports additional filters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>host</code> - technical name of the host that the item prototype belongs to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Limits the number of records returned by subselects.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Applies to the following subselects:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>selectGraphs</code> - results will be sorted by name;</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>selectTriggers</code> - results will be sorted by description.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are: <code>itemid</code>, <code>name</code>, <code>key_</code>, <code>delay</code>, <code>type</code> and <code>status</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving item prototypes from an LLD rule

Retrieve all item prototypes for specific LLD rule ID.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "itemprototype.get",
    "params": {
        "output": "extend",
        "discoveryids": "27426"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [ { 
        "itemid": "23077",
        "type": "0",
        "snmp_oid": "",
        "hostid": "10079",
        "name": "Incoming network traffic on en0",
        "key_": "net.if.in[en0]",
        "delay": "1m",
        "history": "1w",
        "trends": "365d",
        "status": "0",
        "value_type": "3",
        "trapper_hosts": "",
        "units": "bps",
        "logtimefmt": "",
        "templateid": "0",
        "valuemapid": "0",
        "params": "",
        "ipmi_sensor": "",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
```
"itemid": "10010",
"type": "0",
"snmp_oid": "",
"hostid": "10001",
"name": "Processor load (1 min average per core)",
"key_": "system.cpu.load[percpu,avg1]",
"delay": "1m",
"history": "1w",
"trends": "365d",
"status": "0",
"value_type": "0",
"trapper_hosts": "",
"units": "",
"logtimefmt": "",
"templateid": "0",
"valuemapid": "0",
"params": "",
"ipmi_sensor": "",
"authtype": "0",
"username": "",
"password": "",
"publickey": "",
"privatekey": "",
"interfaceid": "0",
"description": "The processor load is calculated as system CPU load divided by number of CPU cores.",
"evaltype": "0",
"jmx_endpoint": "",
"master_itemid": "0",
"timeout": "3s",
"url": "",
"query_fields": []
"posts": "",
"status_codes": "200",
"follow_redirects": "1",
"url": "",
"query_fields": []
"posts": "",
"status_codes": "200",
"follow_redirects": "1"
Finding dependent item

Find one Dependent item for specific item ID.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "item.get",
  "params": {
    "output": "extend",
    "filter": {
      "type": 18,
      "master_itemid": "25545"
    },
    "limit": "1"
  },
  "auth": "038e1d7b1735c6a5436ee95879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "itemid": "25547",
      "type": "18",
      "snmp_oid": "",
      "hostid": "10116",
      "name": "Seconds",
      "key_": "apache.status.uptime.seconds",
      "delay": "0",
      "history": "90d",
      "trends": "365d",
      "status": "0",
      "value_type": "3",
      "trapper_hosts": "",
      "units": "",
      "logtimefmt": "",
      "templateid": "0",
      "valuemapid": "0",
      "params": "",
      "ipmi_sensor": ""
    }
  ]
}
```
Find HTTP agent item prototype

Find HTTP agent item prototype with request method HEAD for specific host ID.

Request:

```
{  
  "jsonrpc": "2.0",
  "method": "itemprototype.get",
  "params": {
    "hostids": "10254",
    "filter": {
      "type": 19,
      "request_method": 3
    }
  
  },
  "id": 17,
  "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

Response:

```
{  
  "jsonrpc": "2.0",
  "result": [
    {  
      "itemid": "28257",
      "type": "19",
      "snmp_oid": ""
    }
  ]
}
```
See also

- Host
- Graph prototype
- Trigger prototype

Source

CItemPrototype::get() in ui/include/classes/api/services/CItemPrototype.php.

itemprototype.update
Description

`object itemprototype.update(object/array itemPrototypes)`

This method allows to update existing item prototypes.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Item prototype properties to be updated.

The `itemid` property must be defined for each item prototype, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard item prototype properties, the method accepts the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>preprocessing</td>
<td>array</td>
<td>Item prototype <code>preprocessing</code> options to replace the current preprocessing options.</td>
</tr>
<tr>
<td>tags</td>
<td>array</td>
<td>Item prototype <code>tags</code>.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated item prototypes under the `itemids` property.

Examples

Changing the interface of an item prototype

Change the host interface that will be used by discovered items.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid": "27428",
        "interfaceid": "132"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": ["27428"
    ],
    "id": 1
}
```

Update dependent item prototype

Update Dependent item prototype with new Master item prototype ID. Only dependencies on same host (template/discovery rule) are allowed, therefore Master and Dependent item should have same hostid and ruleid.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "master_itemid": "25570",
        "itemid": "189030"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```
Update HTTP agent item prototype

Change query fields and remove all custom headers.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid": "28305",
        "query_fields": [
            {
                "random": "qwertyuiopasdfghjklzxcvbnm"
            }
        ],
        "headers": []
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28305"
        ],
        "id": 1
    }
}
```

Updating item preprocessing options

Update an item prototype with item preprocessing rule "Custom multiplier".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid": "44211",
        "preprocessing": [
            {
                "type": 1,
                "params": "4",
                "error_handler": 2,
                "error_handler_params": "5"
            }
        ],
        "headers": []
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ],
        "id": 1
    }
}
```
Updating a script item prototype

Update a script item prototype with a different script and remove unnecessary parameters that were used by previous script.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid": "23865",
        "parameters": [],
        "script": "Zabbix.log(3, 'Log test');\nreturn 1;"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": ["44211"
    ],
    "id": 1
}
```

Source

CItemPrototype::update() in ui/include/classes/api/services/CItemPrototype.php.

**LLD rule**

This class is designed to work with low level discovery rules.

Object references:

- **LLD rule**

Available methods:

- **discoveryrule.copy** - copying LLD rules
- **discoveryrule.create** - creating new LLD rules
- **discoveryrule.delete** - deleting LLD rules
The following objects are directly related to the discoveryrule API.

LLD rule

The low-level discovery rule object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemid</td>
<td>string</td>
<td>(readonly) ID of the LLD rule.</td>
</tr>
<tr>
<td>delay</td>
<td>string</td>
<td>Update interval of the LLD rule. Accepts seconds or time unit with suffix and with or without one or more custom intervals that consist of either flexible intervals and scheduling intervals as serialized strings. Also accepts user macros. Flexible intervals could be written as two macros separated by a forward slash. Intervals are separated by a semicolon.</td>
</tr>
<tr>
<td>hostid</td>
<td>string</td>
<td>ID of the host that the LLD rule belongs to.</td>
</tr>
<tr>
<td>interfaceid</td>
<td>string</td>
<td>ID of the LLD rule’s host interface. Used only for host LLD rules.</td>
</tr>
<tr>
<td>key_</td>
<td>string</td>
<td>LLD rule key.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the LLD rule.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>Type of the LLD rule. Possible values: 0 - Zabbix agent; 2 - Zabbix trapper; 3 - simple check; 5 - Zabbix internal; 7 - Zabbix agent (active); 10 - external check; 11 - database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent; 16 - JMX agent; 18 - Dependent item; 19 - HTTP agent; 20 - SNMP agent; 21 - Script.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>URL string, required for HTTP agent LLD rule. Supports user macros, {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST:HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}.</td>
</tr>
<tr>
<td>allow_traps</td>
<td>integer</td>
<td>HTTP agent LLD rule field. Allow to populate value as in trapper item type also. 0 - (default) Do not allow to accept incoming data. 1 - Allow to accept incoming data.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>authtype</td>
<td>integer</td>
<td>Used only by SSH agent or HTTP agent LLD rules.</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Description of the LLD rule.</td>
</tr>
<tr>
<td>error</td>
<td>string</td>
<td>(readonly) Error text if there are problems updating the LLD rule.</td>
</tr>
<tr>
<td>follow_redirects</td>
<td>integer</td>
<td>HTTP agent LLD rule field. Follow response redirects while pooling data.</td>
</tr>
<tr>
<td>headers</td>
<td>object</td>
<td>HTTP agent LLD rule field. Object with HTTP(S) request headers, where</td>
</tr>
<tr>
<td></td>
<td></td>
<td>header name is used as key and header value as value.</td>
</tr>
<tr>
<td>http_proxy</td>
<td>string</td>
<td>HTTP agent LLD rule field. HTTP(S) proxy connection string.</td>
</tr>
<tr>
<td>ipmi_sensor</td>
<td>string</td>
<td>IPMI sensor. Used only by IPMI LLD rules.</td>
</tr>
<tr>
<td>jmx_endpoint</td>
<td>string</td>
<td>JMX agent custom connection string.</td>
</tr>
<tr>
<td>lifetime</td>
<td>string</td>
<td>Time period after which items that are no longer discovered will be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>deleted. Accepts seconds, time unit with suffix and user macro.</td>
</tr>
<tr>
<td>master_itemid</td>
<td>integer</td>
<td>Master item ID. Recursion up to 3 dependent items and maximum count of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dependent items equal to 999 are allowed. Discovery rule cannot be master</td>
</tr>
<tr>
<td></td>
<td></td>
<td>item for another discovery rule.</td>
</tr>
<tr>
<td>output_format</td>
<td>integer</td>
<td>HTTP agent LLD rule field. Should response be converted to JSON.</td>
</tr>
<tr>
<td>params</td>
<td>string</td>
<td>Additional parameters depending on the type of the LLD rule:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- executed script for SSH and Telnet LLD rules;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- SQL query for database monitor LLD rules;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- formula for calculated LLD rules.</td>
</tr>
<tr>
<td>parameters</td>
<td>array</td>
<td>Additional parameters for script type LLD rule. Array of objects with</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'name' and 'value' properties, where name must be unique.</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>Password for authentication. Used by simple check, SSH, Telnet, database</td>
</tr>
<tr>
<td></td>
<td></td>
<td>monitor, JMX and HTTP agent LLD rules.</td>
</tr>
<tr>
<td>post_type</td>
<td>integer</td>
<td>HTTP agent LLD rule field. Type of post data stored in posts property.</td>
</tr>
<tr>
<td>posts</td>
<td>string</td>
<td>HTTP agent LLD rule field. HTTP(S) request body data. Used with post_type.</td>
</tr>
<tr>
<td>privatekey</td>
<td>string</td>
<td>Name of the private key file.</td>
</tr>
<tr>
<td>publickey</td>
<td>string</td>
<td>Name of the public key file.</td>
</tr>
<tr>
<td>query_fields</td>
<td>array</td>
<td>HTTP agent LLD rule field. Query parameters. Array of objects with key:'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value' pairs, where value can be empty string.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>request_method</td>
<td>integer</td>
<td>HTTP agent LLD rule field. Type of request method.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) GET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - POST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - PUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - HEAD</td>
</tr>
<tr>
<td>retrieve_mode</td>
<td>integer</td>
<td>HTTP agent LLD rule field. What part of response should be stored.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Body.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Headers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Both body and headers will be stored.</td>
</tr>
<tr>
<td>snmp_oid</td>
<td>string</td>
<td>SNMP OID.</td>
</tr>
<tr>
<td>ssl_cert_file</td>
<td>string</td>
<td>HTTP agent LLD rule field. Public SSL Key file path.</td>
</tr>
<tr>
<td>ssl_key_file</td>
<td>string</td>
<td>HTTP agent LLD rule field. Private SSL Key file path.</td>
</tr>
<tr>
<td>ssl_key_password</td>
<td>string</td>
<td>HTTP agent LLD rule field. Password for SSL Key file.</td>
</tr>
<tr>
<td>state</td>
<td>integer</td>
<td>(readonly) State of the LLD rule.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) normal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - not supported.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Status of the LLD rule.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) enabled LLD rule;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - disabled LLD rule.</td>
</tr>
<tr>
<td>status_codes</td>
<td>string</td>
<td>HTTP agent LLD rule field. Ranges of required HTTP status codes separated by commas. Also supports user macros as part of comma separated list.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: 200,200-{$M},{$M},200-400</td>
</tr>
<tr>
<td>templateid</td>
<td>string</td>
<td>(readonly) ID of the parent template LLD rule.</td>
</tr>
<tr>
<td>timeout</td>
<td>string</td>
<td>Item data polling request timeout. Used for HTTP agent and script LLD rules. Supports user macros.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>default: 3s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>maximum value: 60s</td>
</tr>
<tr>
<td>trapper_hosts</td>
<td>string</td>
<td>Allowed hosts. Used by trapper LLD rules or HTTP agent LLD rules.</td>
</tr>
<tr>
<td>username</td>
<td>string</td>
<td>Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent LLD rules.</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Universal unique identifier, used for linking imported LLD rules to already existing ones. Used only for LLD rules on templates. Auto-generated, if not given.</td>
</tr>
<tr>
<td>verify_host</td>
<td>integer</td>
<td>HTTP agent LLD rule field. Validate host name in URL is in Common Name field or a Subject Alternate Name field of host certificate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Do not validate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Validate.</td>
</tr>
<tr>
<td>verify_peer</td>
<td>integer</td>
<td>HTTP agent LLD rule field. Validate is host certificate authentic.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Do not validate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Validate.</td>
</tr>
</tbody>
</table>

**LLD rule filter**

The LLD rule filter object defines a set of conditions that can be used to filter discovered objects. It has the following properties:
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conditions</td>
<td>array</td>
<td>Set of filter conditions to use for filtering results.</td>
</tr>
<tr>
<td>evaltype</td>
<td>integer</td>
<td>Filter condition evaluation method.</td>
</tr>
<tr>
<td>eval_formula</td>
<td>string</td>
<td>(readonly) Generated expression that will be used for evaluating filter conditions. The expression contains IDs that reference specific filter conditions by its formulaid. The value of eval_formula is equal to the value of formula for filters with a custom expression.</td>
</tr>
<tr>
<td>formula</td>
<td>string</td>
<td>User-defined expression to be used for evaluating conditions of filters with a custom expression. The expression must contain IDs that reference specific filter conditions by its formulaid. The IDs used in the expression must exactly match the ones defined in the filter conditions: no condition can remain unused or omitted. Required for custom expression filters.</td>
</tr>
</tbody>
</table>

**LLD rule filter condition**

The LLD rule filter condition object defines a separate check to perform on the value of an LLD macro. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>macro</td>
<td>string</td>
<td>LLD macro to perform the check on.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Value to compare with.</td>
</tr>
<tr>
<td>formulaid</td>
<td>string</td>
<td>Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.</td>
</tr>
<tr>
<td>operator</td>
<td>integer</td>
<td>Condition operator.</td>
</tr>
</tbody>
</table>

Possible values:
- 8 - (default) matches regular expression;
- 9 - does not match regular expression;
- 12 - exists;
- 13 - does not exist.

To better understand how to use filters with various types of expressions, see examples on the discoveryrule.get and discoveryrule.create method pages.

**LLD macro path**

The LLD macro path has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lld_macro</td>
<td>string</td>
<td>LLD macro.</td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td>Selector for value which will be assigned to corresponding macro.</td>
</tr>
</tbody>
</table>

**LLD rule preprocessing**

The LLD rule preprocessing object has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>type</strong></td>
<td>integer</td>
<td>The preprocessing option type.</td>
</tr>
<tr>
<td><strong>params</strong></td>
<td>string</td>
<td>Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n) character.</td>
</tr>
<tr>
<td><strong>error_handler</strong></td>
<td>integer</td>
<td>Action type used in case of preprocessing step failure.</td>
</tr>
<tr>
<td><strong>error_handler_params</strong></td>
<td>string</td>
<td>Error handler parameters. Used with error_handler. Must be empty, if error_handler is 0 or 1. Can be empty if, error_handler is 2. Cannot be empty, if error_handler is 3.</td>
</tr>
</tbody>
</table>

The following parameters and error handlers are supported for each preprocessing type.

<table>
<thead>
<tr>
<th>Preprocessing type</th>
<th>Name</th>
<th>Parameter 1</th>
<th>Parameter 2</th>
<th>Parameter 3</th>
<th>Supported error handlers</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Regular expression</td>
<td>output</td>
<td>output</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>XML XPath</td>
<td>path</td>
<td>path</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>JSONPath</td>
<td>path</td>
<td>path</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Does not match regular expression</td>
<td>pattern</td>
<td>pattern</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Check for error in JSON</td>
<td>path</td>
<td>path</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Check for error in XML</td>
<td>path</td>
<td>path</td>
<td>0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>Preprocessing type</td>
<td>Name</td>
<td>Parameter 1</td>
<td>Parameter 2</td>
<td>Parameter 3</td>
<td>Supported error handlers</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>20</td>
<td>Discard seconds(^4,5,6) unchanged with heartbeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Prometheus pattern(^5,7) to JSON</td>
<td></td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>24</td>
<td>CSV to JSON pattern(^5,7) character(^2) character(^2)</td>
<td>character(^2)</td>
<td>0,1</td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>25</td>
<td>Replace search string(^2) replacement(^2) to JSON</td>
<td>replacement(^2)</td>
<td></td>
<td></td>
<td>0, 1, 2, 3</td>
</tr>
<tr>
<td>27</td>
<td>XML to JSON</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. regular expression
2. string
3. JSONPath or XML XPath
4. positive integer (with support of time suffixes, e.g. 30s, 1m, 2h, 1d)
5. user macro
6. LLD macro
7. Prometheus pattern following the syntax: `<metric name>{<label name>="<label value>", ...} == <value>`. Each Prometheus pattern component (metric, label name, label value and metric value) can be user macro.
8. Prometheus output following the syntax: `<label name>`.

LLD rule overrides

The LLD rule overrides object defines a set of rules (filters, conditions and operations) that are used to override properties of different prototype objects. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name (required)</td>
<td>string</td>
<td>Unique override name.</td>
</tr>
<tr>
<td>step (required)</td>
<td>integer</td>
<td>Unique order number of the override.</td>
</tr>
<tr>
<td>stop</td>
<td>integer</td>
<td>Stop processing next overrides if matches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) don’t stop processing overrides;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - stop processing overrides if filter matches.</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td>Override filter.</td>
</tr>
<tr>
<td>operations</td>
<td>array</td>
<td>Override operations.</td>
</tr>
</tbody>
</table>

LLD rule override filter

The LLD rule override filter object defines a set of conditions that if they match the discovered object the override is applied. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>evaltype (required)</td>
<td>integer</td>
<td>Override filter condition evaluation method.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - and/or;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - and;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - or;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - custom expression.</td>
</tr>
<tr>
<td>conditions (required)</td>
<td>array</td>
<td>Set of override filter conditions to use for matching the discovered objects.</td>
</tr>
</tbody>
</table>
**Property** | **Type** | **Description**
--- | --- | ---
**eval_formula** | string | (readonly) Generated expression that will be used for evaluating override filter conditions. The expression contains IDs that reference specific override filter conditions by its `formulaid`. The value of `eval_formula` is equal to the value of `formula` for filters with a custom expression.

**formula** | string | User-defined expression to be used for evaluating conditions of override filters with a custom expression. The expression must contain IDs that reference specific override filter conditions by its `formulaid`. The IDs used in the expression must exactly match the ones defined in the override filter conditions: no condition can remain unused or omitted.

Required for custom expression override filters.

**LLD rule override filter condition**

The LLD rule override filter condition object defines a separate check to perform on the value of an LLD macro. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>macro</strong></td>
<td>string</td>
<td>LLD macro to perform the check on. (required)</td>
</tr>
<tr>
<td><strong>value</strong></td>
<td>string</td>
<td>Value to compare with. (required)</td>
</tr>
<tr>
<td><strong>formulaid</strong></td>
<td>string</td>
<td>Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.</td>
</tr>
<tr>
<td><strong>operator</strong></td>
<td>integer</td>
<td>Condition operator. Possible values: 8 - (default) matches regular expression; 9 - does not match regular expression; 12 - exists; 13 - does not exist.</td>
</tr>
</tbody>
</table>

**LLD rule override operation**

The LLD rule override operation is combination of conditions and actions to perform on the prototype object. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>operationobject</strong></td>
<td>integer</td>
<td>Type of discovered object to perform the action. (required) Possible values: 0 - Item prototype; 1 - Trigger prototype; 2 - Graph prototype; 3 - Host prototype.</td>
</tr>
<tr>
<td><strong>operator</strong></td>
<td>integer</td>
<td>Override condition operator. Possible values: 0 - (default) equals; 1 - does not equal; 2 - contains; 3 - does not contain; 8 - matches; 9 - does not match.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Pattern to match item, trigger, graph or host prototype name depending on selected object.</td>
</tr>
<tr>
<td>opstatus</td>
<td>object</td>
<td>Override operation status object for item, trigger and host prototype objects.</td>
</tr>
<tr>
<td>opdiscover</td>
<td>object</td>
<td>Override operation discover status object (all object types).</td>
</tr>
<tr>
<td>opperiod</td>
<td>object</td>
<td>Override operation period (update interval) object for item prototype object.</td>
</tr>
<tr>
<td>ophistory</td>
<td>object</td>
<td>Override operation history object for item prototype object.</td>
</tr>
<tr>
<td>optrends</td>
<td>object</td>
<td>Override operation trends object for item prototype object.</td>
</tr>
<tr>
<td>opseverity</td>
<td>object</td>
<td>Override operation severity object for trigger prototype object.</td>
</tr>
<tr>
<td>optag</td>
<td>array</td>
<td>Override operation tag object for trigger and host prototype objects.</td>
</tr>
<tr>
<td>optemplate</td>
<td>array</td>
<td>Override operation template object for host prototype object.</td>
</tr>
<tr>
<td>opinventory</td>
<td>object</td>
<td>Override operation inventory object for host prototype object.</td>
</tr>
</tbody>
</table>

**LLD rule override operation status**

LLD rule override operation status that is set to discovered object. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>integer</td>
<td>Override the status for selected object.</td>
</tr>
</tbody>
</table>

Possible values:
- 0 - Create enabled;
- 1 - Create disabled.

**LLD rule override operation discover**

LLD rule override operation discover status that is set to discovered object. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>discover</td>
<td>integer</td>
<td>Override the discover status for selected object.</td>
</tr>
</tbody>
</table>

Possible values:
- 0 - Yes, continue discovering the objects;
- 1 - No, new objects will not be discovered and existing ones will be marked as lost.

**LLD rule override operation period**

LLD rule override operation period is an update interval value (supports custom intervals) that is set to discovered item. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>delay</td>
<td>string</td>
<td>Override the update interval of the item prototype. Accepts seconds or a time unit with suffix (30s,1m,2h,1d) as well as flexible and scheduling intervals and user macros or LLD macros. Multiple intervals are separated by a semicolon.</td>
</tr>
</tbody>
</table>

**LLD rule override operation history**

LLD rule override operation history value that is set to discovered item. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>history</td>
<td>string</td>
<td>Override the history of item prototype which is a time unit of how long the history data should be stored. Also accepts user macro and LLD macro.</td>
</tr>
</tbody>
</table>
LLD rule override operation trends

LLD rule override operation trends value that is set to discovered item. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trends</td>
<td>string</td>
<td>Overridethe trends of item prototype which is a time unit of how long the trends data should be stored. Also accepts user macro and LLD macro.</td>
</tr>
</tbody>
</table>

LLD rule override operation severity

LLD rule override operation severity value that is set to discovered trigger. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>severity</td>
<td>integer</td>
<td>Override the severity of trigger prototype.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are: 0 - (default) not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.</td>
</tr>
</tbody>
</table>

LLD rule override operation tag

LLD rule override operation tag object contains tag name and value that are set to discovered object. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>New tag name.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>New tag value.</td>
</tr>
</tbody>
</table>

LLD rule override operation template

LLD rule override operation template object that is linked to discovered host. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>templateid</td>
<td>string</td>
<td>Override the template of host prototype linked templates.</td>
</tr>
</tbody>
</table>

LLD rule override operation inventory

LLD rule override operation inventory mode value that is set to discovered host. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inventory_mode</td>
<td>integer</td>
<td>Override the host prototype inventory mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are: -1 - disabled; 0 - (default) manual; 1 - automatic.</td>
</tr>
</tbody>
</table>

discoveryrule.copy

Description

object discoveryrule.copy(object parameters)

This method allows to copy LLD rules with all of the prototypes to the given hosts.
This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the LLD rules to copy and the target hosts.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>discoveryids</td>
<td>array</td>
<td>IDs of the LLD rules to be copied.</td>
</tr>
<tr>
<td>hostids</td>
<td>array</td>
<td>IDs of the hosts to copy the LLD rules to.</td>
</tr>
</tbody>
</table>

Return values

(boolean) Returns true if the copying was successful.

Examples

Copy an LLD rule to multiple hosts

Copy an LLD rule to two hosts.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "discoveryrule.copy",
   "params": {
      "discoveryids": ["27426"],
      "hostids": ["10196", "10197"]
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": true,
   "id": 1
}
```

Source

CDiscoveryrule::copy() in ui/include/classes/api/services/CDiscoveryRule.php.

discoveryrule.create

Description

object discoveryrule.create(object/array lldRules)

This method allows to create new LLD rules.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) LLD rules to create.

Additionally to the standard LLD rule properties, the method accepts the following parameters.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>object</td>
<td>LLD rule filter object for the LLD rule.</td>
</tr>
<tr>
<td>preprocessing</td>
<td>array</td>
<td>LLD rule preprocessing options.</td>
</tr>
<tr>
<td>lld_macro_paths</td>
<td>array</td>
<td>LLD rule lld_macro_path options.</td>
</tr>
<tr>
<td>overrides</td>
<td>array</td>
<td>LLD rule overrides options.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the created LLD rules under the itemids property. The order of the returned IDs matches the order of the passed LLD rules.

Examples

Creating an LLD rule

Create a Zabbix agent LLD rule to discover mounted file systems. Discovered items will be updated every 30 seconds.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Mounted filesystem discovery",
        "key_": "vfs.fs.discovery",
        "hostid": "10197",
        "type": 0,
        "interfaceid": "112",
        "delay": "30s"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ],
        "id": 1
    }
}
```

Using a filter

Create an LLD rule with a set of conditions to filter the results by. The conditions will be grouped together using the logical "and" operator.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Filtered LLD rule",
        "key_": "lld",
        "hostid": "10116",
        "type": 0,
        "interfaceid": "13",
        "delay": "30s",
        "filter": {
            "evaltype": 1,
            "conditions": [
```
Response:
{
  "jsonrpc": "2.0",
  "result": {
    "itemids": [
      "27665"
    ],
  },
  "id": 1
}

Creating a LLD rule with macro paths

Request:
{
  "jsonrpc": "2.0",
  "method": "discoveryrule.create",
  "params": {
    "name": "LLD rule with LLD macro paths",
    "key_": "lld",
    "hostid": "10116",
    "type": 0,
    "interfaceid": "13",
    "delay": "30s",
    "lld_macro_paths": [
      {
        "lld_macro": "#{MACRO1}"
      },
      {
        "lld_macro": "#{MACRO2}"
      }
    ],
    "auth": "038e1d7b1735c6a5436ee9eae096879e",
    "id": 1
  }
}
Using a custom expression filter

Create an LLD rule with a filter that will use a custom expression to evaluate the conditions. The LLD rule must only discover objects the "{#MACRO1}" macro value of which matches both regular expression "regex1" and "regex2", and the value of "{#MACRO2}" matches either "regex3" or "regex4". The formula IDs "A", "B", "C" and "D" have been chosen arbitrarily.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "discoveryrule.create",
  "params": {
    "name": "Filtered LLD rule",
    "key_": "lld",
    "hostid": "10116",
    "type": 0,
    "interfaceid": "13",
    "delay": "30s",
    "filter": {
      "evaltype": 3,
      "formula": "(A and B) and (C or D)",
      "conditions": [
        {
          "macro": "{#MACRO1}",
          "value": "@regex1",
          "formulaid": "A"
        },
        {
          "macro": "{#MACRO1}",
          "value": "@regex2",
          "formulaid": "B"
        },
        {
          "macro": "{#MACRO2}",
          "value": "@regex3",
          "formulaid": "C"
        },
        {
          "macro": "{#MACRO2}",
          "value": "@regex4",
          "formulaid": "D"
        }
      ]
    }
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": {
    "itemids": [
      "27665"
    ],
    "id": 1
  }
}
```
Using custom query fields and headers

Create LLD rule with custom query fields and headers.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "hostid": "10257",
        "interfaceid": "5",
        "type": 19,
        "name": "API HTTP agent",
        "key_": "api_discovery_rule",
        "value_type": 3,
        "delay": "5s",
        "url": "http://127.0.0.1?discoverer.php",
        "query_fields": [
            {
                "mode": "json"
            },
            {
                "elements": "2"
            }
        ],
        "headers": {
            "X-Type": "api",
            "Authorization": "Bearer mF_A.B5f-2.1JcM"
        },
        "allow_traps": 1,
        "trapper_hosts": "127.0.0.1"
    },
    "auth": "d678e0b856688ce578ff061bd29a20d3b",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": ["28336"

    },
    "id": 35
}
```

Creating a LLD rule with preprocessing

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Discovery rule with preprocessing",
        "key_": "lld.with.preprocessing",
```
"hostid": "10001",
"ruleid": "27665",
"type": 0,
"value_type": 3,
"delay": "60s",
"interfaceid": "1155",
"preprocessing": [
  {
    "type": 20,
    "params": "20",
    "error_handler": 0,
    "error_handler_params": ""
  }
],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:

{
  "jsonrpc": "2.0",
  "result": {
    "itemids": [
      "44211"
    ],
  },
  "id": 1
}

Creating a LLD rule with overrides

Request:

{
  "jsonrpc": "2.0",
  "method": "discoveryrule.create",
  "params": {
    "name": "Discover database host",
    "key_": "lld.with.overrides",
    "hostid": "10001",
    "type": 0,
    "value_type": 3,
    "delay": "60s",
    "interfaceid": "1155",
    "overrides": [
      {
        "name": "Discover MySQL host",
        "step": "1",
        "stop": "1",
        "filter": {
          "evaltype": "2",
          "conditions": [
            {
              "macro": "{#UNIT.NAME}",
              "operator": "8",
              "value": "^mysqld\.service$"
            },
            {
              "macro": "{#UNIT.NAME}",
              "operator": "8",
              "value": "^mariadb\.service$"
            }
          ]
        }
      }
    ]
  }
}
"operations": [  
  {  
    "operationobject": "3",
    "operator": "2",
    "value": "Database host",
    "opstatus": {  
      "status": "0"
    },  
    "optemplate": [  
      {  
        "templateid": "10170"
      }
    ],  
    "optag": [  
      {  
        "tag": "Database",
        "value": "MySQL"
      }
    ]
  },  
  {  
    "name": "Discover PostgreSQL host",
    "step": "2",
    "stop": "1",
    "filter": {  
      "evaltype": "0",
      "conditions": [  
        {  
          "macro": "#{UNIT.NAME}^postgresql.service$",
          "operator": "8",
          "value": "postgresql\service$"
        }
      ]
    },  
    "operations": [  
      {  
        "operationobject": "3",
        "operator": "2",
        "value": "Database host",
        "opstatus": {  
          "status": "0"
        },  
        "optemplate": [  
          {  
            "templateid": "10263"
          }
        ],  
        "optag": [  
          {  
            "tag": "Database",
            "value": "PostgreSQL"
          }
        ]
      }
    ]
  }
],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
Create script LLD rule

Create a simple data collection using a script LLD rule.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Script example",
        "key_": "custom.script.lldrule",
        "hostid": "12345",
        "type": 21,
        "value_type": 4,
        "params": "var request = new HttpRequest();

        return request.post("https://postman-echo.com/post",
        JSON.parse(value));",
        "parameters": [{
            "name": "host",
            "value": "{HOST.CONN}"}
        ],
        "timeout": "6s",
        "delay": "30s"
    },
    "auth": "038e1d7b1735c6a5436ee9eae096879e",
    "id": 2
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 3
}
```

See also

- LLD rule filter
- LLD macro paths
- LLD rule preprocessing

Source

CDiscoveryRule::create() in ui/include/classes/api/services/CDiscoveryRule.php.
object discoveryrule.delete(array lldRuleIds)

This method allows to delete LLD rules.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(array) IDs of the LLD rules to delete.

Return values
(object) Returns an object containing the IDs of the deleted LLD rules under the itemids property.

Examples
Deleting multiple LLD rules
Delete two LLD rules.

Request:
{
   "jsonrpc": "2.0",
   "method": "discoveryrule.delete",
   "params": [
      "27665",
      "27668"
   ],
   "auth": "3a57200802b24cda67c4e4010b50c065",
   "id": 1
}

Response:
{
   "jsonrpc": "2.0",
   "result": {
      "ruleids": [
         "27665",
         "27668"
      ]
   },
   "id": 1
}

Source
CDiscoveryRule::delete() in ui/include/classes/api/services/CDiscoveryRule.php.

discoveryrule.get

Description
integer/array discoveryrule.get(object parameters)

The method allows to retrieve LLD rules according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemids</td>
<td>string/array</td>
<td>Return only LLD rules with the given IDs.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only LLD rules that belong to the hosts from the given groups.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only LLD rules that belong to the given hosts.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>inherited</td>
<td>boolean</td>
<td>If set to <code>true</code> return only LLD rules inherited from a template.</td>
</tr>
<tr>
<td>interfaceids</td>
<td>string/array</td>
<td>Return only LLD rules use the given host interfaces.</td>
</tr>
<tr>
<td>monitored</td>
<td>boolean</td>
<td>If set to <code>true</code> return only enabled LLD rules that belong to monitored hosts.</td>
</tr>
<tr>
<td>templated</td>
<td>boolean</td>
<td>If set to <code>true</code> return only LLD rules that belong to templates.</td>
</tr>
<tr>
<td>templateids</td>
<td>string/array</td>
<td>Return only LLD rules that belong to the given templates.</td>
</tr>
<tr>
<td>selectFilter</td>
<td>query</td>
<td>Return a <code>filter</code> property with data of the filter used by the LLD rule.</td>
</tr>
<tr>
<td>selectGraphs</td>
<td>query</td>
<td>Returns a <code>graphs</code> property with graph prototypes that belong to the LLD rule.</td>
</tr>
<tr>
<td>selectHostPrototypes</td>
<td>query</td>
<td>Return a <code>hostPrototypes</code> property with host prototypes that belong to the LLD rule.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return a <code>hosts</code> property with an array of hosts that the LLD rule belongs to.</td>
</tr>
<tr>
<td>selectItems</td>
<td>query</td>
<td>Return an <code>items</code> property with item prototypes that belong to the LLD rule.</td>
</tr>
<tr>
<td>selectTriggers</td>
<td>query</td>
<td>Return a <code>triggers</code> property with trigger prototypes that belong to the LLD rule.</td>
</tr>
<tr>
<td>selectLLDMacroPaths</td>
<td>query</td>
<td>Return an <code>lld_macro_paths</code> property with a list of LLD macros and paths to values assigned to each corresponding macro.</td>
</tr>
<tr>
<td>selectPreprocessing</td>
<td>query</td>
<td>Return a preprocessing property with LLD rule preprocessing options.</td>
</tr>
</tbody>
</table>

It has the following properties:

- **type** - (string) The preprocessing option type:
  - 5 - Regular expression matching;
  - 11 - XML XPath;
  - 12 - JSONPath;
  - 15 - Does not match regular expression;
  - 16 - Check for error in JSON;
  - 17 - Check for error in XML;
  - 20 - Discard unchanged with heartbeat;
  - 23 - Prometheus to JSON;
  - 24 - CSV to JSON;
  - 25 - Replace;
  - 27 - XML to JSON.

- **params** - (string) Additional parameters used by preprocessing option. Multiple parameters are separated by LF (`
`) character.
- **error_handler** - (string) Action type used in case of preprocessing step failure:
  - 0 - Error message is set by Zabbix server;
  - 1 - Discard value;
  - 2 - Set custom value;
  - 3 - Set custom error message.

- **error_handler_params** - (string) Error handler parameters.
- **selectOverrides** - query
  Return an `lld_rule_overrides` property with a list of override filters, conditions and operations that are performed on prototype objects.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>object</td>
<td>Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: host - technical name of the host that the LLD rule belongs to.</td>
</tr>
<tr>
<td>limitSelects</td>
<td>integer</td>
<td>Limits the number of records returned by subselects. Applies to the following subselects: selectItems; selectGraphs; selectTriggers.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties. Possible values are: itemid, name, key_, delay, type and status. These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values
(integer/array) Returns either:
- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples
Retrieving discovery rules from a host
Retrieve all discovery rules for specific host ID.
Request:
```json
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "output": "extend",
        "hostids": "10202"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```
Response:
```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "27425",
            "type": "0",
            "snmp_oid": ",
            "hostid": "10202",
```
"name": "Network interface discovery",
"key": "net.if.discovery",
"delay": "1h",
"status": "0",
"trapper_hosts": "",
"templateid": "22444",
"valuemapid": "0",
"params": "",
"ipmi_sensor": "",
"authtype": "0",
"username": "",
"password": "",
"publickey": "",
"privatekey": "",
"interfaceid": "119",
"description": "Discovery of network interfaces as defined in global regular expression "Network interfaces for discovery",
"lifetime": "30d",
"master_itemid": "0",
"timeout": "3s",
"url": "",
"query_fields": []
"posts": "",
"status_codes": "200",
"follow_redirects": "1",
"post_type": "0",
"http_proxy": "",
"headers": []
"retrieve_mode": "0",
"request_method": "0",
"ssl_cert_file": "",
"ssl_key_file": "",
"ssl_key_password": "",
"verify_peer": "0",
"verify_host": "0",
"allow_traps": "0",
"uuid": "",
"state": "0",
"error": "",
"parameters": []

"itemid": "27426",
"type": "0",
"snmp_oid": "",
"hostid": "10202",
"name": "Mounted filesystem discovery",
"key": "vfs.fs.discovery",
"delay": "1h",
"status": "0",
"trapper_hosts": "",
"templateid": "22450",
"valuemapid": "0",
"params": "",
"ipmi_sensor": "",
"authtype": "0",
"username": "",
"password": "",
"publickey": "",
"privatekey": "",
"interfaceid": "119",
"description": "Discovery of file systems of different types as defined in global regular expression "File systems for discovery"."}
Retrieving the name of the LLD rule "24681" and its filter conditions. The filter uses the "and" evaluation type, so the formula property is empty and eval_formula is generated automatically.

Request:
```
{
  "jsonrpc": "2.0",
  "method": "discoveryrule.get",
  "params": {
    "output": ["name"],
    "selectFilter": "extend",
    "itemids": ["24681"]
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:
```
{
  "jsonrpc": "2.0",
  "result": [ ]
  "itemid": "24681",
  "name": "Filtered LLD rule",
  "filter": {
    "evaltype": "1",
    "formula": "",
    "conditions": [
    {
      "macro": "{#MACRO1}",
      "value": "@regex1",
      "operator": "8",
      "formulaid": "A"
    }
  ]
```
Retrieve LLD rule by URL

Retrieve LLD rule for host by rule URL field value. Only exact match of URL string defined for LLD rule is supported.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "discoveryrule.get",
  "params": {
    "hostids": "10257",
    "filter": {
      "type": 19,
      "url": "http://127.0.0.1/discoverer.php"
    }
  },
  "id": 39,
  "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": [
    {
      "itemid": "28336",
      "type": "19",
      "snmp_oid": "",
      "hostid": "10257",
      "name": "API HTTP agent",
      "key_": "api_discovery_rule",
      "delay": "5s",
      "status": "0",
      "trapper_hosts": "",
      "templateid": "0",
      "valuemapid": "0",
      "params": ""
    }
  ]
}
```
Retrieve LLD rule with overrides

Retrieve one LLD rule that has various override settings.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "discoveryrule.get",
  "params": {
    "output": ["name"],
    "itemids": [30980],
    "select0overrides": ["name", "step", "stop", "filter", "operations"]
  },
  "id": 39,
  "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```
Response:

```json
{
   "jsonrpc": "2.0",
   "result": [
   {
      "name": "Discover database host",
      "overrides": [
      {
         "name": "Discover MySQL host",
         "step": "1",
         "stop": "1",
         "filter": {
            "evaltype": "2",
            "formula": "",
            "conditions": [
            {"macro": "#{UNIT.NAME}",
             "operator": "8",
             "value": "mysqld\.service$",
             "formulaid": "A"},
            {"macro": "#{UNIT.NAME}",
             "operator": "8",
             "value": "mariadb\service$",
             "formulaid": "B"}
         ],
         "eval_formula": "A or B",
         "operations": [
         {"operationobject": "3",
          "operator": "2",
          "value": "Database host",
          "opstatus": {
             "status": "0"
          },
          "optag": ["tag": "Database",
                     "value": "MySQL"
                    ],
          "optemplate": ["templateid": "10170"
                         ]
         }
      }
   ],
   "name": "Discover PostgreSQL host",
   "step": "2",
   "stop": "1",
   "filter": {
      "evaltype": "0",
      "formula": "",
      "conditions": [
      {"macro": "#{UNIT.NAME}"},
   ]
   }
   ]
   }
   ]
}
```
"operator": "8",
"value": "postgresql\service$",
"formulaid": "A"
],
"eval_formula": "A"
},
"operations": [
{
  "operationobject": "3",
  "operator": "2",
  "value": "Database host",
  "opstatus": {
    "status": "0"
  },
  "optag": [
    {
      "tag": "Database",
      "value": "PostgreSQL"
    }
  ],
  "optemplate": [
    {
      "templateid": "10263"
    }
  ]
}
],
"id": 39

See also
• Graph prototype
• Host
• Item prototype
• LLD rule filter
• Trigger prototype

Source
CDiscoveryRule::get() in ui/include/classes/api/services/CDiscoveryRule.php.

discoveryrule.update

Description
object discoveryrule.update(object/array lldRules)

This method allows to update existing LLD rules.
This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(object/array) LLD rule properties to be updated.
The itemid property must be defined for each LLD rule, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard LLD rule properties, the method accepts the following parameters.
### Parameter  |  Type  | Description
--- | --- | ---
filter  | object  | LLD rule filter object to replace the current filter.
preprocessing  | array  | LLD rule preprocessing options to replace the current preprocessing options.
lld_macro_paths  | array  | LLD rule lld_macro_path options.
overrides  | array  | LLD rule overrides options.

**Return values**

*(object)* Returns an object containing the IDs of the updated LLD rules under the *itemids* property.

**Examples**

Adding a filter to an LLD rule

Add a filter so that the contents of the `{#FSTYPE}` macro would match the @File systems for discovery regexp.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "22450",
        "filter": {
            "evaltype": 1,
            "conditions": [
                {
                    "macro": "{#FSTYPE}"
                }
            ],
            "value": "@File systems for discovery"
        }
    }
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22450"
        ],
        "id": 1
    }
}
```

Adding LLD macro paths

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "22450",
        "lld_macro_paths": [
            {
                "lld_macro": "{#MACRO1}"
            },
            {
                "path": ".json.path"
            }
        ]
    }
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}
```
"id": 1
}

Response:
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22450"
        ],
    },
    "id": 1
}

Disable trapping

Disable LLD trapping for discovery rule.

Request:
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "28336",
        "allow_traps": 0
    },
    "id": 36,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}

Response:
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28336"
        ],
    },
    "id": 36
}

Updating LLD rule preprocessing options

Update an LLD rule with preprocessing rule "JSONPath".

Request:
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "44211",
        "preprocessing": [
            {
                "type": 12,
                "params": ".path.to.json",
                "error_handler": 2,
                "error_handler_params": "5"
            }
        ],
        "auth": "700ca65537074ec963db7efabda78259",
        "id": 1
    }
}
Updating LLDrulescript

Update an LLDrulescript with a different script and remove unnecessary parameters that were used by previous script.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "23865",
        "parameters": [],
        "script": "Zabbix.log(3, 'Log test');
        return 1;"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": ["23865"
    ]
    },
    "id": 1
}
```

Source

CDiscoveryRule::update() in ui/include/classes/api/services/CDiscoveryRule.php.

**Maintenance**

This class is designed to work with maintenances.

Object references:

- Maintenance
- Time period

Available methods:

- maintenance.create - creating new maintenances
- maintenance.delete - deleting maintenances
- maintenance.get - retrieving maintenances
- maintenance.update - updating maintenances

> **Maintenance object**

The following objects are directly related to the maintenance API.
Maintenance

The maintenance object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maintenanceid</td>
<td>string</td>
<td>(readonly) ID of the maintenance.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the maintenance.</td>
</tr>
<tr>
<td>active_since</td>
<td>timestamp</td>
<td>Time when the maintenance becomes active. The given value will be rounded</td>
</tr>
<tr>
<td></td>
<td></td>
<td>down to minutes.</td>
</tr>
<tr>
<td>active_till</td>
<td>timestamp</td>
<td>Time when the maintenance stops being active. The given value will be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rounded down to minutes.</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Description of the maintenance.</td>
</tr>
<tr>
<td>maintenance_type</td>
<td>integer</td>
<td>Type of maintenance. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) with data collection; 1 - without data collection.</td>
</tr>
<tr>
<td>tags_evaltype</td>
<td>integer</td>
<td>Problem tag evaluation method.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) And/Or; 2 - Or.</td>
</tr>
</tbody>
</table>

Time period

The time period object is used to define periods when the maintenance must come into effect. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>period</td>
<td>integer</td>
<td>Duration of the maintenance period in seconds. The given value will be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rounded down to minutes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 3600.</td>
</tr>
<tr>
<td>timeperiod_type</td>
<td>integer</td>
<td>Type of time period. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) one time only; 2 - daily; 3 - weekly; 4 - monthly.</td>
</tr>
<tr>
<td>start_date</td>
<td>timestamp</td>
<td>Date when the maintenance period must come into effect. Used only for one</td>
</tr>
<tr>
<td></td>
<td></td>
<td>time periods. The given value will be rounded down to minutes.</td>
</tr>
<tr>
<td>start_time</td>
<td>integer</td>
<td>Time of day when the maintenance starts in seconds. Used for daily, weekly</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and monthly periods. The given value will be rounded down to minutes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>every</td>
<td>integer</td>
<td>Used for daily, weekly and monthly periods.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For daily and weekly periods <strong>every</strong> defines day or week intervals at which the maintenance must come into effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For monthly periods, if <strong>dayofweek</strong> property contains at least one selected day of week, the <strong>every</strong> property defines the week of the month when the maintenance must come into effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) first week;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - second week;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - third week;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - fourth week;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - last week.</td>
</tr>
<tr>
<td>dayofweek</td>
<td>integer</td>
<td>Days of the week when the maintenance must come into effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Days are stored in binary form with each bit representing the corresponding day. For example, 4 equals 100 in binary and means,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>that maintenance will be enabled on Wednesday.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used for weekly and monthly time periods. Required only for weekly time periods.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At least one <strong>dayofweek</strong> or day must be specified for monthly time periods.</td>
</tr>
<tr>
<td>day</td>
<td>integer</td>
<td>Day of the month when the maintenance must come into effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used only for monthly time periods.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At least one <strong>dayofweek</strong> or day must be specified for monthly time periods.</td>
</tr>
<tr>
<td>month</td>
<td>integer</td>
<td>Months when the maintenance must come into effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Months are stored in binary form with each bit representing the corresponding month. For example, 5 equals 101 in binary and means,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>that maintenance will be enabled in January and March.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required only for monthly time periods.</td>
</tr>
</tbody>
</table>

**Problem tag**

The problem tag object is used to define which problems must be suppressed when the maintenance comes into effect. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>tag</strong></td>
<td>string</td>
<td>Problem tag name.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>operator</strong></td>
<td>integer</td>
<td>Condition operator.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Equals;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - (default) Contains.</td>
</tr>
<tr>
<td><strong>value</strong></td>
<td>string</td>
<td>Problem tag value.</td>
</tr>
</tbody>
</table>

Tags can only be specified for maintenance periods with data collection ("maintenance_type":0).
**maintenance.create**

**Description**

object maintenance.create(object/array maintenances)

This method allows to create new maintenances.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) Maintenances to create.

Additionally to the standard maintenance properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Host groups that will undergo maintenance. The host groups must have the groupid property defined.</td>
</tr>
<tr>
<td>hosts</td>
<td>object/array</td>
<td>Hosts that will undergo maintenance. The hosts must have the hostid property defined.</td>
</tr>
<tr>
<td>timeperiods</td>
<td>object/array</td>
<td>Maintenance time periods. At least one object of groups or hosts must be specified.</td>
</tr>
<tr>
<td>tags</td>
<td>object/array</td>
<td>Define what problems must be suppressed. If no tags are given, all active maintenance host problems will be suppressed.</td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns an object containing the IDs of the created maintenances under the maintenanceids property. The order of the returned IDs matches the order of the passed maintenances.

**Examples**

**Creating a maintenance**

Create a maintenance with data collection for host group with ID “2” and with problem tags service:mysql and error. It must be active from 22.01.2013 till 22.01.2014, come in effect each Sunday at 18:00 and last for one hour.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "maintenance.create",
    "params": {
        "name": "Sunday maintenance",
        "active_since": 1358844540,
        "active_till": 1390466940,
        "tags_evaltype": 0,
        "groups": [
            {"groupid": "2"}
        ],
        "timeperiods": [
            {
                "period": 3600,
                "timeperiod_type": 3,
                "start_time": 64800,
                "every": 1,
            }
        ]
    }
}
```
"dayofweek": 64
]
,"tags": [
{
"tag": "service",
"operator": "0",
"value": "mysqld"
},
{
"tag": "error",
"operator": "2",
"value": ""
}
],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
"jsonrpc": "2.0",
"result": {
"maintenanceids": [
"3"
]
},
"id": 1
}

See also
• Time period

Source
CMaintenance::create() in ui/include/classes/api/services/CMaintenance.php.

maintenance.delete

Description
object maintenance.delete(array maintenanceIds)

This method allows to delete maintenance periods.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(array) IDs of the maintenance periods to delete.

Return values
(object) Returns an object containing the IDs of the deleted maintenance periods under the maintenanceids property.

Examples
Deleting multiple maintenances
Delete two maintenance periods.

Request:
{
"jsonrpc": "2.0",
"method": "maintenance.delete",
"params": [
}
Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3",
            "1"
        ],
        "auth": "3a57200802b24cda67c4e4010b50c065",
        "id": 1
    },
    "id": 1
}
```

Source
CMaintenance::delete() in ui/include/classes/api/services/CMaintenance.php.

**maintenance.get**

Description

integer/array maintenance.get(object parameters)

The method allows to retrieve maintenances according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only maintenances that are assigned to the given host groups.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only maintenances that are assigned to the given hosts.</td>
</tr>
<tr>
<td>maintenanceids</td>
<td>string/array</td>
<td>Return only maintenances with the given IDs.</td>
</tr>
<tr>
<td>selectHostGroups</td>
<td>query</td>
<td>Return a host groups property with host groups assigned to the maintenance.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return a hosts property with hosts assigned to the maintenance.</td>
</tr>
<tr>
<td>selectTags</td>
<td>query</td>
<td>Return a tags property with problem tags of the maintenance.</td>
</tr>
<tr>
<td>selectTimeperiods</td>
<td>query</td>
<td>Return a timeperiods property with time periods of the maintenance.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Possible values are: maintenanceid, name and maintenance_type. These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>selectGroups</td>
<td>query</td>
<td>This parameter is deprecated, please use selectHostGroups instead. Return a groups property with host groups assigned to the maintenance.</td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

**Retrieving maintenances**

Retrieve all configured maintenances, and the data about the assigned host groups, defined time periods and problem tags.

**Request:**

```
{
  "jsonrpc": "2.0",
  "method": "maintenance.get",
  "params": {
    "output": "extend",
    "selectHostGroups": "extend",
    "selectTimeperiods": "extend",
    "selectTags": "extend"
  },
  "auth": "038e1d7b1735c6a5436ee9eae096879e",
  "id": 1
}
```

**Response:**

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "maintenanceid": "3",
      "name": "Sunday maintenance",
      "maintenance_type": "0",
      "description": "",
      "active_since": "1358844540",
      "active_till": "1390466940",
      "tags_evaltype": "0",
      "groups": [
        {
          "groupid": "4",
          "name": "Zabbix servers",
          "internal": "0"
        }
      ],
      "timeperiods": [
        {
          "timeperiod_type": "3",
          "every": "1",
          "month": "0",
          "dayofweek": "1",
          "day": "0",
          "start_time": "64800",
          "period": "3600",
          "start_date": "2147483647"
        }
      ]
    }
  ]
}
```
"tags": [
  {
    "tag": "service",
    "operator": "0",
    "value": "mysql",
  },
  {
    "tag": "error",
    "operator": "2",
    "value": "",
  }
],
"id": 1

See also
- Host
- Host group
- Time period

Source
CMaintenance::get() in ui/include/classes/api/services/CMaintenance.php.

**maintenance.update**

Description

object maintenance.update(object/array maintenances)

This method allows to update existing maintenances.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

Parameters

(object/array) Maintenance properties to be updated.

The maintenanceid property must be defined for each maintenance, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard maintenance properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Host groups to replace the current groups.</td>
</tr>
<tr>
<td>hosts</td>
<td>object/array</td>
<td>The host groups must have the groupid property defined. Hosts to replace the current hosts.</td>
</tr>
<tr>
<td>timeperiods</td>
<td>object/array</td>
<td>The hosts must have the hostid property defined. Maintenance time periods to replace the current periods.</td>
</tr>
<tr>
<td>tags</td>
<td>object/array</td>
<td>Problem tags to replace the current tags.</td>
</tr>
</tbody>
</table>

At least one host or host group must be defined for each maintenance.

Return values

(object) Returns an object containing the IDs of the updated maintenances under the maintenanceids property.

Examples

Assigning different hosts

Replace the hosts currently assigned to maintenance with two different ones.
Request:

```json
{
    "jsonrpc": "2.0",
    "method": "maintenance.update",
    "params": {
        "maintenanceid": "3",
        "hosts": [
            {"hostid": "10085"},
            {"hostid": "10084"}
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9ea095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Time period

Source

CMaintenance::update() in ui/include/classes/api/services/CMaintenance.php.

Map

This class is designed to work with maps.

Object references:

- Map
- Map element
- Map link
- Map URL
- Map user
- Map user group
- Map shape
- Map line

Available methods:

- map.create - create new maps
- map.delete - delete maps
- map.get - retrieve maps
- map.update - update maps

> Map object

The following objects are directly related to the map API.

Map

The map object has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysmapid</td>
<td>string</td>
<td>(readonly) ID of the map.</td>
</tr>
<tr>
<td>height</td>
<td>integer</td>
<td>Height of the map in pixels.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the map.</td>
</tr>
<tr>
<td>width</td>
<td>integer</td>
<td>Width of the map in pixels.</td>
</tr>
<tr>
<td>backgroundid</td>
<td>string</td>
<td>ID of the image used as the background for the map.</td>
</tr>
<tr>
<td>expand_macros</td>
<td>integer</td>
<td>Whether to expand macros in labels when configuring the map.</td>
</tr>
<tr>
<td>Possible values:</td>
<td></td>
<td>0 - (default) do not expand macros;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - expand macros.</td>
</tr>
<tr>
<td>expandproblem</td>
<td>integer</td>
<td>Whether the problem trigger will be displayed for elements with a single problem.</td>
</tr>
<tr>
<td>Possible values:</td>
<td></td>
<td>0 - always display the number of problems;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) display the problem trigger if there’s only one problem.</td>
</tr>
<tr>
<td>grid_align</td>
<td>integer</td>
<td>Whether to enable grid aligning.</td>
</tr>
<tr>
<td>Possible values:</td>
<td></td>
<td>0 - disable grid aligning;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) enable grid aligning.</td>
</tr>
<tr>
<td>grid_show</td>
<td>integer</td>
<td>Whether to show the grid on the map.</td>
</tr>
<tr>
<td>Possible values:</td>
<td></td>
<td>0 - do not show the grid;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) show the grid.</td>
</tr>
<tr>
<td>grid_size</td>
<td>integer</td>
<td>Size of the map grid in pixels.</td>
</tr>
<tr>
<td>Supported values:</td>
<td></td>
<td>20, 40, 50, 75 and 100.</td>
</tr>
<tr>
<td>highlight</td>
<td>integer</td>
<td>Whether icon highlighting is enabled.</td>
</tr>
<tr>
<td>Possible values:</td>
<td></td>
<td>0 - highlighting disabled;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) highlighting enabled.</td>
</tr>
<tr>
<td>iconmapid</td>
<td>string</td>
<td>ID of the icon map used on the map.</td>
</tr>
<tr>
<td>label_format</td>
<td>integer</td>
<td>Whether to enable advanced labels.</td>
</tr>
<tr>
<td>Possible values:</td>
<td></td>
<td>0 - (default) disable advanced labels;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - enable advanced labels.</td>
</tr>
<tr>
<td>label_location</td>
<td>integer</td>
<td>Location of the map element label.</td>
</tr>
<tr>
<td>Possible values:</td>
<td></td>
<td>0 - (default) bottom;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - left;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - right;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - top.</td>
</tr>
<tr>
<td>label_string_host</td>
<td>string</td>
<td>Custom label for host elements.</td>
</tr>
<tr>
<td>label_string_hostgroup</td>
<td>string</td>
<td>Custom label for host group elements.</td>
</tr>
<tr>
<td>label_string_image</td>
<td>string</td>
<td>Custom label for image elements.</td>
</tr>
<tr>
<td>Required for maps with custom host label type.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>label_string_image</td>
<td>string</td>
<td>Custom label for image elements.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>label_string_map</td>
<td>string</td>
<td>Custom label for map elements.</td>
</tr>
<tr>
<td>label_string_trigger</td>
<td>string</td>
<td>Required for maps with custom map label type.</td>
</tr>
<tr>
<td>label_type</td>
<td>integer</td>
<td>Map element label type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - label;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - IP address;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - (default) element name;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - status only;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - nothing;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - custom.</td>
</tr>
<tr>
<td>label_type_host</td>
<td>integer</td>
<td>Label type for host elements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - label;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - IP address;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - (default) element name;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - status only;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - nothing;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - custom.</td>
</tr>
<tr>
<td>label_type_hostgroup</td>
<td>integer</td>
<td>Label type for host group elements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - label;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - (default) element name;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - status only;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - nothing;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - custom.</td>
</tr>
<tr>
<td>label_type_image</td>
<td>integer</td>
<td>Label type for host group elements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - label;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - (default) element name;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - nothing;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - custom.</td>
</tr>
<tr>
<td>label_type_map</td>
<td>integer</td>
<td>Label type for map elements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - label;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - (default) element name;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - status only;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - nothing;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - custom.</td>
</tr>
<tr>
<td>label_type_trigger</td>
<td>integer</td>
<td>Label type for trigger elements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - label;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - (default) element name;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - status only;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - nothing;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - custom.</td>
</tr>
<tr>
<td>markelements</td>
<td>integer</td>
<td>Whether to highlight map elements that have recently changed their status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) do not highlight elements;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - highlight elements.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>severity_min</td>
<td>integer</td>
<td>Minimum severity of the triggers that will be displayed on the map.</td>
</tr>
<tr>
<td>show_unack</td>
<td>integer</td>
<td>How problems should be displayed.</td>
</tr>
<tr>
<td>userid</td>
<td>string</td>
<td>Map owner user ID.</td>
</tr>
<tr>
<td>private</td>
<td>integer</td>
<td>Type of map sharing.</td>
</tr>
<tr>
<td>show_suppressed</td>
<td>integer</td>
<td>Whether suppressed problems are shown.</td>
</tr>
</tbody>
</table>

Map element
The map element object defines an object displayed on a map. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>selementid</td>
<td>string</td>
<td>(readonly) ID of the map element.</td>
</tr>
<tr>
<td>elements</td>
<td>array</td>
<td>Element data object. Required for host, host group, trigger and map type elements.</td>
</tr>
<tr>
<td>elementtype</td>
<td>integer</td>
<td>Type of map element.</td>
</tr>
<tr>
<td>iconid_off</td>
<td>string</td>
<td>ID of the image used to display the element in default state.</td>
</tr>
<tr>
<td>areatype</td>
<td>integer</td>
<td>How separate host group hosts should be displayed.</td>
</tr>
<tr>
<td>elementsubtype</td>
<td>integer</td>
<td>How a host group element should be displayed on a map.</td>
</tr>
<tr>
<td>evaltype</td>
<td>integer</td>
<td>Map element tag filtering condition evaluation method.</td>
</tr>
<tr>
<td>height</td>
<td>integer</td>
<td>Height of the fixed size host group element in pixels.</td>
</tr>
</tbody>
</table>

Possible values:
- 0 - (default) display the count of all problems;
- 1 - display only the count of unacknowledged problems;
- 2 - display the count of acknowledged and unacknowledged problems separately.

Possible values:
- 0 - public map;
- 1 - (default) private map.

Possible values:
- 0 - (default) hide suppressed problems;
- 1 - show suppressed problems.

Default: 200.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iconid_disabled</td>
<td>string</td>
<td>ID of the image used to display disabled map elements. Unused for image elements.</td>
</tr>
<tr>
<td>iconid_maintenance</td>
<td>string</td>
<td>ID of the image used to display map elements in maintenance. Unused for image elements.</td>
</tr>
<tr>
<td>iconid_on</td>
<td>string</td>
<td>ID of the image used to display map elements with problems. Unused for image elements.</td>
</tr>
<tr>
<td>label</td>
<td>string</td>
<td>Label of the element.</td>
</tr>
<tr>
<td>label_location</td>
<td>integer</td>
<td>Location of the map element label.</td>
</tr>
<tr>
<td>permission</td>
<td>integer</td>
<td>Type of permission level.</td>
</tr>
<tr>
<td>sysmapid</td>
<td>string</td>
<td>(readonly) ID of the map that the element belongs to.</td>
</tr>
<tr>
<td>urls</td>
<td>array</td>
<td>Map element URLs.</td>
</tr>
</tbody>
</table>

The map element URL object is described in detail below.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>use_iconmap</td>
<td>integer</td>
<td>Whether icon mapping must be used for host elements.</td>
</tr>
<tr>
<td>viewtype</td>
<td>integer</td>
<td>Host group element placing algorithm.</td>
</tr>
<tr>
<td>width</td>
<td>integer</td>
<td>Width of the fixed size host group element in pixels.</td>
</tr>
<tr>
<td>x</td>
<td>integer</td>
<td>X-coordinates of the element in pixels.</td>
</tr>
<tr>
<td>y</td>
<td>integer</td>
<td>Y-coordinates of the element in pixels.</td>
</tr>
</tbody>
</table>

Map element Host

The map element Host object defines one host element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostid</td>
<td>string</td>
<td>Host ID</td>
</tr>
</tbody>
</table>

Map element Host group

The map element Host group object defines one host group element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupid</td>
<td>string</td>
<td>Host group ID</td>
</tr>
</tbody>
</table>

Map element Map

1121
The map element Map object defines one map element.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysmapid</td>
<td>string</td>
<td>Map ID</td>
</tr>
</tbody>
</table>

Map element Trigger

The map element Trigger object defines one or more trigger elements.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>triggerid</td>
<td>string</td>
<td>Trigger ID</td>
</tr>
</tbody>
</table>

Map element tag

The map element tag object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Map element tag name.</td>
</tr>
<tr>
<td>operator</td>
<td>string</td>
<td>Map element tag condition operator.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Contains;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Equals;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Does not contain;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - Does not equal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - Exists;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - Does not exist.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Map element tag value.</td>
</tr>
</tbody>
</table>

Map element URL

The map element URL object defines a clickable link that will be available for a specific map element. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysmapelementurlid</td>
<td>string</td>
<td>(readonly) ID of the map element URL.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Link caption.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>Link URL.</td>
</tr>
<tr>
<td>selementid</td>
<td>string</td>
<td>ID of the map element that the URL belongs to.</td>
</tr>
</tbody>
</table>

Map link

The map link object defines a link between two map elements. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>linkid</td>
<td>string</td>
<td>(readonly) ID of the map link.</td>
</tr>
<tr>
<td>selementid1</td>
<td>string</td>
<td>ID of the first map element linked on one end.</td>
</tr>
<tr>
<td>selementid2</td>
<td>string</td>
<td>ID of the first map element linked on the other end.</td>
</tr>
<tr>
<td>color</td>
<td>string</td>
<td>Line color as a hexadecimal color code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 000000.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>drawtype</td>
<td>integer</td>
<td>Link line draw style.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) line;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - bold line;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - dotted line;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - dashed line.</td>
</tr>
<tr>
<td>label</td>
<td>string</td>
<td>Link label.</td>
</tr>
<tr>
<td>linktriggers</td>
<td>array</td>
<td>Map link triggers to use as link status indicators.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The map link trigger object is described in detail below.</td>
</tr>
<tr>
<td>permission</td>
<td>integer</td>
<td>Type of permission level.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1 - none;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - read only;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - read-write.</td>
</tr>
<tr>
<td>sysmapid</td>
<td>string</td>
<td>ID of the map the link belongs to.</td>
</tr>
</tbody>
</table>

**Map link trigger**

The map link trigger object defines a map link status indicator based on the state of a trigger. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>linktriggerid</td>
<td>string</td>
<td>(readonly) ID of the map link trigger.</td>
</tr>
<tr>
<td>triggerid</td>
<td>string</td>
<td>ID of the trigger used as a link indicator.</td>
</tr>
<tr>
<td>color</td>
<td>string</td>
<td>Indicator color as a hexadecimal color code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: DD0000.</td>
</tr>
<tr>
<td>drawtype</td>
<td>integer</td>
<td>Indicator draw style.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) line;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - bold line;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - dotted line;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - dashed line.</td>
</tr>
<tr>
<td>linkid</td>
<td>string</td>
<td>ID of the map link that the link trigger belongs to.</td>
</tr>
</tbody>
</table>

**Map URL**

The map URL object defines a clickable link that will be available for all elements of a specific type on the map. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysmapurlid</td>
<td>string</td>
<td>(readonly) ID of the map URL.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Link caption.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>Link URL.</td>
</tr>
<tr>
<td>elementtype</td>
<td>integer</td>
<td>Type of map element for which the URL will be available.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to the map element &quot;type&quot; property for a list of supported types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.</td>
</tr>
<tr>
<td>sysmapid</td>
<td>string</td>
<td>ID of the map that the URL belongs to.</td>
</tr>
</tbody>
</table>

**Map user**
List of map permissions based on users. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysmapuserid</td>
<td>string</td>
<td>(readonly) ID of the map user.</td>
</tr>
<tr>
<td>user id</td>
<td>string</td>
<td>User ID.</td>
</tr>
<tr>
<td>permission</td>
<td>integer</td>
<td>Type of permission level.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 2 - read only; 3 - read-write;</td>
</tr>
</tbody>
</table>

Map user group

List of map permissions based on user groups. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysmapusrgrpid</td>
<td>string</td>
<td>(readonly) ID of the map user group.</td>
</tr>
<tr>
<td>usrgrpid</td>
<td>string</td>
<td>User group ID.</td>
</tr>
<tr>
<td>permission</td>
<td>integer</td>
<td>Type of permission level.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 2 - read only; 3 - read-write;</td>
</tr>
</tbody>
</table>

Map shapes

The map shape object defines an geometric shape (with or without text) displayed on a map. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysmap_shapeid</td>
<td>string</td>
<td>(readonly) ID of the map shape element.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>Type of map shape element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - rectangle; 1 - ellipse.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Property is required when new shapes are created.</td>
</tr>
<tr>
<td>x</td>
<td>integer</td>
<td>X-coordinates of the shape in pixels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.</td>
</tr>
<tr>
<td>y</td>
<td>integer</td>
<td>Y-coordinates of the shape in pixels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.</td>
</tr>
<tr>
<td>width</td>
<td>integer</td>
<td>Width of the shape in pixels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 200.</td>
</tr>
<tr>
<td>height</td>
<td>integer</td>
<td>Height of the shape in pixels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 200.</td>
</tr>
<tr>
<td>text</td>
<td>string</td>
<td>Text of the shape.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 200.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>font</td>
<td>integer</td>
<td>Font of the text within shape.</td>
</tr>
<tr>
<td>Possible values:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 - Georgia, serif</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - “Palatino Linotype”, “Book Antiqua”, Palatino, serif</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - “Times New Roman”, Times, serif</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 - Arial, Helvetica, sans-serif</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 - “Arial Black”, Gadget, sans-serif</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 - “Comic Sans MS”, cursive, sans-serif</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 - Impact, Charcoal, sans-serif</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 - “Lucida Sans Unicode”, “Lucida Grande”, sans-serif</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 - Tahoma, Geneva, sans-serif</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 - “Trebuchet MS”, Helvetica, sans-serif</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 - Verdana, Geneva, sans-serif</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 - “Courier New”, Courier, monospace</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 - “Lucida Console”, Monaco, monospace</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default: 9.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>font_size</td>
<td>integer</td>
<td>Font size in pixels.</td>
</tr>
<tr>
<td>Default: 11.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>font_color</td>
<td>string</td>
<td>Font color.</td>
</tr>
<tr>
<td>Default: ‘000000’.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>text_halign</td>
<td>integer</td>
<td>Horizontal alignment of text.</td>
</tr>
<tr>
<td>Possible values:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 - center;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - left;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - right.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default: 0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>text_valign</td>
<td>integer</td>
<td>Vertical alignment of text.</td>
</tr>
<tr>
<td>Possible values:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 - middle;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - top;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - bottom.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default: 0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>border_type</td>
<td>integer</td>
<td>Type of the border.</td>
</tr>
<tr>
<td>Possible values:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 - none;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 - ————;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 - ——;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 - — — —.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default: 0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>border_width</td>
<td>integer</td>
<td>Width of the border in pixels.</td>
</tr>
<tr>
<td>Default: 0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>border_color</td>
<td>string</td>
<td>Border color.</td>
</tr>
<tr>
<td>Default: ‘000000’.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>background_color</td>
<td>string</td>
<td>Background color (fill color).</td>
</tr>
<tr>
<td>Default: (empty).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>zindex</td>
<td>integer</td>
<td>Value used to order all shapes and lines (z-index).</td>
</tr>
<tr>
<td>Default: 0.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Map lines

The map line object defines an line displayed on a map. It has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysmap_shapeid</td>
<td>string</td>
<td>(readonly) ID of the map shape element.</td>
</tr>
<tr>
<td>x1</td>
<td>integer</td>
<td>X-coordinates of the line point 1 in pixels.</td>
</tr>
<tr>
<td>y1</td>
<td>integer</td>
<td>Y-coordinates of the line point 1 in pixels.</td>
</tr>
<tr>
<td>x2</td>
<td>integer</td>
<td>X-coordinates of the line point 2 in pixels.</td>
</tr>
<tr>
<td>y2</td>
<td>integer</td>
<td>Y-coordinates of the line point 2 in pixels.</td>
</tr>
<tr>
<td>line_type</td>
<td>integer</td>
<td>Type of the lines.</td>
</tr>
<tr>
<td>line_width</td>
<td>integer</td>
<td>Width of the lines in pixels.</td>
</tr>
<tr>
<td>line_color</td>
<td>string</td>
<td>Line color.</td>
</tr>
<tr>
<td>zindex</td>
<td>integer</td>
<td>Value used to order all shapes and lines (z-index).</td>
</tr>
</tbody>
</table>

**map.create**

**Description**

**object** map.create(object/array maps)

This method allows to create new maps.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

**Parameters**

(object/array) Maps to create.

Additionally to the standard map properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>links</td>
<td>array</td>
<td>Map links to be created on the map.</td>
</tr>
<tr>
<td>selements</td>
<td>array</td>
<td>Map elements to be created on the map.</td>
</tr>
<tr>
<td>urls</td>
<td>array</td>
<td>Map URLs to be created on the map.</td>
</tr>
<tr>
<td>users</td>
<td>array</td>
<td>Map user shares to be created on the map.</td>
</tr>
<tr>
<td>userGroups</td>
<td>array</td>
<td>Map user group shares to be created on the map.</td>
</tr>
<tr>
<td>shapes</td>
<td>array</td>
<td>Map shapes to be created on the map.</td>
</tr>
<tr>
<td>lines</td>
<td>array</td>
<td>Map lines to be created on the map.</td>
</tr>
</tbody>
</table>
To create map links you’ll need to set a map element `selementid` to an arbitrary value and then use this value to reference this element in the links `selementid1` or `selementid2` properties. When the element is created, this value will be replaced with the correct ID generated by Zabbix. See example.

Return values

(object) Returns an object containing the IDs of the created maps under the `sysmapids` property. The order of the returned IDs matches the order of the passed maps.

Examples

Create an empty map

Create a map with no elements.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Map",
        "width": 600,
        "height": 600
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "8"
        ],
        "id": 1
    }
}
```

Create a host map

Create a map with two host elements and a link between them. Note the use of temporary "selementid1" and "selementid2" values in the map link object to refer to map elements.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Host map",
        "width": 600,
        "height": 600,
        "selements": [
            {
                "selementid": "1",
                "elements": [
                    {
                        "hostid": "1033"
                    }
                ],
                "elementtype": 0,
                "iconid_off": "2"
            },
            {
                "selementid": "2",
                "elements": [
                    {
                        "hostid": "1037"
                    }
                ]
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```
Create a trigger map

Create a map with trigger element, which contains two triggers.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Trigger map",
        "width": 600,
        "height": 600,
        "selements": [
            {
                "elements": [
                    {"triggerid": "12345"},
                    {"triggerid": "67890"}
                ],
                "elementtype": 2,
                "iconid_off": "2"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": ["9"]
    },
    "id": 1
}
```
Map sharing

Create a map with two types of sharing (user and user group).

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Map sharing",
        "width": 600,
        "height": 600,
        "users": [
            {
                "userid": "4",
                "permission": "3"
            }
        ],
        "userGroups": [
            {
                "usrgrpid": "7",
                "permission": "2"
            }
        ],
        "auth": "038e1d7b1735c6a5436ee9eae095879e",
        "id": 1
    }
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ],
        "id": 1
    }
}
```

Map shapes

Create a map with map name title.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Host map",
        "width": 600,
        "height": 600,
        "shapes": [
            {
                "type": 0,
                "x": 0,
                "y": 0,
                "width": 600,
                "height": 11,
                "text": "{MAP.NAME}"
            }
        ]
    }
}
```
Map lines
Create a map line.

Request:
```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Map API lines",
        "width": 500,
        "height": 500,
        "lines": [
            {
                "x1": 30,
                "y1": 10,
                "x2": 100,
                "y2": 50,
                "line_type": 1,
                "line_width": 10,
                "line_color": "009900"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:
```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "11"
        ]
    },
    "id": 1
}
```

See also
- Map element
- Map link
- Map URL
- Map user
- Map user group
- Map shape
- Map line
map.delete

Description

object map.delete(array mapIds)

This method allows to delete maps.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the maps to delete.

Return values

(object) Returns an object containing the IDs of the deleted maps under the sysmapIds property.

Examples

Delete multiple maps

Delete two maps.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "map.delete",
    "params": [
        "12",
        "34"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapIds": [
            "12",
            "34"
        ]
    },
    "id": 1
}
```

Source

CMap::create() in ui/include/classes/api/services/CMap.php.

map.get

Description

integer/array map.get(object parameters)

The method allows to retrieve maps according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

Source

CMap::delete() in ui/include/classes/api/services/CMap.php.
Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysmapids</td>
<td>string/array</td>
<td>Returns only maps with the given IDs.</td>
</tr>
<tr>
<td>userids</td>
<td>string/array</td>
<td>Returns only maps that belong to the given user IDs.</td>
</tr>
<tr>
<td>expandUrls</td>
<td>flag</td>
<td>Adds global map URLs to the corresponding map elements and expands macros in all map element URLs.</td>
</tr>
<tr>
<td>selectIconMap</td>
<td>query</td>
<td>Returns an icon map property with the icon map used on the map.</td>
</tr>
<tr>
<td>selectLinks</td>
<td>query</td>
<td>Returns a links property with the map links between elements.</td>
</tr>
<tr>
<td>selectSelements</td>
<td>query</td>
<td>Returns a selements property with the map elements.</td>
</tr>
<tr>
<td>selectUrls</td>
<td>query</td>
<td>Returns a urls property with the map URLs.</td>
</tr>
<tr>
<td>selectUsers</td>
<td>query</td>
<td>Returns a users property with users that the map is shared with.</td>
</tr>
<tr>
<td>selectUserGroups</td>
<td>query</td>
<td>Returns a userGroups property with user groups that the map is shared with.</td>
</tr>
<tr>
<td>selectShapes</td>
<td>query</td>
<td>Returns a shapes property with the map shapes.</td>
</tr>
<tr>
<td>selectLines</td>
<td>query</td>
<td>Returns a lines property with the map lines.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Possible values are: name, width and height.

These parameters being common for all get methods are described in detail in the reference commentary.

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieve a map

Retrieve all data about map "3".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "map.get",
    "params": {
        "output": "extend",
        "selectSelements": "extend",
        "selectLinks": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
        "selectShapes": "extend",
        "selectLines": "extend",
        "sysmapids": "3"
    },
    "auth": "03e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```
Response:

```json
{
   "jsonrpc": "2.0",
   "result": [
   {
      "selements": [
      {
         "selementid": "10",
         "sysmapid": "3",
         "elementtype": "4",
         "evaltype": "0",
         "iconid_off": "1",
         "iconid_on": "0",
         "label": "Zabbix server",
         "label_location": "3",
         "x": "11",
         "y": "141",
         "iconid_disabled": "0",
         "iconid_maintenance": "0",
         "elementsubtype": "0",
         "areatype": "0",
         "width": "200",
         "height": "200",
         "tags": [
         {
            "tag": "service",
            "value": "mysqld",
            "operator": "0"
         }
         ],
         "viewtype": "0",
         "use_iconmap": "1",
         "urls": [],
         "elements": []
      },
      {
         "selementid": "11",
         "sysmapid": "3",
         "elementtype": "4",
         "evaltype": "0",
         "iconid_off": "1",
         "iconid_on": "0",
         "label": "Web server",
         "label_location": "3",
         "x": "211",
         "y": "191",
         "iconid_disabled": "0",
         "iconid_maintenance": "0",
         "elementsubtype": "0",
         "areatype": "0",
         "width": "200",
         "height": "200",
         "viewtype": "0",
         "use_iconmap": "1",
         "urls": [],
         "elements": []
      },
      {
         "selementid": "12",
         "sysmapid": "3",
         "elementtype": "0",
         "evaltype": "0",
         "iconid_off": "1",
         "iconid_on": "0",
         "label": "Zabbix server",
         "label_location": "3",
         "x": "11",
         "y": "141",
         "iconid_disabled": "0",
         "iconid_maintenance": "0",
         "elementsubtype": "0",
         "areatype": "0",
         "width": "200",
         "height": "200",
         "viewtype": "0",
         "use_iconmap": "1",
         "urls": [],
         "elements": []
      }
   ]
}
```
{ "evaltype": "0", "iconid_off": "185", "iconid_on": "0", "label": "\r\n{HOST.NAME}\r\n{HOST.CONN} \r\n", "label_location": "0", "x": "111", "y": "61", "iconid_disabled": "0", "iconid_maintenance": "0", "elementsubtype": "0", "areatype": "0", "width": "200", "height": "200", "viewtype": "0", "use_icolormap": "0", "tags": [], "urls": [], "elements": [ { "hostid": "10084" } ] ] }, "links": [ { "linkid": "23", "sysmapid": "3", "selementid1": "10", "selementid2": "11", "drawtype": "0", "color": "00CC00", "label": "", "linktriggers": [] } ], "users": [ { "sysmapuserid": "1", "userid": "2", "permission": "2" } ], "userGroups": [ { "sysmapusrgrpайд": "1", "usrgrpiday": "7", "permission": "2" } ], "shapes": [ { "sysmap_shapeid": "1", "type": "0", "x": "0", "y": "0", "width": "680", "height": "15", "text": "{MAP.NAME} ", "font": "9", "font_size": "11", "font_color": "000000" } ]}
map.update

Description

object map.update(object/array maps)

This method allows to update existing maps.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Map properties to be updated.

The mapid property must be defined for each map, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard map properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>links</td>
<td>array</td>
<td>Map links to replace the existing links.</td>
</tr>
<tr>
<td>elements</td>
<td>array</td>
<td>Map elements to replace the existing elements.</td>
</tr>
<tr>
<td>urls</td>
<td>array</td>
<td>Map URLs to replace the existing URLs.</td>
</tr>
<tr>
<td>users</td>
<td>array</td>
<td>Map user shares to replace the existing elements.</td>
</tr>
<tr>
<td>userGroups</td>
<td>array</td>
<td>Map user group shares to replace the existing elements.</td>
</tr>
<tr>
<td>shapes</td>
<td>array</td>
<td>Map shapes to replace the existing shapes.</td>
</tr>
<tr>
<td>lines</td>
<td>array</td>
<td>Map lines to replace the existing lines.</td>
</tr>
</tbody>
</table>

To create map links between new map elements you’ll need to set an element’s selementid to an arbitrary value and then use this value to reference this element in the links selementid1 or selementid2 properties. When the element is created, this value will be replaced with the correct ID generated by Zabbix. See example for map.create.

Return values

(object) Returns an object containing the IDs of the updated maps under the sysmapids property.

Examples

Resize a map

Change the size of the map to 1200x1200 pixels.

Request:

```json
{   "jsonrpc": "2.0",   "method": "map.update",   "params": {       "sysmapid": "8",       "width": 1200,       "height": 1200   },   "auth": "038e1d7b1735c6a5436ee9eae095879e",   "id": 1 }
```

Response:
Change map owner
Available only for admins and super admins.

Request:
{
    "jsonrpc": "2.0",
    "method": "map.update",
    "params": {
        "sysmapid": "9",
        "userid": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae9eae095879e",
    "id": 2
}

Response:
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": ["9"],
        "id": 2
    }
}

See also
- Map element
- Map link
- Map URL
- Map user
- Map user group
- Map shapes
- Map lines

Source
CMap::update() in ui/include/classes/api/services/CMap.php.

Media type

This class is designed to work with media types.

Object references:

- Media type

Available methods:

- mediatype.create - creating new media types
- mediatype.delete - deleting media types
- mediatype.get - retrieving media types
> Media type object

The following objects are directly related to the mediatype API.

Media type

The media type object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mediatypeid</td>
<td>string</td>
<td>(readonly) ID of the media type.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the media type.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>Transport used by the media type.</td>
</tr>
<tr>
<td>exec_path</td>
<td>string</td>
<td>For script media types exec_path contains the name of the executed script.</td>
</tr>
<tr>
<td>gsm_modem</td>
<td>string</td>
<td>Serial device name of the GSM modem.</td>
</tr>
<tr>
<td>passwd</td>
<td>string</td>
<td>Authentication password.</td>
</tr>
<tr>
<td>smtp_email</td>
<td>string</td>
<td>Email address from which notifications will be sent.</td>
</tr>
<tr>
<td>smtp_helo</td>
<td>string</td>
<td>SMTP HELO.</td>
</tr>
<tr>
<td>smtp_server</td>
<td>string</td>
<td>SMTP server.</td>
</tr>
<tr>
<td>smtp_port</td>
<td>integer</td>
<td>SMTP server port to connect to.</td>
</tr>
<tr>
<td>smtp_security</td>
<td>integer</td>
<td>SMTP connection security level to use.</td>
</tr>
<tr>
<td>smtp_verify_host</td>
<td>integer</td>
<td>SSL verify host for SMTP.</td>
</tr>
<tr>
<td>smtp_verify_peer</td>
<td>integer</td>
<td>SSL verify peer for SMTP.</td>
</tr>
<tr>
<td>smtp_authentication</td>
<td>integer</td>
<td>SMTP authentication method to use.</td>
</tr>
</tbody>
</table>

Possible values:
- 0 - None;
- 1 - STARTTLS;
- 2 - SSL/TLS.

Possible values:
- 0 - No;
- 1 - Yes.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>integer</td>
<td>Whether the media type is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) enabled; 1 - disabled.</td>
</tr>
<tr>
<td>username</td>
<td>string</td>
<td>Username. Used for email media types.</td>
</tr>
<tr>
<td>exec_params</td>
<td>string</td>
<td>Used for email media types. Script parameters.</td>
</tr>
<tr>
<td>maxsessions</td>
<td>integer</td>
<td>The maximum number of alerts that can be processed in parallel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for SMS: 1 - (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for other media types: 0-100</td>
</tr>
<tr>
<td>maxattempts</td>
<td>integer</td>
<td>The maximum number of attempts to send an alert.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-100</td>
</tr>
<tr>
<td>attempt_interval</td>
<td>string</td>
<td>The interval between retry attempts. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0-1h</td>
</tr>
<tr>
<td>content_type</td>
<td>integer</td>
<td>Message format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - plain text; 1 - (default) html.</td>
</tr>
<tr>
<td>script</td>
<td>string</td>
<td>Media type webhook script javascript.</td>
</tr>
<tr>
<td>timeout</td>
<td>string</td>
<td>Media type webhook script timeout. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1-60s</td>
</tr>
<tr>
<td>process_tags</td>
<td>integer</td>
<td>Defines should the webhook script response to be interpreted as tags and these tags should be added to associated event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Ignore webhook script response.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Process webhook script response as tags</td>
</tr>
<tr>
<td>show_event_menu</td>
<td>integer</td>
<td>Show media type entry in problem.get and event.get property urls.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Do not add urls entry.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Add media type to urls property.</td>
</tr>
<tr>
<td>event_menu_url</td>
<td>string</td>
<td>Define url property of media type entry in urls property of problem.get and event.get.</td>
</tr>
</tbody>
</table>
Webhook parameters

Parameters passed to webhook script when it is called, have the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Parameter name.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Parameter value, support macros. Supported macros described on page.</td>
</tr>
</tbody>
</table>

Message template

The message template object defines a template that will be used as a default message for action operations to send a notification. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventsource</td>
<td>integer</td>
<td>Event source. Possible values: 0 - triggers; 1 - discovery; 2 - autoregistration; 3 - internal; 4 - services.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>recovery</td>
<td>integer</td>
<td>Operation mode. Possible values: 0 - operations; 1 - recovery operations; 2 - update operations.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>subject</td>
<td>string</td>
<td>Message subject.</td>
</tr>
<tr>
<td>message</td>
<td>string</td>
<td>Message text.</td>
</tr>
</tbody>
</table>

mediatype.create

Description

object mediatype.create(object/array mediaTypes)

This method allows to create new media types. This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Media types to create.

Additionally to the standard media type properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameters</td>
<td>array</td>
<td>Webhook parameters to be created for the media type.</td>
</tr>
<tr>
<td>message_templates</td>
<td>array</td>
<td>Message templates to be created for the media type.</td>
</tr>
</tbody>
</table>

Return values
Returns an object containing the IDs of the created media types under the mediatypeids property. The order of the returned IDs matches the order of the passed media types.

Examples

Creating an e-mail media type

Create a new e-mail media type with a custom SMTP port and message templates.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "mediatype.create",
   "params": {
      "type": "0",
      "name": "E-mail",
      "smtp_server": "mail.example.com",
      "smtp_helo": "example.com",
      "smtp_email": "zabbix@example.com",
      "smtp_port": "587",
      "content_type": "1",
      "message_templates": [
         {
            "eventsource": "0",
            "recovery": "0",
            "subject": "Problem: {EVENT.NAME} on host \"{HOST.NAME}\" started at {EVENT.TIME}."
         },
         {
            "eventsource": "0",
            "recovery": "1",
            "subject": "Resolved in {EVENT.DURATION}: {EVENT.NAME} on host \"{HOST.NAME}\" has been resolved at {EVENT.RECOVERY.DATE} {EVENT.RECOVERY.TIME}.
         },
         {
            "eventsource": "0",
            "recovery": "2",
            "subject": "Updated problem in {EVENT.AGE}: {EVENT.NAME} on host \"{HOST.NAME}\" at {EVENT.UPDATE.DATE} {EVENT.UPDATE.TIME}.
         }
      ],
      "auth": "038e1d7b1735c6a5436ee9eae095879e",
      "id": 1
   }
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "mediatypeids": [7]
   },
   "id": 1
}
```

Creating a script media type

Create a new script media type with a custom value for the number of attempts and the interval between them.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "mediatype.create",
```
"params": {
  "type": "1",
  "name": "Push notifications",
  "exec_path": "push-notification.sh",
  "exec_params": "{ALERT.SENDTO}\n{ALERT.SUBJECT}\n{ALERT.MESSAGE}\n",
  "maxattempts": "5",
  "attempt_interval": "11s"
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": {
    "mediatypeids": [
      8
    ],
  },
  "id": 1
}

Creating a webhook media type

Create a new webhook media type.

Request:
{
  "jsonrpc": "2.0",
  "method": "mediatype.create",
  "params": {
    "type": "4",
    "name": "Webhook",
    "script": "var Webhook = {
      token: null,
      to: null,
      subject: null,
      message: null,

      sendMessage: function() {
        // some logic
      }
    }",
    "parameters": [
      {
        "name": "Message",
        "value": "{ALERT.MESSAGE}"
      },
      {
        "name": "Subject",
        "value": "{ALERT.SUBJECT}"
      },
      {
        "name": "To",
        "value": "{ALERT.SENDTO}"
      },
      {
        "name": "Token",
        "value": "<Token>"
      }
    ]
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": {
    "mediatypeids": [1142]
  },
  "id": 1
}
mediatype.delete

Description

object mediatype.delete(array mediaTypeIds)

This method allows to delete media types.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the media types to delete.

Return values

(object) Returns an object containing the IDs of the deleted media types under the mediatypeids property.

Examples

Delete two media types.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.delete",
    "params": [
        "3",
        "5"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "3",
            "5"
        ]
    },
    "id": 1
}
```

mediatype.get

Description

integer|array mediatype.get(object parameters)
The method allows to retrieve media types according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mediatypeids</td>
<td>string/array</td>
<td>Return only media types with the given IDs.</td>
</tr>
<tr>
<td>mediaids</td>
<td>string/array</td>
<td>Return only media types used by the given media.</td>
</tr>
<tr>
<td>usersids</td>
<td>string/array</td>
<td>Return only media types used by the given users.</td>
</tr>
<tr>
<td>selectMessageTemplates</td>
<td>query</td>
<td>Return a message_templates property with an array of media type messages.</td>
</tr>
<tr>
<td>selectUsers</td>
<td>query</td>
<td>Return a users property with the users that use the media type.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving media types

Retrieve all configured media types.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "mediatype.get",
  "params": {
    "output": "extend",
    "selectMessageTemplates": "extend"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": [
    {
      "mediatypeid": "1"
    }
  ]
}
```
{ "type": "0",
"name": "Email",
"smtp_server": "mail.example.com",
"smtp_helo": "example.com",
"smtp_email": "zabbix@example.com",
"exec_path": "",
"gsm_modem": "",
"username": "",
"passwd": "",
"status": "0",
"smtp_port": "25",
"smtp_security": "0",
"smtp_verify_peer": "0",
"smtp_verify_host": "0",
"smtp_authentication": "0",
"exec_params": "",
"maxsessions": "1",
"maxattempts": "3",
"attempt_interval": "10s",
"content_type": "0",
"script": "",
"timeout": "30s",
"process_tags": "0",
"show_event_menu": "1",
"event_menu_url": "",
"event_menu_name": "",
"description": "",
"message_templates": [
  {
    "eventsource": "0",
    "recovery": "0",
    "subject": "Problem: {EVENT.NAME}"
  },
  {
    "eventsource": "0",
    "recovery": "1",
    "subject": "Resolved: {EVENT.NAME}"
  },
  {
    "eventsource": "0",
    "recovery": "2",
    "subject": "Updated problem: {EVENT.NAME}"
  },
  {
    "eventsource": "1",
    "recovery": "0",
    "subject": "Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IPADDRESS}"
  },
  {
    "eventsource": "2",
    "recovery": "0",
    "subject": "Autoregistration: {HOST.HOST}"
  }
],
"parameters": []
}
"mediatypeid": "3",
"type": "2",
"name": "SMS",
"smtp_server": "",
"smtp_helo": "",
"smtp_email": "",
"exec_path": "",
"gsm_modem": "/dev/ttyS0",
"username": "",
"passwd": "",
"status": "0",
"smtp_port": "25",
"smtp_security": "0",
"smtp_verify_peer": "0",
"smtp_verify_host": "0",
"smtp_authentication": "0",
"exec_params": "",
"maxsessions": "1",
"maxattempts": "3",
"attempt_interval": "10s",
"content_type": "1",
"script": "",
"timeout": "30s",
"process_tags": "0",
"show_event_menu": "1",
"event_menu_url": "",
"event_menu_name": "",
"description": "",
"message_templates": [
{
  "eventsourc"e": "0",
  "recovery": "0",
  "subject": "",
  "message": "{EVENT.SEVERITY}: {EVENT.NAME}\nHost: {HOST.NAME}\nEVENT.DATE} {EVENT.TIME}"

},
{
  "eventsourc"e": "0",
  "recovery": "1",
  "subject": "",
  "message": "RESOLVED: {EVENT.NAME}\nHost: {HOST.NAME}\nEVENT.DATE} {EVENT.TIME}"

},
{
  "eventsourc"e": "0",
  "recovery": "2",
  "subject": "",
  "message": "{USER.FULLNAME} {EVENT.UPDATE.ACTION} problem at {EVENT.UPDATE.DATE} {EVENT.TIME}"

},
{
  "eventsourc"e": "1",
  "recovery": "0",
  "subject": "",
  "message": "Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IPADDRESS}"

},
{
  "eventsourc"e": "2",
  "recovery": "0",
  "subject": "",
  "message": "Autoregistration: {HOST.HOST}\nHost IP: {HOST.IP}\nAgent port: {HOST.PORT}"

}
],
"parameters": []
}
mediatype.update

Description

object mediatype.update(object/array mediaTypes)

This method allows to update existing media types.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Media type properties to be updated.

The mediatypeid property must be defined for each media type, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard media type properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameters</td>
<td>array</td>
<td>Webhook parameters to replace the current webhook parameters.</td>
</tr>
<tr>
<td>message_templates</td>
<td>array</td>
<td>Message templates to replace the current message templates.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated media types under the mediatypeIds property.

Examples

Enabling a media type

Enable a media type, that is, set its status to “0”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.update",
    "params": {
        "mediatypeid": "6",
        "status": "0"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeIds": [
            "6"
        ]
    },
    "id": 1
}
```
Problem

This class is designed to work with problems.

Object references:

- Problem

Available methods:

- problem.get - retrieving problems

> Problem object

Problems are created by the Zabbix server and cannot be modified via the API.

The problem object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventid</td>
<td>string</td>
<td>ID of the problem event.</td>
</tr>
<tr>
<td>source</td>
<td>integer</td>
<td>Type of the problem event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - event created by a trigger;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - internal event;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - event created on service status update.</td>
</tr>
<tr>
<td>object</td>
<td>integer</td>
<td>Type of object that is related to the problem event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for trigger events:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - trigger.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for internal events:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - trigger;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - item;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - LLD rule.</td>
</tr>
<tr>
<td>objectid</td>
<td>string</td>
<td>ID of the related object.</td>
</tr>
<tr>
<td>clock</td>
<td>timestamp</td>
<td>Time when the problem event was created.</td>
</tr>
<tr>
<td>ns</td>
<td>integer</td>
<td>Nanoseconds when the problem event was created.</td>
</tr>
<tr>
<td>r_eventid</td>
<td>string</td>
<td>Recovery event ID.</td>
</tr>
<tr>
<td>r_clock</td>
<td>timestamp</td>
<td>Time when the recovery event was created.</td>
</tr>
<tr>
<td>r_ns</td>
<td>integer</td>
<td>Nanoseconds when the recovery event was created.</td>
</tr>
<tr>
<td>correlationid</td>
<td>string</td>
<td>Correlation rule ID if this event was recovered by global correlation rule.</td>
</tr>
<tr>
<td>userid</td>
<td>string</td>
<td>User ID if the problem was manually closed.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Resolved problem name.</td>
</tr>
<tr>
<td>acknowledged</td>
<td>integer</td>
<td>Acknowledge state for problem.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - not acknowledged;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - acknowledged.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>severity</td>
<td>integer</td>
<td>Problem current severity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - not classified;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - information;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - warning;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - average;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - high;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - disaster.</td>
</tr>
<tr>
<td>suppressed</td>
<td>integer</td>
<td>Whether the problem is suppressed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - problem is in normal state;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - problem is suppressed.</td>
</tr>
<tr>
<td>opdata</td>
<td>string</td>
<td>Operational data with expanded macros.</td>
</tr>
<tr>
<td>urls</td>
<td>array of Media type URLs</td>
<td>Active media types URLs.</td>
</tr>
</tbody>
</table>

Problem tag
The problem tag object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Problem tag name.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Problem tag value.</td>
</tr>
</tbody>
</table>

Media type URLs
Object with media type url have the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Media type defined URL name.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>Media type defined URL value.</td>
</tr>
</tbody>
</table>

Results will contain entries only for active media types with enabled event menu entry. Macro used in properties will be expanded, but if one of properties contain non expanded macro both properties will be excluded from results. Supported macros described on page.

**problem.get**

Description

integer/array problem.get(object parameters)
The method allows to retrieve problems according to the given parameters.

This method is for retrieving unresolved problems. It is also possible, if specified, to additionally retrieve recently resolved problems. The period that determines how old is “recently” is defined in Administration → General. Problems that were resolved prior to that period are not kept in the problem table. To retrieve problems that were resolved further back in the past, use the event.get method.

This method may return problems of a deleted entity if these problems have not been removed by the housekeeper yet.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventids</td>
<td>string/array</td>
<td>Return only problems with the given IDs.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only problems created by objects that belong to the given host groups.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only problems created by objects that belong to the given hosts.</td>
</tr>
<tr>
<td>objectids</td>
<td>string/array</td>
<td>Return only problems created by the given objects.</td>
</tr>
<tr>
<td>source</td>
<td>integer</td>
<td>Return only problems with the given type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to the problem event object page for a list of supported event types.</td>
</tr>
<tr>
<td>object</td>
<td>integer</td>
<td>Return only problems created by objects of the given type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to the problem event object page for a list of supported object types.</td>
</tr>
<tr>
<td>acknowledged</td>
<td>boolean</td>
<td>true - return acknowledged problems only; false - unacknowledged only.</td>
</tr>
<tr>
<td>suppressed</td>
<td>boolean</td>
<td>true - return only suppressed problems; false - return problems in the normal state.</td>
</tr>
<tr>
<td>severities</td>
<td>integer/array</td>
<td>Return only problems with given event severities. Applies only if object is trigger.</td>
</tr>
<tr>
<td>evaltype</td>
<td>integer</td>
<td>Rules for tag searching.</td>
</tr>
<tr>
<td>tags</td>
<td>array of objects</td>
<td>Return only problems with given tags. Exact match by tag and case-insensitive search by value and operator. Format: [{&quot;tag&quot;: &quot;&lt;tag&gt;&quot;, &quot;value&quot;: &quot;&lt;value&gt;&quot;, &quot;operator&quot;: &quot;&lt;operator&gt;&quot;}, ...]. An empty array returns all problems.</td>
</tr>
<tr>
<td>recent</td>
<td>boolean</td>
<td>true - return PROBLEM and recently RESOLVED problems (depends on Display OK triggers for N seconds)</td>
</tr>
<tr>
<td>eventid_from</td>
<td>string</td>
<td>Return only problems with IDs greater or equal to the given ID.</td>
</tr>
<tr>
<td>eventid_till</td>
<td>string</td>
<td>Return only problems with IDs less or equal to the given ID.</td>
</tr>
<tr>
<td>time_from</td>
<td>timestamp</td>
<td>Return only problems that have been created after or at the given time.</td>
</tr>
<tr>
<td>time_till</td>
<td>timestamp</td>
<td>Return only problems that have been created before or at the given time.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>selectAcknowledges</td>
<td>query</td>
<td>Return an acknowledges property with the problem updates. Problem updates are sorted in reverse chronological order. The problem update object has the following properties: acknowledgeid - (string) update's ID; userid - (string) ID of the user that updated the event; eventid - (string) ID of the updated event; clock - (timestamp) time when the event was updated; message - (string) text of the message; action - (integer) type of update action (see event.acknowledge); old_severity - (integer) event severity before this update action; new_severity - (integer) event severity after this update action;</td>
</tr>
<tr>
<td>selectTags</td>
<td>query</td>
<td>Return a tags property with the problem tags. Output format: [{&quot;tag&quot;: &quot;&lt;tag&gt;&quot;, &quot;value&quot;: &quot;&lt;value&gt;&quot;}, ...].</td>
</tr>
<tr>
<td>selectSuppressionData</td>
<td>query</td>
<td>Return a suppression_data property with the list of active maintenances and manual suppressions: maintenanceid - (string) ID of the maintenance; userid - (string) ID of user who suppressed the problem; suppress_until - (integer) time until the problem is suppressed. Sort the result by the given properties. Supports count.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Possible values are: eventid. These parameters being common for all get methods are described in detail in the reference commentary page.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values
(integer/array) Returns either:
- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving trigger problem events

Retrieve recent events from trigger "15112."

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "problem.get",
    "params": {
        "output": "extend",
        "selectAcknowledges": "extend",
        "selectTags": "extend",
        "selectSuppressionData": "extend",
        "objectids": "15112",
        "recent": "true",
        "sortfield": ["eventid"]
    }
}
```
See also
- Alert
- Item
- Host
- LLD rule
• Trigger
Source
CEvent::get() in ui/include/classes/api/services/CProblem.php.

Proxy

This class is designed to work with proxies.

Object references:

  • Proxy
  • Proxy interface

Available methods:

  • proxy.create - create new proxies
  • proxy.delete - delete proxies
  • proxy.get - retrieve proxies
  • proxy.update - update proxies

> Proxy object

The following objects are directly related to the proxy API.

Proxy

The proxy object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proxyid</td>
<td>string</td>
<td>(readonly) ID of the proxy.</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>Name of the proxy.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Type of proxy. Possible values: 5 - active proxy; 6 - passive proxy.</td>
</tr>
<tr>
<td>description</td>
<td>text</td>
<td>Description of the proxy.</td>
</tr>
<tr>
<td>lastaccess</td>
<td>timestamp</td>
<td>(readonly) Time when the proxy last connected to the server.</td>
</tr>
<tr>
<td>tls_connect</td>
<td>integer</td>
<td>Connections to host. Possible values are: 1 - (default) No encryption; 2 - PSK; 4 - certificate.</td>
</tr>
<tr>
<td>tls_accept</td>
<td>integer</td>
<td>Connections from host. Possible bitmap values are: 1 - (default) No encryption; 2 - PSK; 4 - certificate.</td>
</tr>
<tr>
<td>tls_issuer</td>
<td>string</td>
<td>Certificate issuer.</td>
</tr>
<tr>
<td>tls_subject</td>
<td>string</td>
<td>Certificate subject.</td>
</tr>
<tr>
<td>tls_psk_identity</td>
<td>string</td>
<td>(write-only) PSK identity. Required if either tls_connect or tls_accept has PSK enabled. Do not put sensitive information in the PSK identity, it is transmitted unencrypted over the network to inform a receiver which PSK to use.</td>
</tr>
<tr>
<td>tls_psk</td>
<td>string</td>
<td>(write-only) The preshared key, at least 32 hex digits. Required if either tls_connect or tls_accept has PSK enabled.</td>
</tr>
<tr>
<td>proxy_address</td>
<td>string</td>
<td>Comma-delimited IP addresses or DNS names of active Zabbix proxy.</td>
</tr>
</tbody>
</table>
### Proxy Interface

The proxy interface object defines the interface used to connect to a passive proxy. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dns</td>
<td>string</td>
<td>DNS name to connect to. Can be empty if connections are made via IP address.</td>
</tr>
<tr>
<td>ip</td>
<td>string</td>
<td>IP address to connect to. Can be empty if connections are made via DNS names.</td>
</tr>
<tr>
<td>port</td>
<td>string</td>
<td>Port number to connect to. Can be empty if connections are made via DNS names.</td>
</tr>
<tr>
<td>useip</td>
<td>integer</td>
<td>Whether the connection should be made via IP address. Possible values are: 0 - connect using DNS name; 1 - connect using IP address.</td>
</tr>
</tbody>
</table>

### proxy.create

**Description**

```
object proxy.create(object/array proxies)
```

This method allows to create new proxies.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) Proxies to create.

Additionally to the standard proxy properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hosts</td>
<td>array</td>
<td><strong>Hosts</strong> to be monitored by the proxy. If a host is already monitored by a different proxy, it will be reassigned to the current proxy.</td>
</tr>
<tr>
<td>interface</td>
<td>object</td>
<td>The hosts must have the <code>host.id</code> property defined. Host <strong>interface</strong> to be created for the passive proxy. Required for passive proxies.</td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns an object containing the IDs of the created proxies under the `proxy.ids` property. The order of the returned IDs matches the order of the passed proxies.

**Examples**

Create an active proxy

Create an action proxy "Active proxy" and assign a host to be monitored by it.

Request:
Create a passive proxy

Create a passive proxy "Passive proxy" and assign two hosts to be monitored by it.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.create",
    "params": {
        "host": "Passive proxy",
        "status": "6",
        "interface": {
            "ip": "127.0.0.1",
            "dns": "",
            "useip": "1",
            "port": "10051"
        },
        "hosts": [
            {
                "hostid": "10192"
            },
            {
                "hostid": "10139"
            }
        ],
        "auth": "ab9638041ec6922cb140b07982b268f47",
        "id": 1
    }
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": ["10284"]
    },
    "id": 1
}
```
proxy.delete

Description
object proxy.delete(array proxies)

This method allows to delete proxies.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(array) IDs of proxies to delete.

Return values
(object) Returns an object containing the IDs of the deleted proxies under the proxyids property.

Examples
Delete multiple proxies
Delete two proxies.

Request:
{
   "jsonrpc": "2.0",
   "method": "proxy.delete",
   "params": [
      "10286",
      "10285"
   ],
   "auth": "3a57200802b24cda67c4e4010b50c065",
   "id": 1
}

Response:
{
   "jsonrpc": "2.0",
   "result": {
      "proxyids": [
      "10286",
      "10285"
   ],
   "id": 1
}

Source
CProxy::create() in ui/include/classes/api/services/CProxy.php.

See also
• Host
• Proxy interface

Source
CProxy::create() in ui/include/classes/api/services/CProxy.php.
proxy.get

Description

integer/array proxy.get(object parameters)

The method allows to retrieve proxies according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proxyids</td>
<td>string/array</td>
<td>Return only proxies with the given IDs.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return a hosts property with the hosts monitored by the proxy.</td>
</tr>
<tr>
<td>selectInterface</td>
<td>query</td>
<td>Return an interface property with the proxy interface used by a passive proxy.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieve all proxies

Retrieve all configured proxies and their interfaces.

Request:

```
{
   "jsonrpc": "2.0",
   "method": "proxy.get",
   "params": {
      "output": "extend",
      "selectInterface": "extend"
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```

Response:
See also
- Host
- Proxy interface

Source
CProxy::get() in ui/include/classes/api/services/CProxy.php.

proxy.update

Description

object proxy.update(object/array proxies)

This method allows to update existing proxies.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Proxy properties to be updated.
The `proxyid` property must be defined for each proxy, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard proxy properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hosts</td>
<td>array</td>
<td><em>Hosts</em> to be monitored by the proxy. If a host is already monitored by a different proxy, it will be reassigned to the current proxy. The hosts must have the <code>hostid</code> property defined.</td>
</tr>
<tr>
<td>interface</td>
<td>object</td>
<td>Host <em>interface</em> to replace the existing interface for the passive proxy.</td>
</tr>
</tbody>
</table>

Return values

*(object)* Returns an object containing the IDs of the updated proxies under the `proxyids` property.

Examples

Change hosts monitored by a proxy

Update the proxy to monitor the two given hosts.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "proxy.update",
   "params": {
      "proxyid": "10293",
      "hosts": [
         {
            "hostid": "10294"
         },
         {
            "hostid": "10295"
         }
      ]
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "proxyids": [
         "10293"
      ]
   },
   "id": 1
}
```

Change proxy status

Change the proxy to an active proxy and rename it to "Active proxy".

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "proxy.update",
   "params": {
      "proxyid": "10293",
      "host": "Active proxy",
      "status": "5"
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```
Response:
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10293"
        ],
        "id": 1
    },
    "id": 1
}

See also
• Host
• Proxy interface

Source
CProxy::update() in ui/include/classes/api/services/CProxy.php.

Regular expression

This class is designed to work with global regular expressions.

Object references:

• Regular expression

Available methods:

• regexp.create - creating new regular expressions
• regexp.delete - deleting regular expressions
• regexp.get - retrieving regular expressions
• regexp.update - updating regular expressions

> Regular expression object

The following objects are directly related to the regexp API.

Regular expression

The global regular expression object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>regexpid</td>
<td>string</td>
<td>(readonly) ID of the regular expression.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the regular expression.</td>
</tr>
<tr>
<td>test_string</td>
<td>string</td>
<td>Test string.</td>
</tr>
</tbody>
</table>

Expressions object

The expressions object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expression</td>
<td>string</td>
<td>Regular expression.</td>
</tr>
</tbody>
</table>
### regexp.create

**Description**

Object `regexp.create(object/array regularExpressions)`

This method allows to create new global regular expressions.

This method is only available to Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

**(object/array) Regular expressions to create.**

Additionally to the standard properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expressions</td>
<td>array</td>
<td>Expressions options.</td>
</tr>
</tbody>
</table>

**Return values**

**(object) Returns an object containing the IDs of the created regular expressions under the regexpids property.**

**Examples**

Creating a new global regular expression.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "regexp.create",
    "params": {
        "name": "Storage devices for SNMP discovery",
        "test_string": "/boot",
        "expressions": [
            {
                "expression": "~/\(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space\)$",
                "expression_type": "4",
                "case_sensitive": "1"
            }
        ]
    }
}
```
CRegexp::create() in ui/include/classes/api/services/CRegexp.php.

**regexp.delete**

**Description**

object regexp.delete(array regexpids)

This method allows to delete global regular expressions.

This method is only available to Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(array) IDs of the regular expressions to delete.

**Return values**

(object) Returns an object containing the IDs of the deleted regular expressions under the regexpids property.

**Examples**

Deleting multiple global regular expressions.

**Request:**

```json
{
   "jsonrpc": "2.0",
   "method": "regexp.delete",
   "params": [
      "16",
      "17"
   ],
   "auth": "3a57200802b24cda67c4e4010b50c065",
   "id": 1
}
```

**Response:**

```json
{
   "jsonrpc": "2.0",
   "result": {
      "regexpids": [
         "16",
         "17"
      ],
   },
   "id": 1
}
```
**regexp.get**

**Description**

integer/array regexp.get(object parameters)

The method allows to retrieve global regular expressions according to the given parameters.

This method is available only to Super Admin. Permissions to call the method can be revoked in user role settings. See **User roles** for more information.

**Parameters**

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>regexpids</td>
<td>string/array</td>
<td>Return only regular expressions with the given IDs.</td>
</tr>
<tr>
<td>selectExpressions</td>
<td>query</td>
<td>Return a expressions property.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Return values**

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

**Examples**

Retrieving global regular expressions.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "regexp.get",
    "params": {
        "output": ["regexpid", "name"],
        "selectExpressions": ["expression", "expression_type"],
        "regexpids": [1, 2],
        "preservekeys": true
    },
    "auth": "038e1d7b1735c6a5436ec9eae095879e",
    "id": 1
}
```

**Response:**

1163
Description

object regexp.update(object/array regularExpressions)

This method allows to update existing global regular expressions.

This method is only available to Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Regular expression properties to be updated.
The `regexpid` property must be defined for each object, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expressions</td>
<td>array</td>
<td>Expressions options</td>
</tr>
</tbody>
</table>

Return values

*(object)* Returns an object containing the IDs of the updated regular expressions under the `regexpids` property.

Examples

Updating global regular expression for file systems discovery.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "regexp.update",
    "params": {
        "regexpid": "1",
        "name": "File systems for discovery",
        "test_string": "",
        "expressions": [
            {
                "expression": "^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|zfs)$",
                "expression_type": "3",
                "exp_delimiter": ",",
                "case_sensitive": "0"
            },
            {
                "expression": "^(ntfs|fat32|fat16)$",
                "expression_type": "3",
                "exp_delimiter": ",",
                "case_sensitive": "0"
            }
        ]
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "regexpids": [
            "1"
        ],
        "id": 1
    }
}
```

Source

CRegexp::update() in ui/include/classes/api/services/CRegexp.php.

**Report**

This class is designed to work with scheduled reports.

Object references:
• Report
• Users
• User groups

Available methods:

• report.create - create new scheduled reports
• report.delete - delete scheduled reports
• report.get - retrieve scheduled reports
• report.update - update scheduled reports

> Report object

The following objects are directly related to the report API.

Report

The report object has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reportid</td>
<td>string</td>
<td>(readonly) ID of the report.</td>
</tr>
<tr>
<td>userid</td>
<td>string</td>
<td>ID of the user who created the report.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Unique name of the report.</td>
</tr>
<tr>
<td>dashboardid</td>
<td>string</td>
<td>ID of the dashboard that the report is based on.</td>
</tr>
<tr>
<td>period</td>
<td>integer</td>
<td>Period for which the report will be prepared.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) previous day; 1 - previous week; 2 - previous month; 3 - previous year.</td>
</tr>
<tr>
<td>cycle</td>
<td>integer</td>
<td>Period repeating schedule.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) daily; 1 - weekly; 2 - monthly; 3 - yearly.</td>
</tr>
<tr>
<td>start_time</td>
<td>integer</td>
<td>Time of the day, in seconds, when the report will be prepared for sending.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.</td>
</tr>
<tr>
<td>weekdays</td>
<td>integer</td>
<td>Days of the week for sending the report.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required for weekly reports only.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Days of the week are stored in binary form with each bit representing the corresponding week day. For example, 12 equals 1100 in binary and means that reports will be sent every Wednesday and Thursday.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 0.</td>
</tr>
<tr>
<td>active_since</td>
<td>string</td>
<td>On which date to start.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: empty string - (default) not specified (stored as 0); specific date in YYYY-MM-DD format (stored as a timestamp of the beginning of a day (00:00:00)).</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>active_till</td>
<td>string</td>
<td>On which date to end.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>empty string - (default) not specified (stored as 0);</td>
</tr>
<tr>
<td></td>
<td></td>
<td>specific date in YYYY-MM-DD format (stored as a timestamp of the end of a day (23:59:59)).</td>
</tr>
<tr>
<td>subject</td>
<td>string</td>
<td>Report message subject.</td>
</tr>
<tr>
<td>message</td>
<td>string</td>
<td>Report message text.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Whether the report is enabled or disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Disabled;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) Enabled.</td>
</tr>
<tr>
<td>description</td>
<td>text</td>
<td>Description of the report.</td>
</tr>
<tr>
<td>state</td>
<td>integer</td>
<td>(readonly) State of the report.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) report was not yet processed;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - report was generated and successfully sent to all recipients;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - report generating failed; &quot;info&quot; contains error information;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - report was generated, but sending to some (or all) recipients failed;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;info&quot; contains error information.</td>
</tr>
<tr>
<td>lastsent</td>
<td>timestamp</td>
<td>(readonly) Unix timestamp of the last successfully sent report.</td>
</tr>
<tr>
<td>info</td>
<td>string</td>
<td>(readonly) Error description or additional information.</td>
</tr>
</tbody>
</table>

**Users**

The users object has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userid</td>
<td>string</td>
<td>ID of user to send the report to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(required)</td>
</tr>
<tr>
<td>access_userid</td>
<td>string</td>
<td>ID of user on whose behalf the report will be generated.</td>
</tr>
<tr>
<td>exclude</td>
<td>integer</td>
<td>Whether to exclude the user from mailing list.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Include;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Exclude.</td>
</tr>
</tbody>
</table>

**User groups**

The user groups object has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usgrpid</td>
<td>string</td>
<td>ID of user group to send the report to.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(required)</td>
</tr>
<tr>
<td>access_userid</td>
<td>string</td>
<td>ID of user on whose behalf the report will be generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Generate report by recipient.</td>
</tr>
</tbody>
</table>

**report.create**

Description

object report.create(object/array reports)

This method allows to create new scheduled reports.
This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Scheduled reports to create.

Additionally to the standard scheduled report properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>users</td>
<td>object/array of objects</td>
<td>Users to send the report to.</td>
</tr>
<tr>
<td>user_groups</td>
<td>object/array of objects</td>
<td>User groups to send the report to.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the created scheduled reports under the reportids property. The order of the returned IDs matches the order of the passed scheduled reports.

Examples

Creating a scheduled report

Create a weekly report that will be prepared for the previous week every Monday-Friday at 12:00 from 2021-04-01 to 2021-08-31.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "report.create",
    "params": {
        "userid": "1",
        "name": "Weekly report",
        "dashboardid": "1",
        "period": "1",
        "cycle": "1",
        "start_time": "43200",
        "weekdays": "31",
        "active_since": "2021-04-01",
        "active_till": "2021-08-31",
        "subject": "Weekly report",
        "message": "Report accompanying text",
        "status": "1",
        "description": "Report description",
        "users": [
            {
                "userid": "1",
                "access_userid": "1",
                "exclude": "0"
            },
            {
                "userid": "2",
                "access_userid": "0",
                "exclude": "1"
            }
        ],
        "user_groups": [
            {
                "usrgrpid": "7",
                "access_userid": "0"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{} 1168
```
report.delete

Description

object report.delete(array reportids)

This method allows to delete scheduled reports.

This method is only available to Admin and Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the scheduled reports to delete.

Return values

(object) Returns an object containing the IDs of the deleted scheduled reports under the reportids property.

Examples

Deleting multiple scheduled reports

Delete two scheduled reports.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "report.delete",
    "params": [
        "1",
        "2"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "reportids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```
### report.get

**Description**

integer/array report.get(object parameters)

The method allows to retrieve scheduled reports according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

**Parameters**

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reportids</td>
<td>string/array</td>
<td>Return only scheduled reports with the given report IDs.</td>
</tr>
<tr>
<td>expired</td>
<td>boolean</td>
<td>If set to true returns only expired scheduled reports, if false - only active scheduled reports.</td>
</tr>
<tr>
<td>selectUsers</td>
<td>query</td>
<td>Return a <code>users</code> property the report is configured to be sent to.</td>
</tr>
<tr>
<td>selectUserGroups</td>
<td>query</td>
<td>Return a <code>user_groups</code> property the report is configured to be sent to.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Possible values are: reportid, name, status. These parameters being common for all get methods are described in detail in the reference commentary page.</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Return values**

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

**Examples**

**Retrieving report data**

**Request:**

```json
{
   "jsonrpc": "2.0",
   "method": "report.get",
   "params": [
      "output": "extend",
      "selectUsers": "extend",
      "selectUserGroups": "extend",
      "reportids": ["1", "2"]
   ],
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```
"access_userid": "1",
"exclude": "0"
],
"user_groups": []
},
"id": 1
}

See also

- Users
- User groups

Source

CReport::get() in ui/include/classes/api/services/CReport.php.

**report.update**

**Description**

object report.update(object/array reports)

This method allows to update existing scheduled reports.

This method is only available to Admin and Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) Scheduled report properties to be updated.

The reportid property must be defined for each scheduled report, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard scheduled report properties the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>users</td>
<td>object/array of objects</td>
<td>Users to replace the current users assigned to the scheduled report.</td>
</tr>
<tr>
<td>user_groups</td>
<td>object/array of objects</td>
<td>User groups to replace the current user groups assigned to the scheduled report.</td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns an object containing the IDs of the updated scheduled reports under the reportids property.

**Examples**

Disabling scheduled report

**Request:**

```
{
    "jsonrpc": "2.0",
    "method": "report.update",
    "params": {
        "reportid": "1",
        "status": "0"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

**Response:**

1172
Role

This class is designed to work with user roles.

Object references:

- Role
- Role rules
- UI element
- Service
- Service tag
- Module
- Action

Available methods:

- role.create - create new user roles
- role.delete - delete user roles
- role.get - retrieve user roles
- role.update - update user roles

> Role object

The following objects are directly related to the role API.

Role

The role object has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>roleid</td>
<td>string</td>
<td>(readonly) ID of the role.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the role.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>User type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) User;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Admin;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - Super admin.</td>
</tr>
<tr>
<td>readonly</td>
<td>integer</td>
<td>(readonly) Whether the role is readonly.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) No;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Yes.</td>
</tr>
</tbody>
</table>
Role rules

The role rules object has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ui</td>
<td>array</td>
<td>Array of the UI element objects.</td>
</tr>
<tr>
<td>ui.default_access</td>
<td>integer</td>
<td>Whether access to new UI elements is enabled.</td>
</tr>
<tr>
<td>services.read.mode</td>
<td>integer</td>
<td>Read-only access to services.</td>
</tr>
<tr>
<td>services.read.list</td>
<td>array</td>
<td>Array of Service objects.</td>
</tr>
<tr>
<td>services.read.tag</td>
<td>object</td>
<td>The specified services, including child services, will be granted a read-only access to the user role. Read-only access will not override read-write access to the services. Only used if services.read.mode is set to 0.</td>
</tr>
<tr>
<td>services.write.mode</td>
<td>integer</td>
<td>Read-write access to services.</td>
</tr>
<tr>
<td>services.write.list</td>
<td>array</td>
<td>Array of Service objects.</td>
</tr>
<tr>
<td>services.write.tag</td>
<td>object</td>
<td>The tag matched services, including child services, will be granted a read-only access to the user role. Read-only access will not override read-write access to the services. Only used if services.write.mode is set to 0.</td>
</tr>
<tr>
<td>modules</td>
<td>array</td>
<td>Array of the module objects.</td>
</tr>
<tr>
<td>modules.default_access</td>
<td>integer</td>
<td>Whether access to new modules is enabled.</td>
</tr>
</tbody>
</table>

Possible values:

- **0** - Disabled;
- **1** - (default) Enabled.

Possible values:

- **0** - Read-only access to the services, specified by the services.read.list or matched by the services.read.tag properties.
- **1** - (default) Read-only access to all services.

Possible values:

- **0** - (default) Read-write access to the services, specified by the services.write.list or matched by the services.write.tag properties.
- **1** - Read-write access to all services.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>api.access</td>
<td>integer</td>
<td>Whether access to API is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Disabled;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) Enabled.</td>
</tr>
<tr>
<td>api.mode</td>
<td>integer</td>
<td>Mode for treating API methods listed in the api property.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Deny list;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Allow list.</td>
</tr>
<tr>
<td>api</td>
<td>array</td>
<td>Array of API methods.</td>
</tr>
<tr>
<td>actions</td>
<td>array</td>
<td>Array of the action objects.</td>
</tr>
<tr>
<td>actions.default_access</td>
<td>integer</td>
<td>Whether access to new actions is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Disabled;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) Enabled.</td>
</tr>
</tbody>
</table>

**UI element**

The UI element object has the following properties:
### Property

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name</strong></td>
<td>string</td>
<td>Name of the UI element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for users of any type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>monitoring.dashboard - Monitoring → Dashboard;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>monitoring.problems - Monitoring → Problems;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>monitoring.hosts - Monitoring → Hosts;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>monitoring.overview - Monitoring → Overview;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>monitoring.latest_data - Monitoring → Latest data;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>monitoring.maps - Monitoring → Maps;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>monitoring.services - Monitoring → Services;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>inventory.overview - Inventory → Overview;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>inventory.hosts - Inventory → Hosts;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reports.availability_report - Reports → Availability report;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reports.top_triggers - Reports → Triggers top 100.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values only for users of Admin and Super admin user types:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>monitoring.discovery - Monitoring → Discovery;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reports.scheduled_reports - Reports → Scheduled reports;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reports.notifications - Reports → Notifications;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>configuration.host_groups - Configuration → Host groups;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>configuration.templates - Configuration → Templates;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>configuration.hosts - Configuration → Hosts;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>configuration.maintenance - Configuration → Maintenance;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>configuration.actions - Configuration → Actions;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>configuration.discovery - Configuration → Discovery.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values only for users of Super admin user type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reports.system_info - Reports → System information;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reports.audit - Reports → Audit;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reports.action_log - Reports → Action log;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>configuration.event_correlation - Configuration → Event correlation;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>administration.general - Administration → General;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>administration.proxies - Administration → Proxies;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>administration.authentication - Administration → Authentication;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>administration.user_groups - Administration → User groups;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>administration.user_roles - Administration → User roles;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>administration.users - Administration → Users;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>administration.media_types - Administration → Media types;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>administration.scripts - Administration → Scripts;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>administration.queue - Administration → Queue.</td>
</tr>
<tr>
<td><strong>status</strong></td>
<td>integer</td>
<td>Whether access to the UI element is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Disabled;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) Enabled.</td>
</tr>
</tbody>
</table>

### Service

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>serviceid</strong></td>
<td>string</td>
<td>ID of the Service.</td>
</tr>
</tbody>
</table>

### Service tag

1176
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Tag name. If empty string is specified, the service tag will not be used for service matching.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Tag value. If no value or empty string is specified, only the tag name will be used for service matching.</td>
</tr>
</tbody>
</table>

### Module

The module object has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>moduleid</td>
<td>string</td>
<td>ID of the module.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Whether access to the module is enabled. Possible values: 0 - Disabled; 1 - (default) Enabled.</td>
</tr>
</tbody>
</table>

### Action

The action object has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the action. Possible values for users of any type: edit_dashboards - Create and edit dashboards; edit_maps - Create and edit maps; add_problem_comments - Add problem comments; change_severity - Change problem severity; acknowledge_problems - Acknowledge problems; close_problems - Close problems; execute_scripts - Execute scripts; manage_api_tokens - Manage API tokens. Possible values only for users of Admin and Super admin user types: edit_maintenance - Create and edit maintenances; manage_scheduled_reports - Manage scheduled reports. Possible values only for users of User and Admin user types: invoke_execute_now - allows to execute item checks for users that have only read permissions on host.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Whether access to perform the action is enabled. Possible values: 0 - Disabled; 1 - (default) Enabled.</td>
</tr>
</tbody>
</table>

### role.create

Description

object role.create(object/array roles)

This method allows to create new roles.
This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Roles to create.

Additionally to the standard role properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rules</td>
<td>array</td>
<td>Role rules to be created for the role.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the created roles under the roleids property. The order of the returned IDs matches the order of the passed roles.

Examples

Creating a role

Create a role with type "User" and denied access to two UI elements.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "role.create",
    "params": {
        "name": "Operator",
        "type": "1",
        "rules": {
            "ui": [
                {
                    "name": "monitoring.hosts",
                    "status": "0"
                },
                {
                    "name": "monitoring.maps",
                    "status": "0"
                }
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "roleids": [
            "5"
        ]
    },
    "id": 1
}
```

See also

- Role rules
- UI element
- Module
- Action

Source
CRole::create() in ui/include/classes/api/services/CRole.php.

**role.delete**

Description

object role.delete(array roleids)

This method allows to delete roles.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the roles to delete.

Return values

(object) Returns an object containing the IDs of the deleted roles under the roleids property.

Examples

Deleting multiple user roles

Delete two user roles.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "role.delete",
    "params": [
        "4",
        "5"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "roleids": [
            "4",
            "5"
        ]
    },
    "id": 1
}
```

Source

CRole::delete() in ui/include/classes/api/services/CRole.php.

**role.get**

Description

integer/array role.get(object parameters)

The method allows to retrieve roles according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.
The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>roleids</td>
<td>string/array</td>
<td>Return only roles with the given IDs.</td>
</tr>
<tr>
<td>selectRules</td>
<td>query</td>
<td>Return role rules in the rules property.</td>
</tr>
<tr>
<td>selectUsers</td>
<td>query</td>
<td>Select users this role is assigned to.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
</tbody>
</table>

Possible values are: roleid, name.

countOutput   boolean   These parameters being common for all get methods are described in detail in the reference commentary page.

editable      boolean   excludeSearch  boolean   filter     object   limit      integer   output     query   preservekeys boolean   search     object   searchByAny boolean   searchWildcardsEnabled boolean   sortorder  string/array   startSearch  boolean

Return values
(integer/array) Returns either:
- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving role data

Retrieve "Super admin role" role data and its access rules.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "role.get",
   "params": {
      "output": "extend",
      "selectRules": "extend",
      "roleids": "3"
   },
   "auth": "3a57200802b24cda67c4e4010b50c065",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": [
      {
         "roleid": "3",
         "name": "Super admin role",
         "type": "3",
         "readonly": "1",
         "rules": {
            "ui": [
               {
                  "name": "monitoring.dashboard",
                  "status": "1"
               }
            ]
         }
      }
   ]
}
```

1180
"status": "1",
},
{
    "name": "configuration.actions",
    "status": "1"
},
{
    "name": "configuration.discovery",
    "status": "1"
},
{
    "name": "reports.system_info",
    "status": "1"
},
{
    "name": "reports.audit",
    "status": "1"
},
{
    "name": "reports.action_log",
    "status": "1"
},
{
    "name": "configuration.event_correlation",
    "status": "1"
},
{
    "name": "administration.general",
    "status": "1"
},
{
    "name": "administration.proxies",
    "status": "1"
},
{
    "name": "administration.authentication",
    "status": "1"
},
{
    "name": "administration.user_groups",
    "status": "1"
},
{
    "name": "administration.user_roles",
    "status": "1"
},
{
    "name": "administration.users",
    "status": "1"
},
{
    "name": "administration.media_types",
    "status": "1"
},
{
    "name": "administration.scripts",
    "status": "1"
},
{
    "name": "administration.queue",
    "status": "1"
}
"ui.default_access": "1",
"services.read.mode": "1",
"services.read.list": [],
"services.read.tag": {
  "tag": "",
  "value": ""
},
"services.write.mode": "1",
"services.write.list": [],
"services.write.tag": {
  "tag": "",
  "value": ""
},
"modules": [],
"modules.default_access": "1",
"api.access": "1",
"api.mode": "0",
"api": [],
"actions": [
  {
    "name": "edit_dashboards",
    "status": "1"
  },
  {
    "name": "edit_maps",
    "status": "1"
  },
  {
    "name": "acknowledge_problems",
    "status": "1"
  },
  {
    "name": "close_problems",
    "status": "1"
  },
  {
    "name": "change_severity",
    "status": "1"
  },
  {
    "name": "add_problem_comments",
    "status": "1"
  },
  {
    "name": "execute_scripts",
    "status": "1"
  },
  {
    "name": "manage_api_tokens",
    "status": "1"
  },
  {
    "name": "edit_maintenance",
    "status": "1"
  },
  {
    "name": "manage_scheduled_reports",
    "status": "1"
  },
  {
    "name": "invoke_execute_now",
    "status": "1"
See also
- Role rules
- User

Source
CRole::get() in ui/include/classes/api/services/CRole.php.

**role.update**

Description

object role.update(object/array roles)

This method allows to update existing roles.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Role properties to be updated.

The roleid property must be defined for each role, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard role properties the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rules</td>
<td>array</td>
<td>Access rules to replace the current access rules assigned to the role.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated roles under the roleid property.

Examples

Disabling ability to execute scripts

Update role with ID "5", disable ability to execute scripts.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "role.update",
    "params": [
        {
            "roleid": "5",
            "rules": {
                "actions": [
                    {
                        "name": "execute_scripts",
                        "status": "0"
                    }
                ]
            }
        }
    ]
}
```
Limiting access to API

Update role with ID "5", deny to call any "create", "update" or "delete" methods.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "role.update",
    "params": [
        {
            "roleid": "5",
            "rules": {
                "api.access": "1",
                "api.mode": "0",
                "api": ["*.create", "*.update", "*.delete"]
            }
        }
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "roleids": ["5"]
    },
    "id": 1
}
```

Source

CRole::update() in ui/include/classes/api/services/CRole.php.

Script

This class is designed to work with scripts.

Object references:

- Script
- Webhook parameters
- Debug
- Log entry
Available methods:

- `script.create` - create new scripts
- `script.delete` - delete scripts
- `script.execute` - run scripts
- `script.get` - retrieve scripts
- `script.getscriptsbyhosts` - retrieve scripts for hosts
- `script.update` - update scripts

> Script object

The following objects are directly related to the script API.

Script

The script object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>scriptid</code></td>
<td>string</td>
<td>(readonly) ID of the script.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>string</td>
<td>Name of the script.</td>
</tr>
<tr>
<td><code>type</code></td>
<td>integer</td>
<td>Script type. Possible values: 0 - Script; 1 - iPMI; 2 - SSH; 3 - Telnet; 5 - (default) Webhook.</td>
</tr>
<tr>
<td><code>command</code></td>
<td>string</td>
<td>Command to run.</td>
</tr>
<tr>
<td><code>scope</code></td>
<td>integer</td>
<td>Script scope. Possible values: 1 - default action operation; 2 - manual host action; 4 - manual event action.</td>
</tr>
<tr>
<td><code>execute_on</code></td>
<td>integer</td>
<td>Where to run the script. Used if type is 0 (script). Possible values: 0 - run on Zabbix agent; 1 - run on Zabbix server; 2 - (default) run on Zabbix server (proxy).</td>
</tr>
<tr>
<td><code>menu_path</code></td>
<td>string</td>
<td>Folders separated by slash that form a menu like navigation in frontend when clicked on host or event. Used if scope is 2 or 4.</td>
</tr>
<tr>
<td><code>authtype</code></td>
<td>integer</td>
<td>Authentication method used for SSH script type. Used if type is 2. Possible values: 0 - password; 1 - public key.</td>
</tr>
<tr>
<td><code>username</code></td>
<td>string</td>
<td>User name used for authentication. Required if type is 2 or 3.</td>
</tr>
<tr>
<td><code>password</code></td>
<td>string</td>
<td>Password used for SSH scripts with password authentication and Telnet scripts. Used if type is 2 and authtype is 0 or type is 3.</td>
</tr>
<tr>
<td><code>publickey</code></td>
<td>string</td>
<td>Name of the public key file used for SSH scripts with public key authentication. Required if type is 2 and authtype is 1.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>privatekey</td>
<td>string</td>
<td>Name of the private key file used for SSH scripts with public key authentication. Required if type is 2 and auth type is 1.</td>
</tr>
<tr>
<td>port</td>
<td>string</td>
<td>Port number used for SSH and Telnet scripts. Used if type is 2 or 3.</td>
</tr>
<tr>
<td>groupid</td>
<td>string</td>
<td>ID of the host group that the script can be run on. If set to 0, the script will be available on all host groups. Default: 0.</td>
</tr>
<tr>
<td>usgrpvid</td>
<td>string</td>
<td>ID of the user group that will be allowed to run the script. If set to 0, the script will be available for all user groups. Used if scope is 2 or 4. Default: 0.</td>
</tr>
<tr>
<td>host_access</td>
<td>integer</td>
<td>Host permissions needed to run the script. Used if scope is 2 or 4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 2 - (default) read; 3 - write.</td>
</tr>
<tr>
<td>confirmation</td>
<td>string</td>
<td>Confirmation pop up text. The pop up will appear when trying to run the script from the Zabbix frontend. Used if scope is 2 or 4.</td>
</tr>
<tr>
<td>timeout</td>
<td>string</td>
<td>Webhook script execution timeout in seconds. Time suffixes are supported, e.g. 30s, 1m. Required if type is 5. Possible values: 1-60s Default value: 30s</td>
</tr>
<tr>
<td>parameters</td>
<td>array</td>
<td>Array of webhook input parameters. Used if type is 5.</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Description of the script.</td>
</tr>
</tbody>
</table>

**Webhook parameters**

Parameters passed to webhook script when it is called have the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Parameter name. (required)</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Parameter value. Supports macros.</td>
</tr>
</tbody>
</table>

**Debug**

Debug information of executed webhook script. The debug object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logs</td>
<td>array</td>
<td>Array of log entries.</td>
</tr>
<tr>
<td>ms</td>
<td>string</td>
<td>Script execution duration in milliseconds.</td>
</tr>
</tbody>
</table>

**Log entry**

The log entry object has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>level</td>
<td>integer</td>
<td>Log level.</td>
</tr>
<tr>
<td>ms</td>
<td>string</td>
<td>The time elapsed in milliseconds since the script was run before log entry was added.</td>
</tr>
<tr>
<td>message</td>
<td>string</td>
<td>Log message.</td>
</tr>
</tbody>
</table>

**script.create**

**Description**

object script.create(object/array scripts)

This method allows to create new scripts.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

**Parameters**

(object/array) Scripts to create.

The method accepts scripts with the standard script properties.

**Return values**

(object) Returns an object containing the IDs of the created scripts under the scriptids property. The order of the returned IDs matches the order of the passed scripts.

**Examples**

Create a webhook script

Create a webhook script that sends HTTP request to external service.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "script.create",
    "params": {
        "name": "Webhook script",
        "command": "try {
            var request = new HttpRequest(),
            response,
            data;

            request.addHeader('Content-Type: application/json');

            response = request.post('https://localhost/post', value);

            try {
                response = JSON.parse(response);
            } catch (error) {
                response = null;
            }

            if (request.getStatus() !== 200 || !('data' in response)) {
                throw 'Unexpected response.';
            }

            data = JSON.stringify(response.data);

            Zabbix.log(3, 'Webhook Script
            response data: ' + data);
        }
        ,
        {
            "name": "token",
            "value": "${WEBHOOK_TOKEN}"
        },
        {
            "name": "host",
            "value": "{HOST.HOST}"n
        },
        {
            "name": "v",
            "value": "2.2"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": {
```
Create a SSH script

Create a SSH script with public key authentication that can be executed on a host and has a context menu.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.create",
    "params": {
        "name": "SSH script",
        "command": "my script command",
        "type": 2,
        "username": "John",
        "publickey": "pub.key",
        "privatekey": "priv.key",
        "password": "secret",
        "port": "12345",
        "scope": 2,
        "menu_path": "All scripts/SSH",
        "usrgrpid": "7",
        "groupid": "4"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [5]
    },
    "id": 1
}
```

Create a custom script

Create a custom script that will reboot a server. The script will require write access to the host and will display a configuration message before running in the frontend.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.create",
    "params": {
        "name": "Reboot server",
        "command": "reboot server 1",
        "confirmation": "Are you sure you would like to reboot the server?",
        "scope": 2,
        "type": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```
Response:
{
   "jsonrpc": "2.0",
   "result": {
      "scriptids": [
         "4"
      ],
      "id": 1
   }
}

Source
CScript::create() in ui/include/classes/api/services/CScript.php.

script.delete

Description

object script.delete(array scriptIds)

This method allows to delete scripts.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the scripts to delete.

Return values

(object) Returns an object containing the IDs of the deleted scripts under the scriptids property.

Examples

Delete multiple scripts

Delete two scripts.

Request:
{
   "jsonrpc": "2.0",
   "method": "script.delete",
   "params": [
      "3",
      "4"
   ],
   "auth": "3a57200802b24cda67c4e4010b50c065",
   "id": 1
}

Response:
{
   "jsonrpc": "2.0",
   "result": {
      "scriptids": [
         "3",
         "4"
      ],
      "id": 1
   }
}

Source
CScript::delete() in ui/include/classes/api/services/CScript.php.
**script.execute**

Description

`object script.execute(object parameters)`

This method allows to run a script on a host or event.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object) Parameters containing the ID of the script to run and either the ID of the host or the ID of the event.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scriptid</td>
<td>string</td>
<td>ID of the script to run. (required)</td>
</tr>
<tr>
<td>hostid</td>
<td>string</td>
<td>ID of the host to run the script on.</td>
</tr>
<tr>
<td>eventdata</td>
<td>string</td>
<td>ID of the event to run the script on.</td>
</tr>
</tbody>
</table>

Return values

`(object) Returns the result of script execution.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>string</td>
<td>Whether the script was run successfully.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Possible values: success or failed. Script output.</td>
</tr>
<tr>
<td>debug</td>
<td>object</td>
<td>Contains a debug object if a webhook script is executed. For other script types, it contains empty object.</td>
</tr>
</tbody>
</table>

Examples

Run a webhook script

Run a webhook script that sends HTTP request to external service.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "script.execute",
    "params": {
        "scriptid": "4",
        "hostid": "30079"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "response": "success",
        "value": "{\"status\":\"sent\",\"timestamp\":\"1611235391\"}",
        "debug": {
            "logs": [
                {
                    "level": 3,
                    "ms": 480,
                    "message": "{Webhook Script} HTTP status: 200."
                }
            ]
        }
    }
}
```
Run a custom script
Run a "ping" script on a host.
Request:
```
{
    "jsonrpc": "2.0",
    "method": "script.execute",
    "params": {
        "scriptid": "1",
        "hostid": "30079"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```
Response:
```
{
    "jsonrpc": "2.0",
    "result": {
        "response": "success",
        "value": "PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data. 64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.074 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.030 ms
64 bytes from 127.0.0.1: icmp_req=3 ttl=64 time=0.030 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.030/0.044/0.074/0.022 ms",
        "debug": []
    },
    "id": 1
}
```
Source
CScript::execute() in ui/include/classes/api/services/CScript.php.

**script.get**

Description
integer/array script.get(object parameters)
The method allows to retrieve scripts according to the given parameters.
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(object) Parameters defining the desired output.
The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only scripts that can be run on the given host groups.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only scripts that can be run on the given hosts.</td>
</tr>
<tr>
<td>scriptids</td>
<td>string/array</td>
<td>Return only scripts with the given IDs.</td>
</tr>
<tr>
<td>usrgrpids</td>
<td>string/array</td>
<td>Return only scripts that can be run by users in the given user groups.</td>
</tr>
<tr>
<td>selectHostGroups</td>
<td>query</td>
<td>Return a host groups property with host groups that the script can be run on.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return a hosts property with hosts that the script can be run on.</td>
</tr>
<tr>
<td>selectActions</td>
<td>query</td>
<td>Return a actions property with actions that the script is associated with.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Possible values are: scriptid and name.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>selectGroups</td>
<td>query</td>
<td>This parameter is deprecated, please use selectHostGroups instead.</td>
</tr>
<tr>
<td>(deprecated)</td>
<td></td>
<td>Return a groups property with host groups that the script can be run on.</td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieve all scripts

Retrieve all configured scripts.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "script.get",
  "params": {
    "output": "extend"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": [
    {
      "scriptid": "1",
      "name": "Ping",
      "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
      "host_access": "2",
      "usrgrpid": "0",
      "groupid": "0",
      "description": ",",
      "confirmation": ",",
      "type": "0",
      "execute_on": "1",
      "timeout": "30s",
      "parameters": []
    },
    {
    }
  ]
}```
"scriptid": "2",
"name": "Traceroute",
"command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
"host_access": "2",
"usrgrpid": "0",
"groupid": "0",
"description": "",
"confirmation": "",
"type": "0",
"execute_on": "1",
"timeout": "30s",
"parameters": []
},
{
"scriptid": "3",
"name": "Detect operating system",
"command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
"host_access": "2",
"usrgrpid": "7",
"groupid": "0",
"description": "",
"confirmation": "",
"type": "0",
"execute_on": "1",
"timeout": "30s",
"parameters": []
},
{

"scriptid": "4",
"name": "Webhook",
"command": "try {\n var request = new HttpRequest(),\n response,\n data;\n\n request.addHeader
"host_access": "2",
"usrgrpid": "7",
"groupid": "0",
"description": "",
"confirmation": "",
"type": "5",
"execute_on": "1",
"timeout": "30s",
"parameters": [
{
"name": "token",
"value": "{$WEBHOOK.TOKEN}"
},
{
"name": "host",
"value": "{HOST.HOST}"
},
{
"name": "v",
"value": "2.2"
}
]
}
],
"id": 1
}
See also
• Host
• Host group
Source

1194


CScript::get() in ui/include/classes/api/services/CScript.php.

**script.getscriptsbyhosts**

**Description**

object script.getscriptsbyhosts(array hostIds)

This method allows to retrieve scripts available on the given hosts.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

**Parameters**

(string/array) IDs of hosts to return scripts for.

**Return values**

(object) Returns an object with host IDs as properties and arrays of available scripts as values.

The method will automatically expand macros in the confirmation text.

**Examples**

Retrieve scripts by host IDs

Retrieve all scripts available on hosts "30079" and "30073".

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "script.getscriptsbyhosts",
  "params": [
    "30079",
    "30073"
  ],
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": {
    "30079": [
      {
        "scriptid": "3",
        "name": "Detect operating system",
        "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
        "host_access": "2",
        "usrgrpid": "7",
        "groupid": "0",
        "description": "",
        "confirmation": "",
        "type": "0",
        "execute_on": "1",
        "hostid": "10001"
      },
      {
        "scriptid": "1",
        "name": "Ping",
        "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
        "host_access": "2",
        "usrgrpid": "0",
        "groupid": "0",
        "description": "",
        "confirmation": "",
        "id": 1
      }
    ]
  }
}```
script.update

Description

object script.update(object/array scripts)

This method allows to update existing scripts.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Script properties to be updated.

The scriptid property must be defined for each script, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. An exception is type property change from 5 (Webhook) to other: the parameters property will be cleaned.

Return values

(object) Returns an object containing the IDs of the updated scripts under the scriptids property.

Examples

Change script command

Change the command of the script to "/bin/ping -c 10 {HOST.CONN} 2>&1".

Request:

```
{
   "jsonrpc": "2.0",
   "method": "script.update",
   "params": {
      "scriptid": "1",
      "command": "/bin/ping -c 10 {HOST.CONN} 2>&1"
   },
   "auth": "038e1d7b1735c6a5436ee9eae096879e",
   "id": 1
}
```

Response:

```
{
   "jsonrpc": "2.0",
   "result": {
      "scriptids": [
         "1"
      ]
   },
   "id": 1
}
```

Source

CScript::getScriptsByHosts() in ui/include/classes/api/services/CScript.php.

Service

This class is designed to work with IT infrastructure/business services.

Object references:

- Service
- Status rule
• Service tag
• Service alarm
• Problem tag

Available methods:

• service.create - creating new services
• service.delete - deleting services
• service.get - retrieving services
• service.update - updating services

> Service object

The following objects are directly related to the service API.

Service

The service object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceid</td>
<td>string</td>
<td>(readonly) ID of the service.</td>
</tr>
<tr>
<td>algorithm</td>
<td>integer</td>
<td>Status calculation rule. Only applicable if child services exist.</td>
</tr>
<tr>
<td></td>
<td>(required)</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the service.</td>
</tr>
<tr>
<td></td>
<td>(required)</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>integer</td>
<td>Position of the service used for sorting.</td>
</tr>
<tr>
<td></td>
<td>(required)</td>
<td></td>
</tr>
<tr>
<td>weight</td>
<td>integer</td>
<td>Service weight.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(required)</td>
<td></td>
</tr>
<tr>
<td>propagativeredge</td>
<td>integer</td>
<td>Status propagation rule. Must be set together with propagation_value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>propagativevalue</td>
<td>integer</td>
<td>Status propagation value. Must be set together with propagation_rule.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Possible values for propagation_rule with values 0 and 3: 0.

Possible values for propagation_rule with values 1 and 2: 1-5.

Possible values for propagation_rule with value 4:
-1 - OK;
0 - Not classified;
1 - Information;
2 - Warning;
3 - Average;
4 - High;
5 - Disaster.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>integer</td>
<td>(readonly) Whether the service is in OK or problem state.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the service is in problem state, status is equal either to:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the severity of the most critical problem;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- the highest status of a child service in problem state.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the service is in OK state, status is equal to -1.</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Description of the service.</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Universal unique identifier. For update operations this field is readonly.</td>
</tr>
<tr>
<td>created</td>
<td>integer</td>
<td>Unix timestamp when service was created. (readonly)</td>
</tr>
<tr>
<td>readonly</td>
<td>boolean</td>
<td>(readonly) Access to the service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Read-write;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Read-only.</td>
</tr>
</tbody>
</table>

**Status rule**

The status rule object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>integer</td>
<td>Condition for setting (New status) status.</td>
</tr>
<tr>
<td></td>
<td>(required)</td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - if at least (N) child services have (Status) status or above;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - if at least (N%) of child services have (Status) status or above;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - if less than (N) child services have (Status) status or below;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - if less than (N%) of child services have (Status) status or below;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - if weight of child services with (Status) status or above is at least (W);</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - if weight of child services with (Status) status or above is at least (N%);</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 - if weight of child services with (Status) status or below is less than (W);</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 - if weight of child services with (Status) status or below is less than (N%);</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Where:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- N (W) is limit_value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- (Status) is limit_status;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- (New status) is new_status.</td>
</tr>
<tr>
<td>limit_value</td>
<td>integer</td>
<td>Limit value.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- for N and W: 1-100000;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- for N%: 1-100.</td>
</tr>
<tr>
<td>limit_status</td>
<td>integer</td>
<td>Limit status.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1 - OK;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - Not classified;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Information;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Warning;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - Average;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - High;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - Disaster.</td>
</tr>
</tbody>
</table>
**Property** | **Type** | **Description**
--- | --- | ---
new_status | integer | New status value. Possible values:
0 - Not classified;
1 - Information;
2 - Warning;
3 - Average;
4 - High;
5 - Disaster.

**Service tag**

The service tag object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Service tag name.</td>
</tr>
<tr>
<td>(required) value</td>
<td>string</td>
<td>Service tag value.</td>
</tr>
</tbody>
</table>

**Service alarm**

Service alarms cannot be directly created, updated or deleted via the Zabbix API.

The service alarm objects represent a service's state change. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock</td>
<td>timestamp</td>
<td>Time when the service state change has happened.</td>
</tr>
<tr>
<td>value</td>
<td>integer</td>
<td>Status of the service.</td>
</tr>
</tbody>
</table>

Refer to the service status property for a list of possible values.

**Problem tag**

Problem tags allow linking services with problem events. The problem tag object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Problem tag name.</td>
</tr>
</tbody>
</table>
| (required) operator | integer | Mapping condition operator. Possible values:
0 - (default) equals;
2 - like. |
| value | string | Problem tag value. |

**service.create**

Description

object service.create(object/array services)

This method allows to create new services.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) services to create.

Additionally to the standard service properties, the method accepts the following parameters.
Parameter | Type | Description
--- | --- | ---
children | array | Child services to be linked to the service.
The children must have the serviceid property defined.
parents | array | Parent services to be linked to the service.
The parents must have the serviceid property defined.
tags | array | Service tags to be created for the service.
problem_tags | array | Problem tags to be created for the service.
status_rules | array | Status rules to be created for the service.

Return values

(object) Returns an object containing the IDs of the created services under the serviceids property. The order of the returned IDs matches the order of the passed services.

Examples

Creating a service

Create a service that will be switched to problem state, if at least one child has a problem.

Request:

```
{   "jsonrpc": "2.0",
    "method": "service.create",
    "params": {
        "name": "Server 1",
        "algorithm": 1,
        "sortorder": 1
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{   "jsonrpc": "2.0",
    "result": {
        "serviceids": [
        "5"
    ],
    "id": 1
}
```

Source

CService::create() in ui/include/classes/api/services/CService.php.

**service.delete**

Description

object service.delete(array serviceIds)

This method allows to delete services. This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the services to delete.

Return values
(object) Returns an object containing the IDs of the deleted services under the serviceids property.

Examples

Deleting multiple services

Delete two services.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "service.delete",
    "params": [
        "4",
        "5"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "4",
            "5"
        ]
    },
    "id": 1
}
```

Source

CService::delete() in ui/include/classes/api/services/CService.php.

**service.get**

**Description**

integer/array service.get(object parameters)

The method allows to retrieve services according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceids</td>
<td>string/array</td>
<td>Return only services with the given IDs.</td>
</tr>
<tr>
<td>parentids</td>
<td>string/array</td>
<td>Return only services that are linked to the given parent services.</td>
</tr>
<tr>
<td>deep_parentids</td>
<td>string/array</td>
<td>Return all direct and indirect child services. Used together with parentids.</td>
</tr>
<tr>
<td>childids</td>
<td>string/array</td>
<td>Return only services that are linked to the given child services.</td>
</tr>
<tr>
<td>evaltype</td>
<td>integer</td>
<td>Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>tags</td>
<td>object/array of objects</td>
<td>Return only services with given tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value. Format: [{&quot;tag&quot;: &quot;&lt;tag&gt;&quot;, &quot;value&quot;: &quot;&lt;value&gt;&quot;, &quot;operator&quot;: &quot;&lt;operator&gt;&quot;}, ...]. An empty array returns all services. Possible operator values: 0 - (default) Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist.</td>
</tr>
<tr>
<td>problem_tags</td>
<td>object/array of objects</td>
<td>Return only services with given problem tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value. Format: [{&quot;tag&quot;: &quot;&lt;tag&gt;&quot;, &quot;value&quot;: &quot;&lt;value&gt;&quot;, &quot;operator&quot;: &quot;&lt;operator&gt;&quot;}, ...]. An empty array returns all services. Possible operator values: 0 - (default) Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist.</td>
</tr>
<tr>
<td>without_problem_tags</td>
<td>Return only services without problem tags.</td>
<td></td>
</tr>
<tr>
<td>slais</td>
<td>string/array</td>
<td>Return only services that are linked to the specific SLA(s).</td>
</tr>
<tr>
<td>selectChildren</td>
<td></td>
<td>Return a children property with the child services. Supports count.</td>
</tr>
<tr>
<td>selectParents</td>
<td></td>
<td>Return a parents property with the parent services. Supports count.</td>
</tr>
<tr>
<td>selectTagQuery</td>
<td></td>
<td>Return a tags property with service tags. Supports count.</td>
</tr>
<tr>
<td>selectProblemEvents</td>
<td>Return a problem_events property with an array of problem event objects. The problem event object has the following properties: eventid - (string) Event ID; severity - (string) Current event severity; name - (string) Resolved event name. Supports count.</td>
<td></td>
</tr>
<tr>
<td>selectProblemTags</td>
<td>Return a problem_tags property with problem tags. Supports count.</td>
<td></td>
</tr>
<tr>
<td>selectStatusRules</td>
<td>Return a status_rules property with status rules. Supports count.</td>
<td></td>
</tr>
<tr>
<td>selectStatusTimeline</td>
<td>Return a status_timeline property containing service state changes for given periods. Format [{&quot;period_from&quot;: &quot;&lt;period_from&gt;&quot;, &quot;period_to&quot;: &quot;&lt;period_to&gt;&quot;}, ...] - period_from being a starting date (inclusive; integer timestamp) and period_to being an ending date (exclusive; integer timestamp) for the period you’re interested in. Returns an array of entries containing a start_value property and an alarms array for the state changes within specified periods.</td>
<td></td>
</tr>
</tbody>
</table>
**sortfield**  |  string/array  | Sort the result by the given properties.

Possible values are: `serviceid`, `name`, `status`, `sortorder` and `created_at`.

| edit | boolean  | These parameters being common for all `get` methods are described in detail in the reference commentary.
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
</tr>
<tr>
<td>output</td>
<td>query</td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
</tr>
<tr>
<td>search</td>
<td>object</td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving all services

Retrieve all data about all services and their dependencies.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "service.get",
    "params": {
        "output": "extend",
        "selectChildren": "extend",
        "selectParents": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "serviceid": "1",
            "name": "My Service - 0001",
            "status": "-1",
            "algorithm": "2",
            "sortorder": "0",
            "weight": "0",
            "propagation_rule": "0",
            "propagation_value": "0",
            "description": "My Service Description 0001.",
            "uuid": "dfa4d8eaa754e3a95c04d6029182681",
            "created_at": "946684800",
            "readonly": false,
            "parents": [],
            "children": []
        }
    ]
}
```
"serviceid": "2",
"name": "My Service - 0002",
"status": "-1",
"algorithm": "2",
"sortorder": "0",
"weight": "0",
"propagation_rule": "0",
"description": "My Service Description 0002.",
"uuid": "20ea0d85212841219130abeaca28c065",
"created_at": "946684800",
"readonly": false,
"parents": [],
"children": []
}

Source
CService::get() in ui/include/classes/api/services/CService.php.

**service.update**

**Description**
object service.update(object/array services)

This method allows to update existing services.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**
(object/array) service properties to be updated.

The serviceid property must be defined for each service, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard service properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>children</td>
<td>array</td>
<td>Child services to replace the current service children.</td>
</tr>
<tr>
<td>parents</td>
<td>array</td>
<td>The parents must have the serviceid property defined.</td>
</tr>
<tr>
<td>tags</td>
<td>array</td>
<td>Service tags to replace the current service tags.</td>
</tr>
<tr>
<td>problem_tags</td>
<td>array</td>
<td>Problem tags to replace the current problem tags.</td>
</tr>
<tr>
<td>status_rules</td>
<td>array</td>
<td>Status rules to replace the current status rules.</td>
</tr>
</tbody>
</table>

**Return values**
(object) Returns an object containing the IDs of the updated services under the serviceids property.

**Examples**
Setting the parent for a service

Make service with ID "3" to be the parent for service with ID "5".

Request:
Adding a scheduled downtime

Add a downtime for service with ID "4" scheduled weekly from Monday 22:00 till Tuesday 10:00.

Request:
```
{
    "jsonrpc": "2.0",
    "method": "service.update",
    "params": {
        "serviceid": "4",
        "times": [
            {
                "type": "1",
                "ts_from": "165600",
                "ts_to": "201600"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:
```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "4"
        ]
    },
    "id": 1
}
```

Source

CService::update() in ui/include/classes/api/services/CService.php.
**Settings**

This class is designed to work with common administration settings.

Object references:

- **Settings**

Available methods:

- `settings.get` - retrieve settings
- `settings.update` - update settings

> **Settings object**

The following objects are directly related to the `settings` API.

**Settings**

The settings object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default_lang</td>
<td>string</td>
<td>System language by default.</td>
</tr>
<tr>
<td>default_timezone</td>
<td>string</td>
<td>System time zone by default.</td>
</tr>
<tr>
<td>default_theme</td>
<td>string</td>
<td>Default theme.</td>
</tr>
<tr>
<td>search_limit</td>
<td>integer</td>
<td>Limit for search and filter results.</td>
</tr>
<tr>
<td>max_overview_table_size</td>
<td>integer</td>
<td>Max number of columns and rows in Data overview and Trigger overview dashboard widgets.</td>
</tr>
<tr>
<td>max_in_table</td>
<td>integer</td>
<td>Max count of elements to show inside table cell.</td>
</tr>
<tr>
<td>server_check_interval</td>
<td>integer</td>
<td>Show warning if Zabbix server is down.</td>
</tr>
<tr>
<td>work_period</td>
<td>string</td>
<td>Working time.</td>
</tr>
<tr>
<td>show_technical_errors</td>
<td>integer</td>
<td>Show technical errors (PHP/SQL) to non-Super admin users and to users that are not part of user groups with debug mode enabled.</td>
</tr>
</tbody>
</table>

Default: en_GB.

Default: system - system default.

Default: 1000.

Default: 50.

Default: 50.

Default: 1-5, 09:00-18:00.

Possible values:

- 0 - Do not show warning;
- 10 - (default) Show warning.

Possible values:

- 0 - (default) Do not technical errors;
- 1 - Show technical errors.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>history_period</td>
<td>string</td>
<td>Max period to display history data in Latest data, Web, and Data overview dashboard widgets. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 24h.</td>
</tr>
<tr>
<td>period_default</td>
<td>string</td>
<td>Time filter default period. Accepts seconds and time unit with suffix with month and year support (30s,1m,2h,1d,1M,1y).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 1h.</td>
</tr>
<tr>
<td>max_period</td>
<td>string</td>
<td>Max period for time filter. Accepts seconds and time unit with suffix with month and year support (30s,1m,2h,1d,1M,1y).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 2y.</td>
</tr>
<tr>
<td>severity_color_0</td>
<td>string</td>
<td>Color for &quot;Not classified&quot; severity as a hexadecimal color code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 97AAB3.</td>
</tr>
<tr>
<td>severity_color_1</td>
<td>string</td>
<td>Color for &quot;Information&quot; severity as a hexadecimal color code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 7499FF.</td>
</tr>
<tr>
<td>severity_color_2</td>
<td>string</td>
<td>Color for &quot;Warning&quot; severity as a hexadecimal color code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: FFC859.</td>
</tr>
<tr>
<td>severity_color_3</td>
<td>string</td>
<td>Color for &quot;Average&quot; severity as a hexadecimal color code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: FFA059.</td>
</tr>
<tr>
<td>severity_color_4</td>
<td>string</td>
<td>Color for &quot;High&quot; severity as a hexadecimal color code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: E97659.</td>
</tr>
<tr>
<td>severity_color_5</td>
<td>string</td>
<td>Color for &quot;Disaster&quot; severity as a hexadecimal color code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: E45959.</td>
</tr>
<tr>
<td>severity_name_0</td>
<td>string</td>
<td>Name for &quot;Not classified&quot; severity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: Not classified.</td>
</tr>
<tr>
<td>severity_name_1</td>
<td>string</td>
<td>Name for &quot;Information&quot; severity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: Information.</td>
</tr>
<tr>
<td>severity_name_2</td>
<td>string</td>
<td>Name for &quot;Warning&quot; severity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: Warning.</td>
</tr>
<tr>
<td>severity_name_3</td>
<td>string</td>
<td>Name for &quot;Average&quot; severity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: Average.</td>
</tr>
<tr>
<td>severity_name_4</td>
<td>string</td>
<td>Name for &quot;High&quot; severity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: High.</td>
</tr>
<tr>
<td>severity_name_5</td>
<td>string</td>
<td>Name for &quot;Disaster&quot; severity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: Disaster.</td>
</tr>
<tr>
<td>custom_color</td>
<td>integer</td>
<td>Use custom event status colors.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) Do not use custom event status colors; 1 - Use custom event status colors.</td>
</tr>
<tr>
<td>ok_period</td>
<td>string</td>
<td>Display OK triggers period. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 5m.</td>
</tr>
<tr>
<td>blink_period</td>
<td>string</td>
<td>On status change triggers blink period. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 2m.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>problem_unack_color</td>
<td>string</td>
<td>Color for unacknowledged PROBLEM events as a hexadecimal color code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: CC0000.</td>
</tr>
<tr>
<td>problem_ack_color</td>
<td>string</td>
<td>Color for acknowledged PROBLEM events as a hexadecimal color code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: CC0000.</td>
</tr>
<tr>
<td>ok_unack_color</td>
<td>string</td>
<td>Color for unacknowledged RESOLVED events as a hexadecimal color code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 009900.</td>
</tr>
<tr>
<td>ok_ack_color</td>
<td>string</td>
<td>Color for acknowledged RESOLVED events as a hexadecimal color code.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 009900.</td>
</tr>
<tr>
<td>problem_unack_style</td>
<td>integer</td>
<td>Blinking for unacknowledged PROBLEM events.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - Do not show blinking; 1 - (default) Show blinking.</td>
</tr>
<tr>
<td>problem_ack_style</td>
<td>integer</td>
<td>Blinking for acknowledged PROBLEM events.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - Do not show blinking; 1 - (default) Show blinking.</td>
</tr>
<tr>
<td>ok_unack_style</td>
<td>integer</td>
<td>Blinking for unacknowledged RESOLVED events.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - Do not show blinking; 1 - (default) Show blinking.</td>
</tr>
<tr>
<td>ok_ack_style</td>
<td>integer</td>
<td>Blinking for acknowledged RESOLVED events.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - Do not show blinking; 1 - (default) Show blinking.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>Frontend URL.</td>
</tr>
<tr>
<td>discovery_groupid</td>
<td>integer</td>
<td>ID of the host group to which will be automatically placed discovered hosts.</td>
</tr>
<tr>
<td>default_inventory_mode</td>
<td>integer</td>
<td>Default host inventory mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: -1 - (default) Disabled; 0 - Manual; 1 - Automatic.</td>
</tr>
<tr>
<td>alert_usrgrp_id</td>
<td>integer</td>
<td>ID of the user group to which will be sending database down alarm message.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If set to empty, the alarm message will not be sent.</td>
</tr>
<tr>
<td>snmptrap_logging</td>
<td>integer</td>
<td>Log unmatched SNMP traps.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - Do not log unmatched SNMP traps; 1 - (default) Log unmatched SNMP traps.</td>
</tr>
<tr>
<td>login_attempts</td>
<td>integer</td>
<td>Number of failed login attempts after which login form will be blocked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 5.</td>
</tr>
<tr>
<td>login_block</td>
<td>string</td>
<td>Time interval during which login form will be blocked if number of failed login attempts exceeds defined in login_attempts field. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 30s.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>validate_uri_schemes</td>
<td>integer</td>
<td>Validate URI schemes. Possible values: 0 - Do not validate; 1 - (default) Validate.</td>
</tr>
<tr>
<td>uri_valid_schemes</td>
<td>string</td>
<td>Valid URI schemes.</td>
</tr>
<tr>
<td>x_frame_options</td>
<td>string</td>
<td>Default: http,https,ftp,mailto,tel,ssh. X-Frame-Options HTTP header.</td>
</tr>
<tr>
<td>iframe_sandboxing_enabled</td>
<td>integer</td>
<td>Use iframe sandboxing. Possible values: 0 - Do not use; 1 - (default) Use.</td>
</tr>
<tr>
<td>iframe_sandboxing_exceptions</td>
<td>string</td>
<td>Iframe sandboxing exceptions.</td>
</tr>
<tr>
<td>connect_timeout</td>
<td>string</td>
<td>Connection timeout with Zabbix server. Default: 3s.</td>
</tr>
<tr>
<td>socket_timeout</td>
<td>string</td>
<td>Network default timeout. Default: 3s.</td>
</tr>
<tr>
<td>media_type_test_timeout</td>
<td>string</td>
<td>Network timeout for media type test. Default: 65s.</td>
</tr>
<tr>
<td>item_test_timeout</td>
<td>string</td>
<td>Network timeout for item tests. Default: 60s.</td>
</tr>
<tr>
<td>script_timeout</td>
<td>string</td>
<td>Network timeout for script execution. Default: 60s.</td>
</tr>
<tr>
<td>report_test_timeout</td>
<td>string</td>
<td>Network timeout for scheduled report test. Default: 60s.</td>
</tr>
<tr>
<td>auditlog_enabled</td>
<td>integer</td>
<td>Enable audit logging. Possible values: 0 - Disable; 1 - (default) Enable.</td>
</tr>
<tr>
<td>ha_failover_delay</td>
<td>string</td>
<td>Failover delay in seconds. Default: 1m.</td>
</tr>
<tr>
<td>geomaps_tile_provider</td>
<td>string</td>
<td>Geomap tile provider. Possible values: OpenStreetMap.Mapnik - (default) OpenStreetMap Mapnik; OpenTopoMap - OpenTopoMap; Stamen.TonerLite - Stamen Toner Lite; Stamen.Terrain - Stamen Terrain; USGS.USTopo - USGS US Topo; USGS.USImagery - USGS US Imagery. Supports empty string to specify custom values of geomaps_tile_url, geomaps_max_zoom and geomaps_attribution.</td>
</tr>
<tr>
<td>geomaps_tile_url</td>
<td>string</td>
<td>Geomap tile URL if geomaps_tile_provider is set to empty string.</td>
</tr>
<tr>
<td>geomaps_max_zoom</td>
<td>integer</td>
<td>Geomap max zoom level if geomaps_tile_provider is set to empty string. Max zoom must be in the range between 0 and 30.</td>
</tr>
<tr>
<td>geomaps_attribution</td>
<td>string</td>
<td>Geomap attribution text if geomaps_tile_provider is set to empty string.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>vault_provider</td>
<td>integer</td>
<td>Vault provider.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) HashiCorp Vault;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - CyberArk Vault.</td>
</tr>
</tbody>
</table>

settings.get

Description

object settings.get(object parameters)

The method allows to retrieve settings object according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports only one parameter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>query</td>
<td>This parameter being common for all get methods described in the reference commentary.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns settings object.

Examples

Request:

```
{
    "jsonrpc": "2.0",
    "method": "settings.get",
    "params": {
        "output": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "default_theme": "blue-theme",
        "search_limit": "1000",
        "max_in_table": "50",
        "server_check_interval": "10",
        "work_period": "1-5,09:00-18:00",
        "show_technical_errors": "0",
        "history_period": "24h",
        "period_default": "1h",
        "max_period": "2y",
        "severity_color_0": "97AAB3",
        "severity_color_1": "7499FF",
        "severity_color_2": "FFC859",
        "severity_color_3": "FFA059",
        "severity_color_4": "E97659",
    }
}
```
"severity_color_5": "E45959",
"severity_name_0": "Not classified",
"severity_name_1": "Information",
"severity_name_2": "Warning",
"severity_name_3": "Average",
"severity_name_4": "High",
"severity_name_5": "Disaster",
"custom_color": "0",
"ok_period": "5m",
"blink_period": "2m",
"problem_unack_color": "CC0000",
"problem_ack_color": "CC0000",
"ok_unack_color": "009900",
"ok_ack_color": "009900",
"problem_unack_style": "1",
"problem_ack_style": "1",
"ok_unack_style": "1",
"ok_ack_style": "1",
"discovery_groupid": "5",
"default_inventory_mode": "-1",
"alert_usrgrpid": "7",
"snmptrap_logging": "1",
"default_lang": "en_GB",
"default_timezone": "system",
"login_attempts": "5",
"login_block": "30s",
"validate_uri_schemes": "1",
"uri_valid_schemes": "http,https,ftp,file,mailto,tel,ssh",
"x_frame_options": "SAMEORIGIN",
"iframe_sandboxing_enabled": "1",
"iframe_sandboxing_exceptions": "",
"max_overview_table_size": "50",
"connect_timeout": "3s",
"socket_timeout": "3s",
"media_type_test_timeout": "65s",
"script_timeout": "60s",
"item_test_timeout": "60s",
"url": "",
"report_test_timeout": "60s",
"auditlog_enabled": "1",
"ha_failover_delay": "1m",
"geoms_map_tile_provider": "OpenStreetMap.Mapnik",
"geoms_map_tile_url": "",
"geoms_map_max_zoom": "0",
"geoms_map_attribution": "",
"vault_provider": "0"
},
"id": 1
}

Source
CSettings::get() in ui/include/classes/api/services/CSettings.php.

settings.update

Description

object settings.update(object settings)

This method allows to update existing common settings.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.
Parameters

(object) Settings properties to be updated.

Return values

(array) Returns array with the names of updated parameters.

Examples

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "settings.update",
    "params": {
        "login_attempts": "1",
        "login_block": "1m"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        "login_attempts",
        "login_block"
    ],
    "id": 1
}
```

Source

CSettings::update() in ui/include/classes/api/services/CSettings.php.

SLA

This class is designed to work with SLA (Service Level Agreement) objects used to estimate the performance of IT infrastructure and business services.

Object references:

- SLA
- SLA schedule
- SLA excluded downtime
- SLA service tag

Available methods:

- `sla.create` - creating new SLAs
- `sla.delete` - deleting SLAs
- `sla.get` - retrieving SLAs
- `sla.getsli` - retrieving availability information as Service Level Indicator (SLI)
- `sla.update` - updating SLAs

> SLA object

The following objects are directly related to the sla (Service Level Agreement) API.

SLA

The SLA object has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slaid</td>
<td>string</td>
<td>(readonly) ID of the SLA.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the SLA.</td>
</tr>
<tr>
<td>period</td>
<td>integer</td>
<td>Reporting period of the SLA.</td>
</tr>
<tr>
<td>slo</td>
<td>float</td>
<td>Minimum acceptable Service Level Objective expressed as a percent. If the Service Level Indicator (SLI) drops lower, the SLA is considered to be in problem/unfulfilled state.</td>
</tr>
<tr>
<td>effective_date</td>
<td>integer</td>
<td>Effective date of the SLA.</td>
</tr>
<tr>
<td>timezone</td>
<td>string</td>
<td>Reporting time zone, for example: Europe/London, UTC.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Status of the SLA.</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Description of the SLA.</td>
</tr>
</tbody>
</table>

SLA Schedule

The SLA schedule object defines periods where the connected service(s) are scheduled to be in working order. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>period_from</td>
<td>integer</td>
<td>Starting time of the recurrent weekly period of time (inclusive).</td>
</tr>
<tr>
<td>period_to</td>
<td>integer</td>
<td>Ending time of the recurrent weekly period of time (exclusive).</td>
</tr>
</tbody>
</table>

SLA excluded downtime

The excluded downtime object defines periods where the connected service(s) are scheduled to be out of working order, without affecting SLI, e.g. undergoing planned maintenance.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the excluded downtime.</td>
</tr>
<tr>
<td>period_from</td>
<td>integer</td>
<td>Starting time of the excluded downtime (inclusive).</td>
</tr>
<tr>
<td>period_to</td>
<td>integer</td>
<td>Ending time of the excluded downtime (exclusive).</td>
</tr>
</tbody>
</table>

SLA service tag

The SLA service tag object links services to include in the calculations for the SLA. It has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>SLA service tag name.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>operator</td>
<td>integer</td>
<td>SLA service tag operator.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>SLA service tag value.</td>
</tr>
</tbody>
</table>

**sla.create**

Description

**object sla.create(object/array SLAs)**

This method allows to create new SLA objects.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) SLA objects to create.

Additionally to the standard SLA properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_tags</td>
<td>array</td>
<td>SLA service tags to be created for the SLA.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>At least one service tag must be specified.</td>
</tr>
<tr>
<td>schedule</td>
<td>array</td>
<td>SLA schedule to be created for the SLA.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifying an empty parameter will be interpreted as a 24x7 schedule.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 24x7 schedule.</td>
</tr>
<tr>
<td>excluded_downtime</td>
<td>array</td>
<td>SLA excluded downtimes to be created for the SLA.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the created SLAs under the slaids property. The order of the returned IDs matches the order of the passed SLAs.

Examples

Creating an SLA

Instruct to create an SLA entry for: * tracking uptime for SQL-engine related services; * custom schedule of all weekdays excluding last hour on Saturday; * an effective date of the last day of the year 2022; * with 1 hour and 15 minutes long planned downtime starting at midnight on the 4th of July; * SLA weekly report calculation will be on; * the minimum acceptable SLO will be 99.9995%.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "sla.create",
    "params": [
        {
            "name": "Database Uptime",
            "slo": "99.9995",
            "period": "1",
            "timezone": "America/Toronto",
            "description": "Provide excellent uptime for main database engines.",
            "effective_date": 1672444800,
            "status": 1,
            "schedule": [
                {
                    "period_from": 0,
                    "period_to": 601200
                }
            ]
        }
    ]
}
```
{  "jsonrpc": "2.0",  "method": "sla.delete",  "params": [    "4",    "5"  ],  "auth": "3a57200802b24cda67c4e4010b50c065",  "id": 1}

Response:
{
  "jsonrpc": "2.0",  "result": {   "slaids": [     "4",     "5"   ] },  "id": 1
}

Source
CSla::delete() in ui/include/classes/api/services/CSla.php.

**sla.get**

**Description**
integer/array sla.get(object parameters)

The method allows to retrieve SLA objects according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**
(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slaids</td>
<td>string/array</td>
<td>Return only SLAs with the given IDs.</td>
</tr>
<tr>
<td>serviceids</td>
<td>string/array</td>
<td>Return only SLAs matching the specific services.</td>
</tr>
<tr>
<td>selectSchedulequery</td>
<td></td>
<td>Return a schedule property with SLA schedules.</td>
</tr>
<tr>
<td>selectExcludedDowntimes</td>
<td></td>
<td>Return an excluded_downtimes property with SLA excluded downtimes.</td>
</tr>
<tr>
<td>selectServiceTags</td>
<td></td>
<td>Return a service_tags property with SLA service tags.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td>the reference commentary.</td>
</tr>
</tbody>
</table>

1217
Parameter | Type | Description
--- | --- | ---
xcludeSearch | boolean | 
filter | object | 
limit | integer | 
output | query | 
preservekeys | boolean | 
search | object | 
searchByAny | boolean | 
searchWildcardsEnabled | 
sortorder | string/array | 
startSearch | boolean | 

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving all SLAs

Retrieve all data about all SLAs and their properties.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "sla.get",
   "params": {
      "output": "extend",
      "selectSchedule": ["period_from", "period_to"],
      "selectExcludedDowntimes": ["name", "period_from", "period_to"],
      "selectServiceTags": ["tag", "operator", "value"],
      "preservekeys": true
   },
   "auth": "85dd04b94cbfad794616eb923be13c71",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "i": {
         "slaid": "i",
         "name": "Database Uptime",
         "period": "1",
         "slo": "99.9995",
         "effective_date": "1672444800",
         "timezone": "America/Toronto",
         "status": "1",
         "description": "Provide excellent uptime for main SQL database engines.",
         "service_tags": [
            {
               "tag": "Database",
               "operator": "0",
               "value": "MySQL"
            },
            {
               "tag": "Database",
               "operator": "0",
               "value": "PostgreSQL"
            }
         ]
      }
   }
}
```


Source

CSla:get() in ui/include/classes/api/services/CSla.php.

sla.getsli

Description

object sla.getsli(object parameters)

This method allows to calculate the Service Level Indicator (SLI) data.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters containing the SLA ID, reporting periods and, optionally, the IDs of the services - to calculate the SLI for.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slaid</td>
<td>string</td>
<td>IDs of services to return availability information for.</td>
</tr>
<tr>
<td>period_from</td>
<td>integer</td>
<td>Starting date (inclusive) to report the SLI for.</td>
</tr>
<tr>
<td>period_to</td>
<td>integer</td>
<td>Ending date (exclusive) to report the SLI for.</td>
</tr>
<tr>
<td>periods</td>
<td>array</td>
<td>Preferred number of periods to report.</td>
</tr>
<tr>
<td>serviceIds</td>
<td>string/array</td>
<td>Possible values: <code>1-100</code></td>
</tr>
</tbody>
</table>

Partitioning of periods

The following demonstrates the arrangement of returned period slices based on combinations of parameters.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>period_from</td>
<td>The last 20 periods (including the current one) but not past the first available period based on the effective date of the SLA.</td>
</tr>
<tr>
<td>- specified</td>
<td>The last periods specified by the <code>periods</code> parameter.</td>
</tr>
<tr>
<td>- specified</td>
<td>The last 20 periods before the <code>specified date</code>, but not past the first available period based on the effective date of the SLA.</td>
</tr>
<tr>
<td>- specified</td>
<td>The last periods specified by the <code>periods</code> parameter before the <code>specified date</code>.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>specified</td>
<td>The first 20 periods (including the current one) but not past the current one.</td>
</tr>
<tr>
<td>specified specified</td>
<td>The first periods specified by the <code>periods</code> parameter starting with the specified date.</td>
</tr>
<tr>
<td>specified specified specified</td>
<td>Periods within the specified date range, but no more than 100 and not past the first available period based on the effective date of the SLA.</td>
</tr>
<tr>
<td>specified specified specified specified</td>
<td>Periods within the specified date range, but no more than the specified number of periods and not past the first available period based on the effective date of the SLA.</td>
</tr>
</tbody>
</table>

Return values

*(object)* Returns the results of the calculation.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| `periods` array | List of the reported periods. Each reported period is represented as an object consisting of:  
- `period_from` - Starting date of the reported period (timestamp).  
- `period_to` - Ending date of the reported period (timestamp).  
  Periods are sorted by `period_from` field ascending. |
| `serviceIds` array | List of service IDs in the reported periods.  
The sorting order of the list is not defined. Even if `serviceIds` parameter was passed to the sla.getsla method. |
| `sli` array    | SLI data (as a two-dimensional array) for each reported period and service.  
The index of the `periods` property is used as the first dimension of the `sli` property.  
The index of the `serviceIds` property is used as the second dimension of the `sli` property. |

SLI data

The SLI data returned for each reported period and service consists of:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uptime</code> integer</td>
<td>Amount of time service spent in an OK state during scheduled uptime, less the excluded downtimes.</td>
</tr>
<tr>
<td><code>downtime</code> integer</td>
<td>Amount of time service spent in a not OK state during scheduled uptime, less the excluded downtimes.</td>
</tr>
<tr>
<td><code>sli</code> float</td>
<td>SLI (per cent of total uptime), based on uptime and downtime.</td>
</tr>
<tr>
<td><code>error_budget</code> integer</td>
<td>Error budget (in seconds), based on the SLI and the SLO.</td>
</tr>
</tbody>
</table>
| `excluded_downtimes` array | Array of excluded downtimes in this reporting period.  
  Each object will contain the following parameters:  
  - `name` - Name of the excluded downtime.  
  - `period_from` - Starting date and time (inclusive) of the excluded downtime.  
  - `period_to` - Ending date and time (exclusive) of the excluded downtime.  
  Excluded downtimes are sorted by `period_from` field ascending. |

Examples

Calculating SLI

Retrieve SLI on services with IDs “50, 60 and 70” linked to an SLA with ID of “5” for 3 periods starting from Nov 01, 2021.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "service.getsla",
  "params": {
    "slaid": "5",
```
"serviceids": [50, 60, 70],
"periods": 3,
"period_from": "1635724800"
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{

"jsonrpc": "2.0",
"result": {

"periods": [

{
"period_from": 1635724800,
"period_to": 1638316800
},
{
"period_from": 1638316800,
"period_to": 1640995200
},
{
"period_from": 1640995200,
"period_to": 1643673600
}
],
"serviceids": [50, 60, 70],
"sli": [

{
"uptime": 1186212,
"downtime": 0,
"sli": 100,
"error_budget": 0,
"excluded_downtimes": [

{
"name": "Excluded Downtime - 1",
"period_from": 1637836212,
"period_to": 1638316800
}
]
},
{
"uptime": 1186212,
"downtime": 0,
"sli": 100,
"error_budget": 0,
"excluded_downtimes": [

{
"name": "Excluded Downtime - 1",
"period_from": 1637836212,
"period_to": 1638316800
}
]
}
}
{ "uptime": 1186212, "downtime": 0, "sli": 100, "error_budget": 0,
"excluded_downtimes": [
  { "name": "Excluded Downtime - 1",
    "period_from": 1637836212,
    "period_to": 1638316800
  }
]
},

{ "uptime": 1147548, "downtime": 0, "sli": 100, "error_budget": 0,
"excluded_downtimes": [
  { "name": "Excluded Downtime - 1",
    "period_from": 1638439200,
    "period_to": 1639109652
  }
]
},

{ "uptime": 1147548, "downtime": 0, "sli": 100, "error_budget": 0,
"excluded_downtimes": [
  { "name": "Excluded Downtime - 1",
    "period_from": 1638439200,
    "period_to": 1639109652
  }
]
},

{ "uptime": 1674000, "downtime": 0, "sli": 100, "error_budget": 0,
"excluded_downtimes": [
] }
null

Source

cService::getSla() in ui/include/classes/api/services/CSpecification.php

**sla.update**

Description

object sla.update(object/array slaids)

This method allows to update existing SLA entries.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) SLA properties to be updated.

The slaid property must be defined for each SLA, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard SLA properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_tags</td>
<td>array</td>
<td>SLA service tags to replace the current SLA service tags.</td>
</tr>
<tr>
<td>schedule</td>
<td>array</td>
<td>SLA schedule to replace the current one.</td>
</tr>
<tr>
<td>excluded_downtimes</td>
<td></td>
<td>Specifying parameter as empty will be interpreted as a 24x7 schedule.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated SLAs under the slaiids property.

Examples

Updating service tags

Make SLA with ID "5" to be calculated at monthly intervals for NoSQL related services, without changing its schedule or excluded downtimes; set SLO to 95%.

Request:
Changing the schedule of an SLA

Switch the SLA with ID "5" to a 24x7 schedule.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "service.update",
  "params": {
    "slaid": "5",
    "schedule": []
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": {
    "slaids": [
      "5"
    ]
  },
  "id": 1
}
```
Changing the excluded downtimes for an SLA

Add a planned 4 hour long RAM upgrade downtime on the 6th of April, 2022, while keeping (needs to be defined anew) a previously existing software upgrade planned on the 4th of July for the SLA with ID "5".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "service.update",
    "params": {
        "slaid": "5",
        "excluded_downtimes": [
            {
                "name": "Software version upgrade rollout",
                "period_from": "1648760400",
                "period_to": "1648764900"
            },
            {
                "name": "RAM upgrade",
                "period_from": "1649192400",
                "period_to": "1649206800"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "slaids": [
            "5"
        ],
    },
    "id": 1
}
```

Source

CSla::update() in ui/include/classes/api/services/CSla.php.

Task

This class is designed to work with tasks (such as checking items or low-level discovery rules without config reload).

Object references:

- Task
- 'Execute now' request object
- 'Diagnostic information' request object
- Statistic request object
- Statistic result object

Available methods:

- task.create - creating new tasks
- task.get - retrieving tasks
The following objects are directly related to the task API.

The task object has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskid</td>
<td>string</td>
<td>(readonly) ID of the task.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>Type of the task.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Diagnostic information;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Refresh proxy configuration;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 - Execute now.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>(readonly) Status of the task.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - new task;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - task in progress;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - task is completed;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - task is expired.</td>
</tr>
<tr>
<td>clock</td>
<td>timestamp</td>
<td>(readonly) Time when the task was created.</td>
</tr>
<tr>
<td>ttl</td>
<td>integer</td>
<td>(readonly) The time in seconds after which task expires.</td>
</tr>
<tr>
<td>proxy_hostid</td>
<td>string</td>
<td>ID of the proxy about which diagnostic information statistic is collected. Ignored for 'Execute now' tasks.</td>
</tr>
<tr>
<td>request</td>
<td>object</td>
<td>Task request object according to the task type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Object of 'Execute now' task is described in detail below;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Object of 'Refresh proxy configuration' task is described in detail below;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Object of 'Diagnostic information' task is described in detail below.</td>
</tr>
<tr>
<td>result</td>
<td>object</td>
<td>(readonly) Result object of the diagnostic information task. May contain NULL if result is not yet ready. Result object is described in detail below.</td>
</tr>
</tbody>
</table>

'Execute now' request object

The 'Execute now' task request object has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemid</td>
<td>string</td>
<td>ID of item and low-level discovery rules.</td>
</tr>
</tbody>
</table>

'Refresh proxy configuration' request object

The 'Refresh proxy configuration' task request object has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proxy_hostid</td>
<td>array</td>
<td>Proxy IDs.</td>
</tr>
</tbody>
</table>

'Diagnostic information' request object

The diagnostic information task request object has the following properties. Statistic request object for all types of properties is described in detail below.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>historycache</td>
<td>object</td>
<td>History cache statistic request. Available on server and proxy.</td>
</tr>
<tr>
<td>valuecache</td>
<td>object</td>
<td>Items cache statistic request. Available on server.</td>
</tr>
<tr>
<td>preprocessing</td>
<td>object</td>
<td>Preprocessing manager statistic request. Available on server and proxy.</td>
</tr>
<tr>
<td>alerting</td>
<td>object</td>
<td>Alert manager statistic request. Available on server.</td>
</tr>
<tr>
<td>lld</td>
<td>object</td>
<td>LLD manager statistic request. Available on server.</td>
</tr>
</tbody>
</table>

Statistic request object
Statistic request object is used to define what type of information should be collected about server/proxy internal processes. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stats</td>
<td>query</td>
<td>Statistic object properties to be returned. The list of available fields for each type of diagnostic information statistic are described in detail below. Default: extend will return all available statistic fields.</td>
</tr>
</tbody>
</table>
| top      | object | Object to sort and limit returned statistic values. The list of available fields for each type of diagnostic information statistic are described in detail below. Example: 

```json
  { "source.alerts": 10 }
```

List of statistic fields available for each type of diagnostic information request

Following statistic fields can be requested for each type of diagnostic information request property.

<table>
<thead>
<tr>
<th>Diagnostic type</th>
<th>Available fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>historycache</td>
<td>items</td>
<td>Number of cached items.</td>
</tr>
<tr>
<td></td>
<td>values</td>
<td>Number of cached values.</td>
</tr>
<tr>
<td></td>
<td>memory</td>
<td>Shared memory statistics (free space, number of used chunks, number of free chunks, max size of free chunk).</td>
</tr>
<tr>
<td></td>
<td>memory.data</td>
<td>History data cache shared memory statistics.</td>
</tr>
<tr>
<td></td>
<td>memory.index</td>
<td>History index cache shared memory statistics.</td>
</tr>
<tr>
<td>valuecache</td>
<td>items</td>
<td>Number of cached items.</td>
</tr>
<tr>
<td></td>
<td>values</td>
<td>Number of cached values.</td>
</tr>
<tr>
<td></td>
<td>mode</td>
<td>Value cache mode.</td>
</tr>
<tr>
<td></td>
<td>memory</td>
<td>Shared memory statistics (free space, number of used chunks, number of free chunks, max size of free chunk).</td>
</tr>
<tr>
<td></td>
<td>memory.data</td>
<td>History data cache shared memory statistics.</td>
</tr>
<tr>
<td></td>
<td>memory.index</td>
<td>History index cache shared memory statistics.</td>
</tr>
<tr>
<td>preprocessing</td>
<td>values</td>
<td>Number of queued values.</td>
</tr>
<tr>
<td></td>
<td>preproc.values</td>
<td>Number of queued values with preprocessing steps.</td>
</tr>
<tr>
<td>alerting</td>
<td>alerts</td>
<td>Number of queued alerts.</td>
</tr>
<tr>
<td>lld</td>
<td>rules</td>
<td>Number of queued rules.</td>
</tr>
<tr>
<td></td>
<td>values</td>
<td>Number of queued values.</td>
</tr>
</tbody>
</table>

List of sorting fields available for each type of diagnostic information request

Following statistic fields can be used to sort and limit requested information.

<table>
<thead>
<tr>
<th>Diagnostic type</th>
<th>Available fields</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>historycache</td>
<td>values</td>
<td>integer</td>
</tr>
<tr>
<td>valuecache</td>
<td>values</td>
<td>integer</td>
</tr>
<tr>
<td>valuecache</td>
<td>request.values</td>
<td>integer</td>
</tr>
<tr>
<td>preprocessing</td>
<td>media.values</td>
<td>integer</td>
</tr>
<tr>
<td>preprocessing</td>
<td>source.alerts</td>
<td>integer</td>
</tr>
<tr>
<td>alerting</td>
<td>source.alerts</td>
<td>integer</td>
</tr>
<tr>
<td>lld</td>
<td>values</td>
<td>integer</td>
</tr>
</tbody>
</table>

Statistic result object

Statistic result object is retrieved in result field of task object.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>integer</td>
<td>(readonly) Status of the task result. Possible values: -1 - error occurred during performing task; 0 - task result is created.</td>
</tr>
<tr>
<td>data</td>
<td>string/object</td>
<td>Results according to the statistic request object of particular diagnostic information task. Contains error message string if error occurred during performing task.</td>
</tr>
</tbody>
</table>

**task.create**

**Description**

T object `task.create(object/array tasks)`

This method allows to create a new task (such as collect diagnostic data or check items or low-level discovery rules without config reload).

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) A task to create.

The method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>type</strong></td>
<td>integer</td>
<td>Task type. Possible values: 1 - Diagnostic information; 2 - Refresh proxy configuration; 6 - Execute now.</td>
</tr>
<tr>
<td><strong>request</strong></td>
<td>object</td>
<td>Task request object according to the task type. Correct format of request object is described in Task object section.</td>
</tr>
<tr>
<td>proxy_hostid</td>
<td>integer</td>
<td>Proxy about which Diagnostic information task will collect data. Ignored for ‘Execute now’ tasks.</td>
</tr>
</tbody>
</table>

Note that ‘Execute now’ tasks can be created only for the following types of items/discovery rules:

- Zabbix agent
- SNMPv1/v2/v3 agent
- Simple check
- Internal check
- External check
- Database monitor
- HTTP agent
- IPMI agent
- SSH agent
- TELNET agent
- Calculated check
- JMX agent
- Dependent item

If item or discovery rule is of type Dependent item, then top level master item must be of type: Zabbix agent - SNMPv1/v2/v3 agent - Simple check - Internal check - External check - Database monitor - HTTP agent - IPMI agent - SSH agent - TELNET agent - Calculated check - JMX agent

**Return values**

(object) Returns an object containing the IDs of the created tasks under the taskids property. One task is created for each item and low-level discovery rule. The order of the returned IDs matches the order of the passed itemids.
Examples

Creating a task

Create a task Execute now for two items. One is an item, the other is a low-level discovery rule.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "task.create",
    "params": [
        {
            "type": 6,
            "request": {
                "itemid": "10092"
            }
        },
        {
            "type": 6,
            "request": {
                "itemid": "10093"
            }
        }
    ],
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "taskids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

Create a task Refresh proxy configuration for two proxies.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "task.create",
    "params": [
        {
            "type": 2,
            "request": {
                "proxy_hostids": ["10459", "10460"]
            }
        }
    ],
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "taskids": [
            "1"
        ]
    }
}
```
Create a task diagnostic information.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "task.create",
    "params": [
        {
            "type": 1,
            "request": {
                "alerting": {
                    "stats": [
                    "alerts"
                ],
                "top": {
                    "media.alerts": 10
                }
            },
            "lld": {
                "stats": "extend",
                "top": {
                    "values": 5
                }
            }
        },
        "proxy_hostid": 0
    ],
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 2
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "taskids": [
            "3"
        ]
    },
    "id": 2
}
```

See also

- Task
- 'Execute now' request object
- 'Diagnostic information' request object
- Statistic request object

Source

CTask::create() in ui/include/classes/api/services/CTask.php.

**task.get**

Description

integer/array task.get(object parameters)
The method allows to retrieve tasks according to the given parameters. Method returns details only about 'diagnostic information' tasks.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>taskids</td>
<td>string/array</td>
<td>Return only tasks with the given IDs.</td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns an array of objects.

Examples

Retrieve task by ID

Retrieve all the data about the task with the ID "1".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "task.get",
    "params": {
        "output": "extend",
        "taskids": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "taskid": "1",
            "type": "7",
            "status": "3",
            "clock": "1601039076",
            "ttl": "3600",
            "proxy_hostid": null,
            "request": {
                "alerting": {
                    "stats": [
                        "alerts"
                    ],
                    "top": {
                        "media.alerts": 10
                    }
                },
                "lld": {
                    "stats": "extend",
                    "top": {
                        "values": 5
                    }
                }
            }
        }
    ]
}
```


```json
{}
,
"result": {
  "data": {
    "alerting": {
      "alerts": 0,
      "top": {
        "media.alerts": [],
        "time": 0.000663
      },
      "time": 0.000663
    },
    "lld": {
      "rules": 0,
      "values": 0,
      "top": {
        "values": [],
        "time": 0.000442
      },
      "time": 0.000442
    },
    "status": "0"
  }
},
"id": 1
}
```

See also

- Task
- Statistic result object

Source

CTask::get() in ui/include/classes/api/services/CTask.php.

**Template**

This class is designed to work with templates.

Object references:

- Template

Available methods:

- template.create - creating new templates
- template.delete - deleting templates
- template.get - retrieving templates
- template.massadd - adding related objects to templates
- template.massremove - removing related objects from templates
- template.massupdate - replacing or removing related objects from templates
- template.update - updating templates

> Template object

The following objects are directly related to the template API.

**Template**

The template object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>templateid</td>
<td>string</td>
<td>(readonly) ID of the template.</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>Technical name of the template.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>description</td>
<td>text</td>
<td>Description of the template.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Visible name of the template.</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Universal unique identifier, used for linking imported templates to already existing ones. Auto-generated, if not given. For update operations this field is read only.</td>
</tr>
</tbody>
</table>

**Template tag**

The template tag object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Template tag name.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Template tag value.</td>
</tr>
</tbody>
</table>

**template.create**

**Description**

object template.create(object/array templates)

This method allows to create new templates.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) Templates to create.

Additionally to the standard template properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Template groups to add the template to.</td>
</tr>
<tr>
<td>tags</td>
<td>object/array</td>
<td>The template groups must have the groupid property defined. Template tags.</td>
</tr>
<tr>
<td>templates</td>
<td>object/array</td>
<td>Templates to be linked to the template.</td>
</tr>
<tr>
<td>macros</td>
<td>object/array</td>
<td>The templates must have the templateid property defined. User macros to be created for the template.</td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns an object containing the IDs of the created templates under the templateids property. The order of the returned IDs matches the order of the passed templates.

**Examples**

**Creating a template**

Create a template with tags and link two templates to this template.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "template.create",
    "params": {
        "host": "Linux template",
        "groups": {
            "1233" : 
```
"groupid": 1,
},
"templates": [
{
"templateid": "11115"
},
{
"templateid": "11116"
}
],
"tags": [
{
"tag": "Host name",
"value": "\{HOST.NAME\}"
}
],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
"jsonrpc": "2.0",
"result": {
"templateids": [
"11117"
],
"id": 1
}
}

Source
CTemplate::create() in ui/include/classes/api/services/CTemplate.php.

**template.delete**

**Description**

object template.delete(array templateIds)

This method allows to delete templates.

Deleting a template will cause deletion of all template entities (items, triggers, graphs, etc.). To leave template entities with the hosts, but delete the template itself, first unlink the template from required hosts using one of these methods: template.update, template.massupdate, host.update, host.massupdate.

This method is only available to Admin and Superadmin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(array) IDs of the templates to delete.

**Return values**

(object) Returns an object containing the IDs of the deleted templates under the templateIds property.

**Examples**

Deleting multiple templates

Delete two templates.

Request:
{
  "jsonrpc": "2.0",
}
"method": "template.delete",
"params": [
  "13",
  "32"
],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": {
    "templateids": [
      "13",
      "32"
    ],
  },
  "id": 1
}

Source
CTemplate::delete() in ui/include/classes/api/services/CTemplate.php.

template.get

Description
integer/array template.get(object parameters)
The method allows to retrieve templates according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>templateids</td>
<td>string/array</td>
<td>Return only templates with the given template IDs.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only templates that belong to the given template groups.</td>
</tr>
<tr>
<td>parentTemplateids</td>
<td>string/array</td>
<td>Return only templates that are children of the given templates.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only templates that are linked to the given hosts/templates.</td>
</tr>
<tr>
<td>graphids</td>
<td>string/array</td>
<td>Return only templates that contain the given graphs.</td>
</tr>
<tr>
<td>itemids</td>
<td>string/array</td>
<td>Return only templates that contain the given items.</td>
</tr>
<tr>
<td>triggerids</td>
<td>string/array</td>
<td>Return only templates that contain the given triggers.</td>
</tr>
<tr>
<td>with_items</td>
<td>flag</td>
<td>Return only templates that have items.</td>
</tr>
<tr>
<td>with_triggers</td>
<td>flag</td>
<td>Return only templates that have triggers.</td>
</tr>
<tr>
<td>with_graphs</td>
<td>flag</td>
<td>Return only templates that have graphs.</td>
</tr>
<tr>
<td>with_http tests</td>
<td>flag</td>
<td>Return only templates that have web scenarios.</td>
</tr>
<tr>
<td>evaltype</td>
<td>integer</td>
<td>Rules for tag searching.</td>
</tr>
</tbody>
</table>

Possible values:
0 - (default) And/Or;
2 - Or.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| tags               | array/object  | Return only templates with given tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value.  
|                    |               | Format: [{"tag": "<tag>", "value": "<value>", "operator": "<operator>", ...}. An empty array returns all templates.                        |
| selectTags         | query         | Return template tags in the tags property.                                                                                                                                                           |
| selectHosts        | query         | Return the hosts that are linked to the template in the hosts property.                                                                                                                                |
| selectTemplateGroups | query       | Return the template groups that the template belongs to in the template groups property.                                                                                                               |
| selectTemplates    | query         | Return the child templates in the templates property.                                                                                                                                                 |
| selectParentTemplates | query       | Return the parent templates in the parentTemplates property.                                                                                                                                          |
| selectHttpTests    | query         | Return the web scenarios from the template in the httpTests property.                                                                                                                                   |
| selectItems        | query         | Return items from the template in the items property.                                                                                                                                                 |
| selectDiscoveries  | query         | Return low-level discoveries from the template in the discoveries property.                                                                                                                             |
| selectTriggers     | query         | Return triggers from the template in the triggers property.                                                                                                                                              |
| selectGraphs       | query         | Return graphs from the template in the graphs property.                                                                                                                                                 |
| selectMacros       | query         | Return the macros from the template in the macros property.                                                                                                                                              |
| selectDashboards   | query         | Return dashboards from the template in the dashboards property.                                                                                                                                          |
| selectValueMaps    | query         | Return a valuemaps property with template value maps.                                                                                                                                                   |
| limitSelects       | integer       | Limits the number of records returned by subselects.                                                                                                                                                    |
| sortfield          | string/array  | Sort the result by the given properties.                                                                                                                                                               |

Possible operator values:
- 0 - (default) Contains;
- 1 - Equals;
- 2 - Not like;
- 3 - Not equal
- 4 - Exists;
- 5 - Not exists.

Possible values are: hostid, host, name, status.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>selectGroups</td>
<td>query</td>
<td>This parameter is deprecated, please use selectTemplateGroups instead.</td>
</tr>
<tr>
<td>(deprecated)</td>
<td></td>
<td>Return the template groups that the template belongs to in the groups property.</td>
</tr>
</tbody>
</table>

**Return values**

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

**Examples**

**Retrieving templates by name**

Retrieve all data about two templates named "Linux" and "Windows".

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": "extend",
        "filter": {
            "host": [
                "Linux",
                "Windows"
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "proxy_hostid": "0",
            "host": "Linux",
            "status": "3",
            "disable_until": "0",
            "error": ""
        }
    ]
}
```
"ipmi_username": "", "ipmi_password": "", "ipmi_disable_until": "0", "ipmi_available": "0", "snmp_disable_until": "0", "snmp_available": "0", "maintenanceid": "0", "maintenance_status": "0", "maintenance_type": "0", "maintenance_from": "0", "ipmi_errors_from": "0", "snmp_errors_from": "0", "ipmi_error": "", "snmp_error": "", "jmx_disable_until": "0", "jmx_available": "0", "jmx_errors_from": "0", "jmx_error": "", "name": "Linux", "flags": "0", "templateid": "10001", "description": "", "tls_connect": "1", "tls_accept": "1", "tls_issuer": "", "tls_subject": "", "tls_psk_identity": "", "tls_psk": "", "uuid": "282ffe33afc74cccaf1524d9aa9dc502" },
{
  "proxy_hostid": "0", "host": "Windows", "status": "3", "disable_until": "0", "error": "", "available": "0", "errors_from": "0", "lastaccess": "0", "ipmi_authtype": "0", "ipmi_privilege": "2", "ipmi_username": "", "ipmi_password": "", "ipmi_disable_until": "0", "ipmi_available": "0", "snmp_disable_until": "0", "snmp_available": "0", "maintenanceid": "0", "maintenance_status": "0", "maintenance_type": "0", "maintenance_from": "0", "ipmi_errors_from": "0", "snmp_errors_from": "0", "ipmi_error": "", "snmp_error": "", "jmx_disable_until": "0", "jmx_available": "0", "jmx_errors_from": "0", "jmx_error": "", "name": "Windows", "flags": "0", "templateid": "10081", "uuid": "1238"}
Searching by templatetags

Retrieve templates that have tag "Host name" equal to "{HOST.NAME}".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": ["hostid"],
        "selectTags": "extend",
        "evaltype": 0,
        "tags": [
            {
                "tag": "Host name",
                "value": "{HOST.NAME}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10402",
            "tags": [
                {
                    "tag": "Host name",
                    "value": "{HOST.NAME}"
                }
            ]
        }
    ],
    "id": 1
}
```

See also
- Template group
- Template
- User macro
- Host interface

Source

CTemplate::get() in ui/include/classes/api/services/CTemplate.php.
**template.massadd**

Description

```object template.massadd(object parameters)```

This method allows to simultaneously add multiple related objects to the given templates.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

**Parameters**

( object ) Parameters containing the IDs of the templates to update and the objects to add to the templates.

The method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>templates</td>
<td>object/array</td>
<td>Templates to be updated. The templates must have the <code>templateid</code> property defined.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Template groups to add the given templates to. The template groups must have the <code>groupid</code> property defined.</td>
</tr>
<tr>
<td>macros</td>
<td>object/array</td>
<td>User macros to be created for the given templates. Templates to link to the given templates.</td>
</tr>
<tr>
<td>templates_link</td>
<td>object/array</td>
<td></td>
</tr>
</tbody>
</table>

**Return values**

( object ) Returns an object containing the IDs of the updated templates under the `templateids` property.

**Examples**

**Link a group to a templates**

**Add template group "2" to a two templates.**

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "template.massadd",
    "params": {
        "templates": [
            {
                "templateid": "10085"
            },
            {
                "templateid": "10086"
            }
        ],
        "groups": [
            {
                "groupid": "2"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": {
```
Link two templates to a template

Link templates “10106” and “10104” to template.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "template.massadd",
    "params": {
        "templates": [
            {
                "templateid": "10073"
            }
        ],
        "templates_link": [
            {
                "templateid": "10106"
            },
            {
                "templateid": "10104"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10073"
        ]
    },
    "id": 1
}
```

See also

- template.update
- Host
- Template group
- User macro

Source

CTemplate::massAdd() in ui/include/classes/api/services/CTemplate.php.

**template.massremove**

Description

object template.massremove(object parameters)

This method allows to remove related objects from multiple templates.
This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

Parameters containing the IDs of the templates to update and the objects that should be removed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>templateids</td>
<td>string/array</td>
<td>IDs of the templates to be updated.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Template groups to remove the given templates from.</td>
</tr>
<tr>
<td>macros</td>
<td>string/array</td>
<td>User macros to delete from the given templates.</td>
</tr>
<tr>
<td>templateids_clear</td>
<td>string/array</td>
<td>Templates to unlink and clear from the given templates (upstream).</td>
</tr>
<tr>
<td>templateids_link</td>
<td>string/array</td>
<td>Templates to unlink from the given templates (upstream).</td>
</tr>
</tbody>
</table>

Return values

Returns an object containing the IDs of the updated templates under the templateids property.

Examples

Removing templates from a group

Remove two templates from group "2".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.massremove",
    "params": {
        "templateids": [
            "10085",
            "10086"
        ],
        "groupids": "2"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
    "id": 1
}
```

Unlinking templates from a host

Unlink templates "10106", "10104" from template "10085".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.massremove",
    "params": {
        "templateids": "10085",
        "templateids_link": [
            "10106",
            "10104"
        ]
    }
}
```
Response:
{
   "jsonrpc": "2.0",
   "result": {
      "templateids": [
         "10085"
      ],
   },
   "id": 1
}

See also
- template.update
- User macro

Source
CTemplate::massRemove() in ui/include/classes/api/services/CTemplate.php.

**template.massupdate**

Description

object template.massupdate(object parameters)

This method allows to simultaneously replace or remove related objects and update properties on multiple templates.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

Parameters

(object) Parameters containing the IDs of the templates to update and the objects to replace for the templates.

The method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>templates</td>
<td>object/array</td>
<td>Templates to be updated. Templates to be updated. The templates must have the templateid property defined.</td>
</tr>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Template groups to replace the current template groups the templates belong to.</td>
</tr>
<tr>
<td>macros</td>
<td>object/array</td>
<td>User macros to replace the current user macros on the given templates.</td>
</tr>
<tr>
<td>templates_clear</td>
<td>object/array</td>
<td>Templates to unlink and clear from the given templates.</td>
</tr>
<tr>
<td>templates_link</td>
<td>object/array</td>
<td>Templates to replace the currently linked templates. The templates must have the templateid property defined.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated templates under the templateids property.

Examples

Replacing template groups
Unlink and clear template "10091" from the given templates.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "template.massupdate",
    "params": {
        "templates": [
            {
                "templateid": "10085"
            },
            {
                "templateid": "10086"
            }
        ],
        "templates_clear": [
            {
                "templateid": "10091"
            }
        ],
        "auth": "038e1d7b1735c6a5436ee9eae095879e",
        "id": 1
    }
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
    "id": 1
}
```

See also

- template.update
- template.massadd
- Template group
- User macro

Source

CTemplate::massUpdate() in ui/include/classes/api/services/CTemplate.php.

**template.update**

Description

object template.update(object/array templates)

This method allows to update existing templates.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Template properties to be updated.

The templateid property must be defined for each template, all other properties are optional. Only the given properties will be updated, all others will remain unchanged.

Additionally to the standard template properties, the method accepts the following parameters.
### Parameter Type Description

| groups | object/array | Template groups to replace the current template groups the templates belong to. |
| tags | object/array | The template groups must have the groupid property defined. |
| macros | object/array | Template tags to replace the current template tags. |
| templates | object/array | User macros to replace the current user macros on the given templates. |
| templates_clear | object/array | Templates to replace the currently linked templates. Templates that are not passed are only unlinked. |

The templates must have the templateid property defined.

| templates_clear | object/array | Templates to unlink and clear from the given templates. |

The templates must have the templateid property defined.

---

**Return values**

*(object)* Returns an object containing the IDs of the updated templates under the templateids property.

**Examples**

### Renaming a template

Rename the template to "Template OS Linux".

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": {
        "templateid": "10086",
        "name": "Template OS Linux"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": ["10086"
    ],
    "id": 1
}
```

### Updating template tags

Replace all template tags with a new one.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": {
        "templateid": "10086",
        "tags": [
            {
                "tag": "Host name",
                "value": "{HOST.NAME}"
            }
        ],
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```
Response:

```json
{
  "jsonrpc": "2.0",
  "result": {
    "hostids": [
      "10086"
    ],
  },
  "id": 1
}
```

Source

CTemplate::update() in ui/include/classes/api/services/CTemplate.php.

**Template dashboard**

This class is designed to work with template dashboards.

Object references:

- Template dashboard
- Template dashboard page
- Template dashboard widget
- Template dashboard widget field

Available methods:

- templatedashboard.create - creating new template dashboards
- templatedashboard.delete - deleting template dashboards
- templatedashboard.get - retrieving template dashboards
- templatedashboard.update - updating template dashboards

**> Template dashboard object**

The following objects are directly related to the templatedashboard API.

**Template dashboard**

The template dashboard object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dashboardid</td>
<td>string</td>
<td>(readonly) ID of the template dashboard.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>(required) Name of the template dashboard.</td>
</tr>
<tr>
<td>templateid</td>
<td>string</td>
<td>(required) ID of the template the dashboard belongs to.</td>
</tr>
<tr>
<td>display_period</td>
<td>integer</td>
<td>Default page display period (in seconds).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 10, 30, 60, 120, 600, 1800, 3600.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default: 30.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>auto_start</td>
<td>integer</td>
<td>Auto start slideshow.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - do not auto start slideshow;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) auto start slideshow.</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Universal unique identifier, used for linking imported template dashboards to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>already existing ones. Auto-generated, if not given. For update operations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>this field is readonly.</td>
</tr>
</tbody>
</table>

Template dashboard page

The template dashboard page object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dashboard_pageid</td>
<td>string</td>
<td>(readonly) ID of the dashboard page.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Dashboard page name.</td>
</tr>
<tr>
<td>display_period</td>
<td>integer</td>
<td>Default: empty string. Dashboard page display period (in seconds).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0, 10, 30, 60, 120, 600, 1800, 3600.</td>
</tr>
<tr>
<td>widgets</td>
<td>array</td>
<td>Array of the template dashboard widget objects.</td>
</tr>
</tbody>
</table>

Template dashboard widget

The template dashboard widget object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>widgetid</td>
<td>string</td>
<td>(readonly) ID of the dashboard widget.</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>Type of the dashboard widget.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>clock - Clock;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>graph - Graph (classic);</td>
</tr>
<tr>
<td></td>
<td></td>
<td>graphprototype - Graph prototype;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>item - Item value;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>plaintext - Plain text;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>url - URL;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Custom widget name.</td>
</tr>
<tr>
<td>x</td>
<td>integer</td>
<td>A horizontal position from the left side of the dashboard.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid values range from 0 to 23.</td>
</tr>
<tr>
<td>y</td>
<td>integer</td>
<td>A vertical position from the top of the dashboard.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Valid values range from 0 to 62.</td>
</tr>
<tr>
<td>width</td>
<td>integer</td>
<td>The widget width.</td>
</tr>
<tr>
<td>height</td>
<td>integer</td>
<td>The widget height.</td>
</tr>
<tr>
<td>view_mode</td>
<td>integer</td>
<td>The widget view mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) default widget view;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - with hidden header;</td>
</tr>
</tbody>
</table>
Template dashboard widget field

The template dashboard widget field object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>integer</td>
<td>Type of the widget field.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Widget field name.</td>
</tr>
<tr>
<td>value</td>
<td>mixed</td>
<td>Widget field value depending of type.</td>
</tr>
</tbody>
</table>

**templatedashboard.create**

Description

object templatedashboard.create(object/array template Dashboards)

This method allows to create new template dashboards.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Template dashboards to create.

Additionally to the standard template dashboard properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pages</td>
<td>array</td>
<td>Template dashboard pages to be created for the dashboard. Dashboard pages will be ordered in the same order as specified. At least one dashboard page object is required for pages property.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the created template dashboards under the dashboardids property. The order of the returned IDs matches the order of the passed template dashboards.

Examples

Creating a template dashboard

Create a template dashboard named “Graphs” with one Graph widget on a single dashboard page.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "templatedashboard.create",
    "params": {
        "templateid": "10318",
        "name": "Graphs",
        "pages": [ {
            "widgets": [ 
```
See also

- Template dashboard page
- Template dashboard widget
- Template dashboard widget field

Source

CTemplateDashboard::create() in ui/include/classes/api/services/CTemplateDashboard.php.

**templatedashboard.delete**

**Description**

object templatedashboard.delete(array templateDashboardIds)

This method allows to delete template dashboards.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(array) IDs of the template dashboards to delete.

**Return values**

(object) Returns an object containing the IDs of the deleted template dashboards under the dashboardids property.

**Examples**

Deleting multiple template dashboards

Delete two template dashboards.
Request:

```json
{
    "jsonrpc": "2.0",
    "method": "templatedashboard.delete",
    "params": [
        "45",
        "46"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "45",
            "46"
        ]
    },
    "id": 1
}
```

Source

CTemplateDashboard::delete() in ui/include/classes/api/services/CTemplateDashboard.php.

**templatedashboard.get**

Description

integer/array templatedashboard.get(object parameters)

The method allows to retrieve template dashboards according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dashboardids</td>
<td>string/array</td>
<td>Return only template dashboards with the given IDs.</td>
</tr>
<tr>
<td>templateids</td>
<td>string/array</td>
<td>Return only template dashboards that belong to the given templates.</td>
</tr>
<tr>
<td>selectPages</td>
<td>string/array</td>
<td>Return a pages property with template dashboard pages, correctly ordered.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Possible values are: dashboardid and name.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td>reference commentary.</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Return values**

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

**Examples**

**Retrieving template dashboards**

Retrieve all template dashboards with widgets for a specified template.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "templatedashboard.get",
    "params": {
        "output": "extend",
        "selectPages": "extend",
        "templateids": "10001"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dashboardid": "23",
            "name": "Docker overview",
            "templateid": "10001",
            "display_period": "30",
            "auto_start": "1",
            "uuid": "6dfcbe0bc54d400ec912cd7649282f",
            "pages": [
                {
                    "dashboard_pageid": "1",
                    "name": "",
                    "display_period": "0",
                    "widgets": [
                        {
                            "widgetid": "220",
                            "type": "graph",
                            "name": "",
                            "x": "0",
                            "y": "0",
                            "width": "12",
                            "height": "5",
                            "view_mode": "0",
                            "fields": [
                                {
                                    "type": "6",
                                    "name": "graphid",
                                    "value": "1125"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}
```
See also

- Template dashboard page
- Template dashboard widget
- Template dashboard widget field

Source

CTemplateDashboard::get() in ui/include/classes/api/services/CTemplateDashboard.php.

templatedashboard.update

Description

object templatedashboard.update(object/array templateDashboards)

This method allows to update existing template dashboards.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Template dashboard properties to be updated.

The dashboardid property must be specified for each dashboard, all other properties are optional. Only the specified properties will be updated.

Additionally to the standard template dashboard properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pages</td>
<td>array</td>
<td>Template dashboard pages to replace the existing dashboard pages.</td>
</tr>
</tbody>
</table>

Dashboard pages are updated by the dashboard_pageid property. New dashboard pages will be created for objects without dashboard_pageid property and the existing dashboard pages will be deleted if not reused. Dashboard pages will be ordered in the same order as specified. Only the specified properties of the dashboard pages will be updated. At least one dashboard page object is required for pages property.

Return values

(object) Returns an object containing the IDs of the updated template dashboards under the dashboardids property.

Examples

Renaming a template dashboard

Rename a template dashboard to "Performance graphs".

Request:

{
    "jsonrpc": "2.0",
    "method": "templatedashboard.update",
}
Renaming dashboard pages:

Update the first dashboard page, replace widgets on the second dashboard page and add a new page as the third one. Delete all other dashboard pages.

Request:

```json
{  
  "jsonrpc": "2.0",  
  "method": "templatedashboard.update",  
  "params": {  
    "dashboardid": "2",  
    "pages": [  
      {  
        "dashboard_pageid": 1,  
        "name": 'Renamed Page'  
      },  
      {  
        "dashboard_pageid": 2,  
        "widgets": [  
          {  
            "type": "clock",  
            "x": 0,  
            "y": 0,  
            "width": 4,  
            "height": 3  
          }  
        ]  
      },  
      {  
        "display_period": 60  
      }  
    ]  
  },  
  "auth": "038e1d7b1735c6a5436ee9eae095879e",  
  "id": 1
}
```

Response:

```json
{  
  "jsonrpc": "2.0",  
  "result": {  
    "dashboardids": [  
      "2"  
    ]  
  },  
  "id": 1
}
```
See also

- Template dashboard widget
- Template dashboard widget field

Source

CTemplateDashboard::update() in ui/include/classes/api/services/CTemplateDashboard.php.

Template group

This class is designed to work with template groups.

Object references:

- Template group

Available methods:

- `templategroup.create` - creating new template groups
- `templategroup.delete` - deleting template groups
- `templategroup.get` - retrieving template groups
- `templategroup.massadd` - adding related objects to template groups
- `templategroup.massremove` - removing related objects from template groups
- `templategroup.massupdate` - replacing or removing related objects from template groups
- `templategroup.propagate` - propagating permissions to template groups' subgroups
- `templategroup.update` - updating template groups

> Template group object

The following objects are directly related to the `templategroup` API.

Template group

The template group object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupid</td>
<td>string</td>
<td>(readonly) ID of the template group.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the template group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(required) Universal unique identifier, used for linking imported template groups to already existing ones. Auto-generated, if not given. For update operations this field is readonly.</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

`templategroup.create`
**templategroup.create**

```php
object templategroup.create(object/array templateGroups)
```

This method allows to create new template groups.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

**Parameters**

(object/array) Template groups to create. The method accepts template groups with the [standard template group properties](#).

**Return values**

(object) Returns an object containing the IDs of the created template groups under the groupids property. The order of the returned IDs matches the order of the passed template groups.

**Examples**

Creating a template group

Create a template group called “Templates/Databases”.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "templategroup.create",
   "params": {
      "name": "Templates/Databases"
   },
   "auth": "038e1d7b1735c6a5436ee9eae096879e",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "groupids": [
         "107820"
      ]
   },
   "id": 1
}
```

**Source**

CTemplateGroup::create() in ui/include/classes/api/services/CTemplateGroup.php.

---

**templategroup.delete**

**Description**

```php
object templategroup.delete(array templateGroupIds)
```

This method allows to delete template groups.

A template group can not be deleted if it contains templates that belong to this group only.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

**Parameters**

(array) IDs of the template groups to delete.

**Return values**

(object) Returns an object containing the IDs of the deleted template groups under the groupids property.

**Examples**

```json
```
Deleting multiple template groups

Delete two template groups.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "templategroup.delete",
    "params": [
        "107814",
        "107815"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107814",
            "107815"
        ]
    },
    "id": 1
}
```

Source

CTemplateGroup::delete() in ui/include/classes/api/services/CTemplateGroup.php.

**templategroup.get**

**Description**

integer/array templategroup.get(object parameters)

The method allows to retrieve template groups according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphids</td>
<td>string/array</td>
<td>Return only template groups that contain templates with the given graphs. Return only template groups with the given template group IDs. Return only template groups that contain the given templates.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only template groups that contain the given templates.</td>
</tr>
<tr>
<td>templateids</td>
<td>string/array</td>
<td>Return only template groups that contain the given templates.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>triggerids</td>
<td>string/array</td>
<td>Return only template groups that contain templates with the given triggers.</td>
</tr>
<tr>
<td>with_graphs</td>
<td>flag</td>
<td>Return only template groups that contain templates with graphs.</td>
</tr>
<tr>
<td>with_graph_prototypes</td>
<td>flag</td>
<td>Return only template groups that contain templates with graph prototypes.</td>
</tr>
<tr>
<td>with_httptests</td>
<td>flag</td>
<td>Return only template groups that contain templates with web checks.</td>
</tr>
<tr>
<td>with_items</td>
<td>flag</td>
<td>Return only template groups that contain templates with items.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overrides the with_simple_graph_items parameter.</td>
</tr>
<tr>
<td>with_item_prototypes</td>
<td>flag</td>
<td>Return only template groups that contain templates with item prototypes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overrides the with_simple_graph_item_prototypes parameter.</td>
</tr>
<tr>
<td>with_simple_graph_prototypes</td>
<td>flag</td>
<td>Return only template groups that contain templates with item prototypes, which are enabled for creation and have numeric type of information.</td>
</tr>
<tr>
<td>with_templates</td>
<td>flag</td>
<td>Return only template groups that contain templates with numeric items.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Return only template groups that contain templates.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>with_triggers</td>
<td>flag</td>
<td>Return only template groups that contain templates with triggers.</td>
</tr>
<tr>
<td>selectTemplates</td>
<td>query</td>
<td>Return a templates property with the templates that belong to the template group.</td>
</tr>
<tr>
<td>limitSelects</td>
<td>integer</td>
<td>Supports count. Limits the number of records returned by subselects.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties. Possible values are: groupid, name.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary page.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values
(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving data by name
Retrieve all data about two template groups named "Templates/Databases" and "Templates/Modules".

Request:

```
{
   "jsonrpc": "2.0",
   "method": "templategroup.get",
   "params": {
      "output": "extend",
      "filter": {
         "name": [
            "Templates/Databases",
            "Templates/Modules"
         ]
      }
   },
   "auth": "6f38cddc44cfbb6c1bd186f9a220b5a0",
   "id": 1
}
```

Response:

```
{
   "jsonrpc": "2.0",
   "result": [
      {
         "groupid": "13",
         "name": "Templates/Databases",
         "uuid": "748ad4d098d447d492bb935c907f652f"
      },
      {
         "groupid": "8",
         "name": "Templates/Modules",
         "uuid": "57b7ae836ca64446ba2c296389c009b7"
      }
   ],
   "id": 1
}
```

See also

- Template Source

CTemplateGroup::get() in ui/include/classes/api/services/CTemplateGroup.php.

**templategroup.massadd**

**Description**

object templategroup.massadd(object parameters)

This method allows to simultaneously add multiple related objects to all the given template groups.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object) Parameters containing the IDs of the template groups to update and the objects to add to all the template groups.

The method accepts the following parameters.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Template groups to be updated.</td>
</tr>
<tr>
<td></td>
<td>(required)</td>
<td>The template groups must have the groupid property defined.</td>
</tr>
<tr>
<td>templates</td>
<td>object/array</td>
<td>Templates to add to all template groups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The templates must have the templateid property defined.</td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns an object containing the IDs of the updated template groups under the groupids property.

**Examples**

**Adding templates to template groups**

Add two templates to template groups with IDs 12 and 13.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "templategroup.massadd",
    "params": {
        "groups": [
            {
                "groupid": "12"
            },
            {
                "groupid": "13"
            }
        ],
        "templates": [
            {
                "templateid": "10486"
            },
            {
                "templateid": "10487"
            }
        ]
    },
    "auth": "f223adf833b2bf2ff38574a67bba6372",
    "id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "12",
            "13"
        ]
    },
}
```
See also

- Template

Source

CTemplateGroup::massAdd() in ui/include/classes/api/services/CTemplateGroup.php.

templategroup.massremove

Description

object templategroup.massremove(object parameters)

This method allows to remove related objects from multiple template groups.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters containing the IDs of the template groups to update and the objects that should be removed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>IDs of the template groups to be updated.</td>
</tr>
<tr>
<td></td>
<td>(required)</td>
<td></td>
</tr>
<tr>
<td>templateids</td>
<td>string/array</td>
<td>Templates to remove from all template groups.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated template groups under the groupids property.

Examples

Removing templates from template groups

Remove two templates from the given template groups.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "hostgroup.massremove",
    "params": {
        "groupids": ["5", "6"],
        "hostids": ["30050", "30001"]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
```
Source
CTemplateGroup::massRemove() in ui/include/classes/api/services/CTemplateGroup.php.

templategroup.massupdate

Description

object templategroup.massupdate(object parameters)

This method allows to replace templates with the specified ones in multiple template groups.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters containing the IDs of the template groups to update and the objects that should be updated.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Template groups to be updated. The template groups must have the groupid property defined.</td>
</tr>
<tr>
<td>templates</td>
<td>object/array</td>
<td>Templates to replace the current template on the given template groups. All other templates, except the ones mentioned, will be excluded from template groups. The templates must have the templateid property defined.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated template groups under the groupids property.

Examples

Replacing templates in a template group

Replace all templates in a template group to ones mentioned templates.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "templategroup.massupdate",
    "params": {
        "groups": [1, 2, 3],
        "templates": [10, 20, 30]
    }
}
```
Response:
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "8",
        ],
        "id": 1
    }
}

See also
- `templategroup.update`
- `templategroup.massadd`
- `Template`

Source
CTemplateGroup::massUpdate() in ui/include/classes/api/services/CTemplateGroup.php.

**templategroup.propagate**

Description
object templategroup.propagate(object parameters)

This method allows to apply permissions to all template groups' subgroups.

This method is only available to Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>object/array</td>
<td>Template groups to propagate.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>permissions</td>
<td>boolean</td>
<td>The template groups must have the groupid property defined.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>Set <code>true</code> if need to propagate permissions.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the propagated template groups under the groupids property.

Examples

Propagating template group permissions to its subgroups.
Propagate template group permissions to its subgroups.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "templategroup.propagate",
   "params": {
      "groups": [
         {
            "groupid": "15"
         }
      ],
      "permissions": true
   },
   "auth": "f223adf833b2bf2ff38574a67bba6372",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "groupids": [
         "15",
      ]
   },
   "id": 1
}
```

See also

- `templategroup.update`
- `templategroup.massadd`
- `Template`

Source

CTemplateGroup::propagate() in ui/include/classes/api/services/CTemplateGroup.php.

**templategroup.update**

Description

object templategroup.update(object/array templateGroups)

This method allows to update existing template groups.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Template group properties to be updated.

The groupid property must be defined for each template group, all other properties are optional. Only the given properties will be updated, all others will remain unchanged.

Return values

(object) Returns an object containing the IDs of the updated template groups under the groupids property.

Examples

Renaming a template group

Rename a template group to “Templates/Databases”

Request:
{ "jsonrpc": "2.0", "method": "templategroup.update", "params": { "groupid": "7", "name": "Templates/Databases" }, "auth": "700ca65537074ec963db7efabda78259", "id": 1 }

Response:
{
  "jsonrpc": "2.0",
  "result": {
    "groupids": [
      "7"
    ],
  },
  "id": 1
}

Source
CTemplateGroup::update() in ui/include/classes/api/services/CTemplateGroup.php.

**Token**

This class is designed to work with tokens.

Object references:

- **Token**

Available methods:

- `token.create` - create new tokens
- `token.delete` - delete tokens
- `token.get` - retrieve tokens
- `token.update` - update tokens
- `token.generate` - generate tokens

> **Token object**

The following objects are directly related to the token API.

**Token**

The token object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tokenid</td>
<td>string</td>
<td>(readonly) ID of the token.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the token.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>text</td>
<td>Description of the token.</td>
</tr>
<tr>
<td>userid</td>
<td>string</td>
<td>Description of the token. (readonly for update) A user the token has been assigned to.</td>
</tr>
<tr>
<td>lastaccess</td>
<td>timestamp</td>
<td>Default: current user. (readonly) Most recent date and time the token was authenticated.</td>
</tr>
</tbody>
</table>

Zero if the token has never been authenticated.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>integer</td>
<td>Token status. Possible values: 0 (default) enabled token; 1 - disabled token.</td>
</tr>
<tr>
<td>expires_at</td>
<td>timestamp</td>
<td>Token expiration date and time. Zerofornever-expiringtokens.</td>
</tr>
<tr>
<td>created_at</td>
<td>timestamp</td>
<td>(readonly) Token creation date and time.</td>
</tr>
<tr>
<td>creator_userid</td>
<td>string</td>
<td>(readonly) The creator user of the token.</td>
</tr>
</tbody>
</table>

**token.create**

Description

object token.create(object/array tokens)

This method allows to create new tokens.

Only Super admin user type is allowed to manage tokens for other users.

A token created by this method has to be generated before it is usable.

Parameters

(object/array) Tokens to create.

The method accepts tokens with the standard token properties.

Return values

(object) Returns an object containing the IDs of the created tokens under the tokenids property. The order of the returned IDs matches the order of the passed tokens.

Examples

Create a token

Create an enabled token that never expires and authenticates user of ID 2.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "token.create",
   "params": {
      "name": "Your token",
      "userid": "2"
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "tokenids": [
         "188"
      ],
   "id": 1
}
```

Create a disabled token that expires at January 21st, 2021. This token will authenticate current user.

Request:
JSON object:

```json
{
  "jsonrpc": "2.0",
  "method": "token.create",
  "params": {
    "name": "Your token",
    "status": "1",
    "expires_at": "1611238072"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": {
    "tokenids": ["189"]
  },
  "id": 1
}
```

Source

CToken::create() in ui/include/classes/api/services/CToken.php.

token.delete

Description

object token.delete(array tokens)

This method allows to delete tokens.

Only Super admin user type is allowed to manage tokens for other users.

Parameters

(array) IDs of the token to delete.

Return values

(object) Returns an object containing the IDs of the deleted tokens under the tokenids property.

Examples

Delete multiple tokens

Delete two tokens.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "token.delete",
  "params": [
    "188",
    "192"
  ],
  "auth": "3a57200802b24cda67c4e4010b50c065",
  "id": 1
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": {
  }
}
```
JSON

```json
{
    "tokenids": [
        "188",
        "192"
    ],
    "id": 1
}
```

Source

CToken::delete() in ui/include/classes/api/services/CToken.php.

**token.generate**

**Description**

object token.generate(array tokenids)

This method allows to generate tokens.

Only Super admin user type is allowed to manage tokens for other users.

**Parameters**

(array) IDs of the tokens to generate.

**Return values**

(array) Returns an array of objects containing the ID of the generated token under the tokenid property and generated authorization string under token property.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tokenid</td>
<td>string</td>
<td>ID of the token.</td>
</tr>
<tr>
<td>token</td>
<td>string</td>
<td>The generated authorization string for this token.</td>
</tr>
</tbody>
</table>

**Examples**

Generate multiple tokens

Generate two tokens.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "token.generate",
    "params": [
        "1",
        "2"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "tokenid": "1",
            "token": "bbcfce79a2d95037502f7e9a534906d3466c9a1484beb6ea0f4e7be28e8b8ce2"
        },
        {
            "tokenid": "2",
            "token": "fa1258a83d518eab87698a96bd7f07e5a6ae8aeb8463cae33d50b91dd21bd6d"
        }
    ]
}
```
token.get

Description

integer/array token.get(object parameters)

The method allows to retrieve tokens according to the given parameters.

Only Super admin user type is allowed to view tokens for other users.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tokenids</td>
<td>string/array</td>
<td>Return only tokens with the given IDs.</td>
</tr>
<tr>
<td>userids</td>
<td>string/array</td>
<td>Return only tokens created for the given users.</td>
</tr>
<tr>
<td>token</td>
<td>string</td>
<td>Return only tokens created for the given Auth token.</td>
</tr>
<tr>
<td>valid_at</td>
<td>timestamp</td>
<td>Return only tokens which are valid (not expired) at the given date and time.</td>
</tr>
<tr>
<td>expired_at</td>
<td>timestamp</td>
<td>Return only tokens which are expired (not valid) at the given date and time.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are: tokenid, name, lastaccess, status, expires_at and created_at.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieve an token

Retrieve all data for token with ID "2".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "token.get",
    "params": {
        "output": "extend",
```
"tokens": "2",
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": [
    {
      "tokenid": "1",
      "name": "The Token",
      "description": "",
      "userid": "1",
      "lastaccess": "0",
      "status": "0",
      "expires_at": "1609406220",
      "created_at": "1611239454",
      "creator_userid": "1"
    }
  ],
  "id": 1
}

Source
CToken::get() in ui/include/classes/api/services/CToken.php.

token.update

Description
object token.update(object/array tokens)
This method allows to update existing tokens.
Only Super admin user type is allowed to manage tokens for other users.

Parameters
(object/array) Token properties to be updated.
The tokenid property must be defined for each token, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.
The method accepts tokens with the standard token properties.

Return values
(object) Returns an object containing the IDs of the updated tokens under the tokenids property.

Examples
Remove token expiry
Remove expiry date from token.

Request:
{
  "jsonrpc": "2.0",
  "method": "token.update",
  "params": {
    "tokenid": "2",
    "expires_at": "0"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "tokenids": [
            "2"
        ],
        "id": 1
    }
}
```

Source

`CToken::update()` in `ui/include/classes/api/services/CToken.php`.

**Trend**

This class is designed to work with trend data.

Object references:

- `Trend`

Available methods:

- `trend.get` - retrieving trends

> **Trend object**

The following objects are directly related to the `trend` API.

Trend objects differ depending on the item’s type of information. They are created by the Zabbix server and cannot be modified via the API.

**Float trend**

The float trend object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock</td>
<td>timestamp</td>
<td>Timestamp of an hour for which the value was calculated. E. g.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>timestamp of 04:00:00 means values calculated for period 04:00:00-04:59:59.</td>
</tr>
<tr>
<td>itemid</td>
<td>integer</td>
<td>ID of the related item.</td>
</tr>
<tr>
<td>num</td>
<td>integer</td>
<td>Number of values that were available for the hour.</td>
</tr>
<tr>
<td>value_min</td>
<td>float</td>
<td>Hourly minimum value.</td>
</tr>
<tr>
<td>value_avg</td>
<td>float</td>
<td>Hourly average value.</td>
</tr>
<tr>
<td>value_max</td>
<td>float</td>
<td>Hourly maximum value.</td>
</tr>
</tbody>
</table>

**Integer trend**

The integer trend object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock</td>
<td>timestamp</td>
<td>Timestamp of an hour for which the value was calculated. E. g.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>timestamp of 04:00:00 means values calculated for period 04:00:00-04:59:59.</td>
</tr>
<tr>
<td>itemid</td>
<td>integer</td>
<td>ID of the related item.</td>
</tr>
<tr>
<td>num</td>
<td>integer</td>
<td>Number of values that were available for the hour.</td>
</tr>
<tr>
<td>value_min</td>
<td>integer</td>
<td>Hourly minimum value.</td>
</tr>
<tr>
<td>value_avg</td>
<td>integer</td>
<td>Hourly average value.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>value_max</td>
<td>integer</td>
<td>Hourly maximum value.</td>
</tr>
</tbody>
</table>

**trend.get**

Description

integer/array trend.get(object parameters)

The method allows to retrieve trend data according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemids</td>
<td>string/array</td>
<td>Return only trends with the given item IDs.</td>
</tr>
<tr>
<td>time_from</td>
<td>timestamp</td>
<td>Return only values that have been collected after or at the given time.</td>
</tr>
<tr>
<td>time_till</td>
<td>timestamp</td>
<td>Return only values that have been collected before or at the given time.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Count the number of retrieved objects.</td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td>Limit the amount of retrieved objects.</td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td>Set fields to output.</td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving item trend data

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "trend.get",
    "params": {
        "output": [
            "itemid",
            "clock",
            "num",
            "value_min",
            "value_avg",
            "value_max",
        ],
        "itemids": [
            "23715"
        ],
        "limit": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:
```json
{
"jsonrpc": "2.0",
"result": [
{
"itemid": "23715",
"clock": "1446199200",
"num": "60",
"value_min": "0.165",
"value_avg": "0.2168",
"value_max": "0.35",
}
],
"id": 1
}
```

Source
CTrend::get() in ui/include/classes/api/services/CTrend.php.

**Trigger**

This class is designed to work with triggers.

Object references:

* **Trigger**

Available methods:

* trigger.create - creating new triggers
* trigger.delete - deleting triggers
* trigger.get - retrieving triggers
* trigger.update - updating triggers

> **Trigger object**

The following objects are directly related to the trigger API.

**Trigger**

The trigger object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>triggerid</td>
<td>string</td>
<td>(readonly) ID of the trigger.</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Name of the trigger.</td>
</tr>
<tr>
<td>expression</td>
<td>string</td>
<td>Reduced trigger expression.</td>
</tr>
<tr>
<td>event_name</td>
<td>string</td>
<td>Event name generated by the trigger.</td>
</tr>
<tr>
<td>opdata</td>
<td>string</td>
<td>Operational data.</td>
</tr>
<tr>
<td>comments</td>
<td>string</td>
<td>Additional description of the trigger.</td>
</tr>
<tr>
<td>error</td>
<td>string</td>
<td>(readonly) Error text if there have been any problems when updating the state of the trigger.</td>
</tr>
<tr>
<td>flags</td>
<td>integer</td>
<td>(readonly) Origin of the trigger.</td>
</tr>
</tbody>
</table>

Possible values are:

0 - (default) a plain trigger;
4 - a discovered trigger.

lastchange   | timestamp | (readonly) Time when the trigger last changed its state.                   |
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority</td>
<td>integer</td>
<td>Severity of the trigger.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) not classified;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - information;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - warning;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - average;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - high;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - disaster.</td>
</tr>
<tr>
<td>state</td>
<td>integer</td>
<td>(readonly) State of the trigger.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) trigger state is up to date;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - current trigger state is unknown.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Whether the trigger is enabled or disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) enabled;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - disabled.</td>
</tr>
<tr>
<td>templateid</td>
<td>string</td>
<td>(readonly) ID of the parent template trigger.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>Whether the trigger can generate multiple problem events.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) do not generate multiple events;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - generate multiple events.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>URL associated with the trigger.</td>
</tr>
<tr>
<td>value</td>
<td>integer</td>
<td>(readonly) Whether the trigger is in OK or problem state.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) OK;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - problem.</td>
</tr>
<tr>
<td>recovery_mode</td>
<td>integer</td>
<td>OK event generation mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Expression;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Recovery expression;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - None.</td>
</tr>
<tr>
<td>recovery_expression</td>
<td>string</td>
<td>Reduced trigger recovery expression.</td>
</tr>
<tr>
<td>correlation_mode</td>
<td>integer</td>
<td>OK event closes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) All problems;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - All problems if tag values match.</td>
</tr>
<tr>
<td>correlation_tag</td>
<td>string</td>
<td>Tag for matching.</td>
</tr>
<tr>
<td>manual_close</td>
<td>integer</td>
<td>Allow manual close.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) No;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Yes.</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Universal unique identifier, used for linking imported triggers to already</td>
</tr>
<tr>
<td></td>
<td></td>
<td>existing ones. Used only for triggers on templates.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Auto-generated, if not given.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For update operations this field is readonly.</td>
</tr>
</tbody>
</table>

**Trigger tag**

The trigger tag object has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Trigger tag name.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Trigger tag value.</td>
</tr>
</tbody>
</table>

**trigger.create**

**Description**

object trigger.create(object/array triggers)

This method allows to create new triggers.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) Triggers to create.

Additionally to the standard trigger properties the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependencies</td>
<td>array</td>
<td>Triggers that the trigger is dependent on.</td>
</tr>
<tr>
<td>tags</td>
<td>array</td>
<td>The triggers must have the triggerid property defined.</td>
</tr>
</tbody>
</table>

The trigger expression has to be given in its expanded form.

**Return values**

(object) Returns an object containing the IDs of the created triggers under the triggerids property. The order of the returned IDs matches the order of the passed triggers.

**Examples**

Creating a trigger

Create a trigger with a single trigger dependency.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "trigger.create",
  "params": [
    {
      "description": "Processor load is too high on {HOST.NAME}"
    },
    {
      "description": "Service status",
      "dependencies": [
        {
          "triggerid": "17367"
        }
      ],
      "tags": [ |
```
"tag": "service",
"value": "{{ITEM.VALUE}.regsub("Service (.*) has stopped", \"\\\"\")}",
},
{ "tag": "error",
"value": ""
}
],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": {
    "triggerids": [
      "17369",
      "17370"
    ],
    "id": 1
  }
}

Source
CTrigger::create() in ui/include/classes/api/services/CTrigger.php.

trigger.delete

Description
object trigger.delete(array triggerIds)

This method allows to delete triggers.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(array) IDs of the triggers to delete.

Return values
(object) Returns an object containing the IDs of the deleted triggers under the triggerids property.

Examples
Delete multiple triggers

Delete two triggers.

Request:
{
  "jsonrpc": "2.0",
  "method": "trigger.delete",
  "params": [
    "12002",
    "12003"
  ],
  "auth": "3a57200802b24cda67c4e4010b50c065",
  "id": 1
}

Response:
trigger.get

Description

integer/array trigger.get(object parameters)

The method allows to retrieve triggers according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>triggerids</td>
<td>string/array</td>
<td>Return only triggers with the given IDs.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only triggers that belong to hosts or templates from the given host groups or template groups.</td>
</tr>
<tr>
<td>templateids</td>
<td>string/array</td>
<td>Return only triggers that belong to the given templates.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only triggers that belong to the given hosts.</td>
</tr>
<tr>
<td>itemids</td>
<td>string/array</td>
<td>Return only triggers that contain the given items.</td>
</tr>
<tr>
<td>functions</td>
<td>string/array</td>
<td>Return only triggers that use the given functions.</td>
</tr>
</tbody>
</table>

Refer to the supported function page for a list of supported functions.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>string</td>
<td>Return only triggers that belong to hosts or templates from the host group or template group with the given name.</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>Return only triggers that belong to host with the given name.</td>
</tr>
<tr>
<td>inherited</td>
<td>boolean</td>
<td>If set to true return only triggers inherited from a template.</td>
</tr>
<tr>
<td>templated</td>
<td>boolean</td>
<td>If set to true return only triggers that belong to templates.</td>
</tr>
<tr>
<td>dependent</td>
<td>boolean</td>
<td>If set to true return only triggers that have dependencies. If set to false return only triggers that do not have dependencies.</td>
</tr>
<tr>
<td>monitored</td>
<td>flag</td>
<td>Return only enabled triggers that belong to monitored hosts and contain only enabled items.</td>
</tr>
<tr>
<td>active</td>
<td>flag</td>
<td>Return only enabled triggers that belong to monitored hosts.</td>
</tr>
<tr>
<td>maintenance</td>
<td>boolean</td>
<td>If set to true return only enabled triggers that belong to hosts in maintenance.</td>
</tr>
<tr>
<td>withUnacknowledgedEvents</td>
<td>flag</td>
<td>Return only triggers that have unacknowledged events.</td>
</tr>
<tr>
<td>withAcknowledgedEvents</td>
<td>flag</td>
<td>Return only triggers with all events acknowledged.</td>
</tr>
<tr>
<td>withLastEventUnacknowledged</td>
<td>flag</td>
<td>Return only triggers with the last event unacknowledged.</td>
</tr>
<tr>
<td>skipDependent</td>
<td>flag</td>
<td>Skip triggers in a problem state that are dependent on other triggers. Note that the other triggers are ignored if disabled, have disabled items or disabled item hosts.</td>
</tr>
<tr>
<td>lastChangeSince</td>
<td>timestamp</td>
<td>Return only triggers that have changed their state after the given time.</td>
</tr>
<tr>
<td>lastChangeTill</td>
<td>timestamp</td>
<td>Return only triggers that have changed their state before the given time.</td>
</tr>
<tr>
<td>only_true</td>
<td>flag</td>
<td>Return only triggers that have recently been in a problem state.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>min_severity</td>
<td>integer</td>
<td>Return only triggers with severity greater or equal than the given severity.</td>
</tr>
<tr>
<td>evaltype</td>
<td>integer</td>
<td>Rules for tag searching.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) And/Or;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Or.</td>
</tr>
<tr>
<td>tags</td>
<td>array of objects</td>
<td>Return only triggers with given tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value. Format: [{&quot;tag&quot;: &quot;&lt;tag&gt;&quot;, &quot;value&quot;: &quot;&lt;value&gt;&quot;, &quot;operator&quot;: &quot;&lt;operator&gt;&quot;, ...}]. An empty array returns all triggers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible operator types:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Like;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Equal;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Not like;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - Not equal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - Exists;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - Not exists.</td>
</tr>
<tr>
<td>expandComment</td>
<td>flag</td>
<td>Expand macros in the trigger description.</td>
</tr>
<tr>
<td>expandDescription</td>
<td>flag</td>
<td>Expand macros in the name of the trigger.</td>
</tr>
<tr>
<td>expandExpression</td>
<td>flag</td>
<td>Expand functions and macros in the trigger expression.</td>
</tr>
<tr>
<td>selectHostGroups</td>
<td>query</td>
<td>Return the host groups that the trigger belongs to in the host groups property.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return the hosts that the trigger belongs to in the hosts property.</td>
</tr>
<tr>
<td>selectItems</td>
<td>query</td>
<td>Return items contained by the trigger in the items property.</td>
</tr>
<tr>
<td>selectFunctions</td>
<td>query</td>
<td>Return functions used in the trigger in the functions property.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The function objects represent the functions used in the trigger expression and has the following properties: functionid - (string) ID of the function; itemid - (string) ID of the item used in the function; function - (string) name of the function; parameter - (string) parameter passed to the function. Query parameter is replaced by $ symbol in returned string.</td>
</tr>
<tr>
<td>selectDependencies</td>
<td>query</td>
<td>Return triggers that the trigger depends on in the dependencies property.</td>
</tr>
<tr>
<td>selectDiscoveryRule</td>
<td>query</td>
<td>Return the low-level discovery rule that created the trigger.</td>
</tr>
<tr>
<td>selectLastEvent</td>
<td>query</td>
<td>Return the last significant trigger event in the lastEvent property.</td>
</tr>
<tr>
<td>selectTags</td>
<td>query</td>
<td>Return the trigger tags in tags property.</td>
</tr>
<tr>
<td>selectTemplateGroups</td>
<td>query</td>
<td>Return the template groups that the trigger belongs to in the template groups property.</td>
</tr>
<tr>
<td>selectTriggerDiscovery</td>
<td>query</td>
<td>Return the trigger discovery object in the triggerDiscovery property. The trigger discovery objects link the trigger to a trigger prototype from which it was created.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It has the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parent_triggerid - (string) ID of the trigger prototype from which the trigger has been created.</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td>Return only those results that exactly match the given filter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supports additional filters:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>host - technical name of the host that the trigger belongs to;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hostid - ID of the host that the trigger belongs to.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>limitSelects</td>
<td>integer</td>
<td>Limits the number of records returned by subselects.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Applies to the following subselects: selectHosts - results will be sorted by host.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sortfield string/array</td>
</tr>
<tr>
<td></td>
<td></td>
<td>countOutput boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>editable boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>excludeSearch boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>limit integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>output query</td>
</tr>
<tr>
<td></td>
<td></td>
<td>preservekeys boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>search object</td>
</tr>
<tr>
<td></td>
<td></td>
<td>searchByAny boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>searchWildcardsEnabled boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sortOrder string/array</td>
</tr>
<tr>
<td></td>
<td></td>
<td>startSearch boolean</td>
</tr>
<tr>
<td>selectGroups</td>
<td>query</td>
<td>(deprecated) This parameter is deprecated, please use selectHostGroups or selectTemplateGroups instead.</td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:
- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving data by trigger ID

Retrieve all data and the functions used in trigger "14062".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "triggerids": "14062",
        "output": "extend",
        "selectFunctions": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae096879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "14062",
            "expression": "{13513}<10m",
            "description": "{HOST.NAME} has been restarted (uptime < 10m)",
            "url": "",
            "status": "0",
            "value": "0",
            "priority": "2",
        }
    ]
}
```
Retrieving triggers in problem state

Retrieve the ID, name and severity of all triggers in problem state and sort them by severity in descending order.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "output": ["triggerid", "description", "priority"],
        "filter": {
            "value": 1
        },
        "sortfield": "priority",
        "sortorder": "DESC"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "13907",
            "description": "Zabbix self-monitoring processes < 100% busy",
            "priority": "4"
        },
        {
            "triggerid": "13824",
            "description": "Zabbix discoverer processes more than 75% busy",
```
Retrieving a specific trigger with tags

Retrieve a specific trigger with tags.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "output": [
            "triggerid",
            "description"
        ],
        "selectTags": "extend",
        "triggerids": [
            "17578"
        ],
        "auth": "038e1d7b1735c6a5436ee9eae095879e",
        "id": 1
    }
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "17370",
            "description": "Service status",
            "tags": [
                {
                    "tag": "service",
                    "value": "{{ITEM.VALUE}.regsub("Service (.*) has stopped", "\1")}"
                },
                {
                    "tag": "error",
                    "value": ""
                }
            ]
        }
    ],
    "id": 1
}
```

See also
- Discovery rule
- Item
- Host
- Host group
- Template group

Source
CTrigger::get() in ui/include/classes/api/services/CTrigger.php.

trigger.update
object trigger.update(object/array triggers)

This method allows to update existing triggers.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Trigger properties to be updated.

The triggerid property must be defined for each trigger, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard trigger properties the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependencies</td>
<td>array</td>
<td>Triggers that the trigger is dependent on.</td>
</tr>
<tr>
<td>tags</td>
<td>array</td>
<td>The triggers must have the triggerid property defined. Trigger tags.</td>
</tr>
</tbody>
</table>

The trigger expression has to be given in its expanded form.

Return values

(object) Returns an object containing the IDs of the updated triggers under the triggerids property.

Examples

Enabling a trigger

Enable a trigger, that is, set its status to 0.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": {
        "triggerid": "13938",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ],
        "id": 1
    }
}
```

Replacing triggers tags

Replace tags for trigger.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": {
```
"triggerid": "13938",
"tags": [
{
"tag": "service",
"value": "{ITEM.VALUE}.regexp("Service (\*) has stopped", "\\\"\\\")}'
},
{
"tag": "error",
"value": "
}
],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
"jsonrpc": "2.0",
"result": {
"triggerids": [
"13938"
]
},
"id": 1
}

Replacing dependencies

Replace dependencies for trigger.

Request:
{
"jsonrpc": "2.0",
"method": "trigger.update",
"params": {
"triggerid": "22713",
"dependencies": [
{ "triggerid": "22712"
},
{ "triggerid": "22772"
}
]
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
"jsonrpc": "2.0",
"result": {
"triggerids": [
"22713"
]
},
"id": 1
}

Source
CTrigger::update() in ui/include/classes/api/services/CTrigger.php.
Trigger prototype

This class is designed to work with trigger prototypes.

Object references:

- Trigger prototype

Available methods:

- triggerprototype.create - creating new trigger prototypes
- triggerprototype.delete - deleting trigger prototypes
- triggerprototype.get - retrieving trigger prototypes
- triggerprototype.update - updating trigger prototypes

> Trigger prototype object

The following objects are directly related to the triggerprototype API.

Trigger prototype

The trigger prototype object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>triggerid</td>
<td>string</td>
<td>(readonly) ID of the trigger prototype.</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Name of the trigger prototype.</td>
</tr>
<tr>
<td>expression</td>
<td>string</td>
<td>Reduced trigger expression.</td>
</tr>
<tr>
<td>event_name</td>
<td>string</td>
<td>Event name generated by the trigger.</td>
</tr>
<tr>
<td>opdata</td>
<td>string</td>
<td>Operational data.</td>
</tr>
<tr>
<td>comments</td>
<td>string</td>
<td>Additional comments to the trigger prototype.</td>
</tr>
<tr>
<td>priority</td>
<td>integer</td>
<td>Severity of the trigger prototype.</td>
</tr>
<tr>
<td>status</td>
<td>integer</td>
<td>Whether the trigger prototype is enabled or disabled.</td>
</tr>
<tr>
<td>templateid</td>
<td>string</td>
<td>(readonly) ID of the parent template trigger prototype.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>Whether the trigger prototype can generate multiple problem events.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>URL associated with the trigger prototype.</td>
</tr>
<tr>
<td>recovery_mode</td>
<td>integer</td>
<td>OK event generation mode.</td>
</tr>
<tr>
<td>recovery_expression</td>
<td>string</td>
<td>Reduced trigger recovery expression.</td>
</tr>
</tbody>
</table>

Possible values for priority:
0 - (default) not classified;
1 - information;
2 - warning;
3 - average;
4 - high;
5 - disaster.

Possible values for status:
0 - (default) enabled;
1 - disabled.

Possible values for type:
0 - (default) do not generate multiple events;
1 - generate multiple events.

Possible values for recovery_mode:
0 - (default) Expression;
1 - Recovery expression;
2 - None.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>correlation_mode</td>
<td>integer</td>
<td>OK event closes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) All problems;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - All problems if tag values match.</td>
</tr>
<tr>
<td>correlation_tag</td>
<td>string</td>
<td>Tag for matching.</td>
</tr>
<tr>
<td>manual_close</td>
<td>integer</td>
<td>Allow manual close.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) No;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Yes.</td>
</tr>
<tr>
<td>discover</td>
<td>integer</td>
<td>Trigger prototype discovery status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) new triggers will be discovered;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - new triggers will not be discovered and existing triggers will be marked as lost.</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Universal unique identifier, used for linking imported trigger prototypes to already existing ones. Used only for trigger prototypes on templates. Auto-generated, if not given.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For update operations this field is readonly.</td>
</tr>
</tbody>
</table>

Trigger prototype tag

The trigger prototype tag object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Trigger prototype tag name.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Trigger prototype tag value.</td>
</tr>
</tbody>
</table>

triggerprototype.create

Description

Object triggerprototype.create(object/array triggerPrototypes)

This method allows to create new trigger prototypes.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(Object/array) Trigger prototypes to create.

Additionally to the standard trigger prototype properties the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependencies</td>
<td>array</td>
<td>Triggers and trigger prototypes that the trigger prototype is dependent on.</td>
</tr>
<tr>
<td>tags</td>
<td>array</td>
<td>The triggers must have the triggerid property defined. Trigger prototype tags.</td>
</tr>
</tbody>
</table>

The trigger expression has to be given in its expanded form and must contain at least one item prototype.

Return values

(Object) Returns an object containing the IDs of the created trigger prototypes under the triggerids property. The order of the returned IDs matches the order of the passed trigger prototypes.
Examples

Creating a trigger prototype

Create a trigger prototype to detect when a file system has less than 20% free disk space.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.create",
    "params": {
        "description": "Free disk space is less than 20% on volume {#FSNAME}'",
        "expression": "last(/Zabbix server/vfs.fs.size[#{FSNAME},pFree])<20",
        "tags": [
            {
                "tag": "volume",
                "value": "#{FSNAME}"
            },
            {
                "tag": "type",
                "value": "#{FSTYPE}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": ["17372"]
    },
    "id": 1
}
```

Source

CTriggerPrototype::create() in ui/include/classes/api/services/CTriggerPrototype.php.

**triggerprototype.delete**

Description

object triggerprototype.delete(array triggerPrototypeIds)

This method allows to delete trigger prototypes.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

Parameters

(array) IDs of the trigger prototypes to delete.

Return values

(object) Returns an object containing the IDs of the deleted trigger prototypes under the triggerIds property.

Examples

Deleting multiple trigger prototypes
Delete two trigger prototypes.

Request:
triggerprototype.get

integer/array triggerprototype.get(object parameters)

The method allows to retrieve trigger prototypes according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>flag</td>
<td>Return only enabled trigger prototypes that belong to monitored hosts.</td>
</tr>
<tr>
<td>discoveryids</td>
<td>string/array</td>
<td>Return only trigger prototypes that belong to the given LLD rules.</td>
</tr>
<tr>
<td>functions</td>
<td>string/array</td>
<td>Return only triggers that use the given functions.</td>
</tr>
</tbody>
</table>

Refer to the supported trigger functions page for a list of supported functions.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>string</td>
<td>Return only trigger prototypes that belong to hosts or templates from the host groups or template groups with the given name.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only trigger prototypes that belong to hosts or templates from the given host groups or template groups.</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>Return only trigger prototypes that belong to hosts with the given name.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only trigger prototypes that belong to the given hosts.</td>
</tr>
<tr>
<td>inherited</td>
<td>boolean</td>
<td>If set to true return only trigger prototypes inherited from a template.</td>
</tr>
<tr>
<td>maintenance</td>
<td>boolean</td>
<td>If set to true return only enabled trigger prototypes that belong to hosts in maintenance.</td>
</tr>
<tr>
<td>min_severity</td>
<td>integer</td>
<td>Return only trigger prototypes with severity greater or equal than the given severity.</td>
</tr>
<tr>
<td>monitored</td>
<td>flag</td>
<td>Return only enabled trigger prototypes that belong to monitored hosts and contain only enabled items.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>templated</td>
<td>boolean</td>
<td>If set to true return only trigger prototypes that belong to templates.</td>
</tr>
<tr>
<td>templateids</td>
<td>string/array</td>
<td>Return only trigger prototypes that belong to the given templates.</td>
</tr>
<tr>
<td>triggerids</td>
<td>string/array</td>
<td>Return only trigger prototypes with the given IDs.</td>
</tr>
<tr>
<td>expandExpression</td>
<td>flag</td>
<td>Expand functions and macros in the trigger expression.</td>
</tr>
<tr>
<td>selectDependencies</td>
<td>query</td>
<td>Return trigger prototypes and triggers that the trigger prototype depends on in the dependencies property.</td>
</tr>
<tr>
<td>selectDiscoveryRule</td>
<td>query</td>
<td>Return the LLD rule that the trigger prototype belongs to.</td>
</tr>
<tr>
<td>selectFunctions</td>
<td>query</td>
<td>Return functions used in the trigger prototype in the functions property.</td>
</tr>
</tbody>
</table>

The function objects represent the functions used in the trigger expression and has the following properties:

- `functionid` - (string) ID of the function;
- `itemid` - (string) ID of the item used in the function;
- `function` - (string) name of the function;
- `parameter` - (string) parameter passed to the function. Query parameter is replaced by `$` symbol in returned string.

| selectHostGroups       | query     | Return the host groups that the trigger prototype belongs to in the host groups property. |
| selectHosts            | query     | Return the hosts that the trigger prototype belongs to in the hosts property.         |
| selectItems            | query     | Return items and item prototypes used the trigger prototype in the items property.   |
| selectTags             | query     | Return the trigger prototype tags in tags property.                                 |
| selectTemplateGroups   | query     | Return the template groups that the trigger prototype belongs to in the template groups property. |
| filter                 | object    | Return only those results that exactly match the given filter.                    |

Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against.

Supports additional filters:
- `host` - technical name of the host that the trigger prototype belongs to;
- `hostid` - ID of the host that the trigger prototype belongs to.

| limitSelects           | integer   | Limits the number of records returned by subselects.                          |
| sortfield              | string/array | Sort the result by the given properties.                                    |
|                       |           | Possible values are: `triggerid`, `description`, `status` and `priority`.     |
|                       |           | These parameters being common for all get methods are described in detail in the reference commentary. |

| editable               | boolean   |                                                                    |
| excludeSearch         | boolean   |                                                                    |
| limit                 | integer   |                                                                    |
| output                | query     |                                                                    |
| preservekeys          | boolean   |                                                                    |
| search                | object    |                                                                    |
| searchByAny           | boolean   |                                                                    |
| searchWildcardsEnabled | boolean       |                                                                    |
| sortOrder             | string/array |                                                                    |
| startSearch           | boolean   |                                                                    |
| selectGroups          | query     |                                                                    |

This parameter is deprecated, please use `selectHostGroups` or `selectTemplateGroups` instead.

Return the host groups and template groups that the trigger prototype belongs to in the groups property.
(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieve trigger prototypes from an LLD rule

Retrieve all trigger prototypes and their functions from an LLD rule.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.get",
    "params": {
        "output": "extend",
        "selectFunctions": "extend",
        "discoveryids": "22450"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "13272",
            "expression": "{12598}<20",
            "description": "Free inodes is less than 20% on volume {#FSNAME}"
        },
        {
            "triggerid": "13266",
            "expression": "{13500}<20",
            "description": "Free disk space is less than 20% on volume {#FSNAME}"
        }
    ]
}
```
Retrieving a specific trigger prototype with tags

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.get",
    "params": {
        "output": [
            "triggerid",
            "description"
        ],
        "selectTags": "extend",
        "triggerids": [
            "17373"
        ],
        "params": [ {
            "output": [ "triggerid", "description" ],
            "selectTags": "extend",
            "triggerids": [ "17373" ]
        }, {
            "auth": "038e1d7b1735c6a5436ee9eae095879e",
            "id": 1
        }]
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [ {
        "triggerid": "17373",
        "description": "Free disk space is less than 20% on volume \\
{#FSNAME}\",
        "tags": [ {
            "tag": "volume",
            "value": "\{#FSNAME\}"
        }, {
            "tag": "type",
            "value": "\{#FSTYPE\}"
        }]
    }],
    "id": 1
}
```
triggerprototype.update

Description

object triggerprototype.update(object/array triggerPrototypes)

This method allows to update existing trigger prototypes.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) **Trigger prototype properties** to be updated.

The triggerid property must be defined for each trigger prototype, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard trigger prototype properties the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependencies</td>
<td>array</td>
<td>Triggers and trigger prototypes that the trigger prototype is dependent on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tags</td>
<td>array</td>
<td>The triggers must have the triggerid property defined. Trigger prototype tags.</td>
</tr>
</tbody>
</table>

The trigger expression has to be given in its expanded form and must contain at least one item prototype.

Return values

(object) Returns an object containing the IDs of the updated trigger prototypes under the triggerids property.

Examples

Enabling a trigger prototype

Enable a trigger prototype, that is, set its status to 0.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.update",
    "params": {
        "triggerid": "13938",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:
Replacing trigger prototype tags

Replace tags for one trigger prototype.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "triggerprototype.update",
   "params": {
      "triggerid": "17373",
      "tags": [
         {
            "tag": "volume",
            "value": "{#FSNAME}"
         },
         {
            "tag": "type",
            "value": "{#FSTYPE}"
         }
      ]
   },
   "auth": "038e1d7b1735c6a5436ee9eae096879e",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "triggerids": ["17373"
   ],
   "id": 1
}
```

Source

CTriggerPrototype::update() in ui/include/classes/api/services/CTriggerPrototype.php.

**User**

This class is designed to work with users.

Object references:

- User

Available methods:

- user.checkauthentication - checking and prolonging user sessions
- user.create - creating new users
- user.delete - deleting users
- `user.get` - retrieving users
- `user.login` - logging in to the API
- `user.logout` - logging out of the API
- `user.unblock` - unblocking users
- `user.update` - updating users

> **User object**

The following objects are directly related to the user API.

**User**

The user object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userid</td>
<td>string</td>
<td>(readonly) ID of the user.</td>
</tr>
<tr>
<td><code>username</code></td>
<td>string</td>
<td>User name.</td>
</tr>
<tr>
<td>attempt_clock</td>
<td>timestamp</td>
<td>(readonly) Time of the last unsuccessful login attempt.</td>
</tr>
<tr>
<td>attempt_failed</td>
<td>integer</td>
<td>(readonly) Recent failed login attempt count.</td>
</tr>
<tr>
<td>attempt_ip</td>
<td>string</td>
<td>(readonly) IP address from where the last unsuccessful login attempt came from.</td>
</tr>
<tr>
<td>autologin</td>
<td>integer</td>
<td>Whether to enable auto-login.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: 0 - (default) auto-login disabled; 1 - auto-login enabled.</td>
</tr>
<tr>
<td>autologout</td>
<td>string</td>
<td>User session life time. Accepts seconds and time unit with suffix. If set to 0s, the session will never expire.</td>
</tr>
<tr>
<td>lang</td>
<td>string</td>
<td>Language code of the user’s language, for example, en_GB.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the user.</td>
</tr>
<tr>
<td>refresh</td>
<td>string</td>
<td>Automatic refresh period. Accepts seconds and time unit with suffix.</td>
</tr>
<tr>
<td>rows_per_page</td>
<td>integer</td>
<td>Amount of object rows to show per page.</td>
</tr>
<tr>
<td>surname</td>
<td>string</td>
<td>Surname of the user.</td>
</tr>
<tr>
<td>theme</td>
<td>string</td>
<td>User’s theme.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values: default - (default) system default; blue-theme - Blue; dark-theme - Dark.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>URL of the page to redirect the user to after logging in.</td>
</tr>
<tr>
<td>timezone</td>
<td>string</td>
<td>User’s time zone, for example, Europe/London, UTC.</td>
</tr>
<tr>
<td>roleid</td>
<td>string</td>
<td>Role ID of the user.</td>
</tr>
</tbody>
</table>

**Media**

The media object has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mediatypeid</td>
<td>string</td>
<td>ID of the media type used by the media.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sendto</td>
<td>string/array</td>
<td>Address, user name or other identifier of the recipient.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IftypeofMediatype</td>
<td></td>
<td></td>
</tr>
<tr>
<td>is-mail</td>
<td></td>
<td></td>
</tr>
<tr>
<td>values are</td>
<td></td>
<td></td>
</tr>
<tr>
<td>represented as</td>
<td></td>
<td></td>
</tr>
<tr>
<td>array. For</td>
<td></td>
<td></td>
</tr>
<tr>
<td>other types of</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mediatypes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>value is</td>
<td></td>
<td></td>
</tr>
<tr>
<td>represented as</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a string.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>active</td>
<td>integer</td>
<td>Whether the media is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>severity</td>
<td>integer</td>
<td>Trigger severities to send notifications about.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>period</td>
<td>string</td>
<td>Time when the notifications can be sent as a time period</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or user macros separated by a semicolon.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>user.checkAuthentication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>object user.checkAuthentication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>checks and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>prolongs user</td>
<td></td>
<td></td>
</tr>
<tr>
<td>session.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>user.checkAuthentication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>method prolongs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>user session</td>
<td></td>
<td></td>
</tr>
<tr>
<td>by default.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>accepts the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>following</td>
<td></td>
<td></td>
</tr>
<tr>
<td>parameters.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>extend</td>
<td>boolean</td>
<td>Default value: &quot;true&quot;. Setting it's value to &quot;false&quot; allows to check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>session without extending it's lifetime. Supported since Zabbix 4.0.</td>
</tr>
<tr>
<td>sessionid</td>
<td>string</td>
<td>User session id.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing information about user.

Examples

Request:
```
{
  "jsonrpc": "2.0",
  "method": "user.checkAuthentication",
  "params": {
    "sessionid": "673b8ba11562a35da902c66cf5c23fa2"
  },
  "id": 1
}
Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userid": "1",
        "username": "Admin",
        "name": "Zabbix",
        "surname": "Administrator",
        "url": "",
        "autologin": "1",
        "autologout": "0",
        "lang": "ru_RU",
        "refresh": "0",
        "theme": "default",
        "attempt_failed": "0",
        "attempt_ip": "127.0.0.1",
        "attempt_clock": "1355919038",
        "rows_per_page": "50",
        "timezone": "Europe/Riga",
        "roleid": "3",
        "type": 3,
        "sessionid": "673b8ba11562a35da902c66cf5c23fa2",
        "debug_mode": 0,
        "userid": "127.0.0.1",
        "gui_access": 0,
        "userdirectoryid": 0
    },
    "id": 1
}
```

Response is similar to User.login call response with "userData" parameter set to true (the difference is that user data is retrieved by session id and not by username / password).

Source
CUser::checkAuthentication() in ui/include/classes/api/services/CUser.php.

**user.create**

**Description**

object user.create(object/array users)

This method allows to create new users.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

The strength of user password is validated according the password policy rules defined by Authentication API. See Authentication API for more information.

**Parameters**

(object/array) Users to create.

Additionally to the standard user properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>passwd</td>
<td>string</td>
<td>User’s password.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>usrgrps</td>
<td>array</td>
<td>Can be omitted if user is added only to groups that have LDAP access.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td>User groups to add the user to.</td>
</tr>
<tr>
<td>medias</td>
<td>array</td>
<td>The user groups must have the usgrpid property defined.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User media to be created.</td>
</tr>
</tbody>
</table>
Return values

(object) Returns an object containing the IDs of the created users under the userids property. The order of the returned IDs matches the order of the passed users.

Examples

Creating a user

Create a new user, add him to a user group and create a new media for him.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "user.create",
   "params": {
      "username": "John",
      "passwd": "Doe123",
      "roleid": "5",
      "usrgrps": [
         { "usrgrpid": "7" }
      ],
      "medias": [
         { "mediatypeid": "1",
            "sendto": [ "support@company.com" ],
            "active": 0,
            "severity": 63,
            "period": "1-7,00:00-24:00"
         }
      ],
      "auth": "038e1d7b1735c6a5436ee9eae095879e",
      "id": 1
   }
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "userids": [ "12" ],
      "id": 1
   }
}
```

See also

- Authentication
- Media
- User group
- Role

Source

CUser::create() in ui/include/classes/api/services/CUser.php.

**user.delete**

Description

object user.delete(array users)
This method allows to delete users.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(array) IDs of users to delete.

Return values
(object) Returns an object containing the IDs of the deleted users under the userid property.

Examples
Deleting multiple users
Delete two users.

Request:
```
{
  "jsonrpc": "2.0",
  "method": "user.delete",
  "params": [
    "1",
    "5"
  ],
  "auth": "3a57200802b24cda67c4e4010b50c065",
  "id": 1
}
```

Response:
```
{
  "jsonrpc": "2.0",
  "result": {
    "userid": [
      "1",
      "5"
    ]
  },
  "id": 1
}
```

Source
CUser::delete() in ui/include/classes/api/services/CUser.php.

user.get

Description
integer/array user.get(object parameters)

The method allows to retrieve users according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mediaids</td>
<td>string/array</td>
<td>Return only users that use the given media.</td>
</tr>
<tr>
<td>mediatypeids</td>
<td>string/array</td>
<td>Return only users that use the given media types.</td>
</tr>
<tr>
<td>userids</td>
<td>string/array</td>
<td>Return only users with the given IDs.</td>
</tr>
<tr>
<td>usrgrps</td>
<td>string/array</td>
<td>Return only users that belong to the given user groups.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>getAccess</td>
<td>flag</td>
<td>Adds additional information about user permissions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Adds the following properties for each user:</td>
</tr>
<tr>
<td>gui_access</td>
<td>(integer)</td>
<td>user’s frontend authentication method. Refer to the gui_access property of the user group object for a list of possible values.</td>
</tr>
<tr>
<td>debug_mode</td>
<td>(integer)</td>
<td>indicates whether debug is enabled for the user. Possible values: 0 - debug disabled, 1 - debug enabled.</td>
</tr>
<tr>
<td>users_status</td>
<td>(integer)</td>
<td>indicates whether the user is disabled. Possible values: 0 - user enabled, 1 - user disabled.</td>
</tr>
</tbody>
</table>

| selectMedias     | query | Return media used by the user in the medias property.                        |
| selectMediatypes | query | Return media types used by the user in the mediatypes property.              |
| selectUsrgrps    | query | Return user groups that the user belongs to in the usrgrps property.         |
| selectRole       | query | Return user role in the role property.                                      |
| sortfield        | string/array | Sort the result by the given properties. Possible values are: userid and username. |
| countOutput      | boolean | These parameters being common for all get methods are described in detail in the reference commentary. |

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving users

Retrieve all of the configured users.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "user.get",
    "params": {
        "output": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae096879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "userid": "1",
            "username": "Admin",
            "name": "Zabbix"
        }
    ]
}
```
Retrieving user data

Retrieve data of a user with ID "12".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "user.get",
    "params": {
```
Response:
{
"jsonrpc": "2.0",
"result": [
{
"userid": "12",
"username": "John",
"role": {
"roleid": "5",
"name": "Operator",
"type": "1",
"readonly": "0"
}
}
],
"id": 1
}

See also
- Media
- Media type
- User group
- Role

Source
CUser::get() in ui/include/classes/api/services/CUser.php.

user.login

Description
string/object user.login(object parameters)

This method allows to log in to the API and generate an authentication token.

When using this method, you also need to do user.logout to prevent the generation of a large number of open session records.

This method is only available to unauthenticated users and must be called without the auth parameter in the JSON-RPC request.

Parameters
(object) Parameters containing the user name and password.

The method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>string</td>
<td>User password.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>username</td>
<td>string</td>
<td>User name.</td>
</tr>
<tr>
<td>(required)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

userData flag Return information about the authenticated user.

Return values
(string/object) If the userData parameter is used, returns an object containing information about the authenticated user.

Additionally to the standard user properties, the following information is returned:
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>debug_mode</td>
<td>boolean</td>
<td>Whether debug mode is enabled for the user.</td>
</tr>
<tr>
<td>gui_access</td>
<td>integer</td>
<td>User’s authentication method to the frontend.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to the gui_access property of the user group object for a list of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>possible values.</td>
</tr>
<tr>
<td>sessionid</td>
<td>string</td>
<td>Authentication token, which must be used in the following API requests.</td>
</tr>
<tr>
<td>userip</td>
<td>string</td>
<td>IP address of the user.</td>
</tr>
</tbody>
</table>

If a user has been successfully authenticated after one or more failed attempts, the method will return the current values for the attempt_clock, attempt_failed and attempt_ip properties and then reset them.

If the userData parameter is not used, the method returns an authentication token.

The generated authentication token should be remembered and used in the auth parameter of the following JSON-RPC requests. It is also required when using HTTP authentication.

Examples

Authenticating a user

Authenticate a user.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "user.login",
   "params": {
      "username": "Admin",
      "password": "zabbix"
   },
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": "0424bd59b807674191e7d77572075f33",
   "id": 1
}
```

Requesting authenticated user’s information

Authenticate and return additional information about the user.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "user.login",
   "params": {
      "username": "Admin",
      "password": "zabbix",
      "userData": true
   },
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "userid": "1",
      "username": "Admin",
      "name": "Zabbix",
      "surname": "Administrator",
      1302
   }
}
```
See also

- user.logout

Source

CUser::login() in ui/include/classes/api/services/CUser.php.

**user.logout**

**Description**

```php
string/object user.logout(array)
```

This method allows to log out of the API and invalidates the current authentication token. This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

- **(array)** The method accepts an empty array.

**Return values**

- **(boolean)** Returns true if the user has been logged out successfully.

**Examples**

**Logging out**

**Log out from the API.**

**Request:**

```json
{
   "jsonrpc": "2.0",
   "method": "user.logout",
   "params": [],
   "id": 1,
   "auth": "16a46baf181ef9602e1687f3110abf8a"
}
```

**Response:**

```json
{
   "jsonrpc": "2.0",
   "result": true
}
```
See also

- user.login

Source

CUser::login() in ui/include/classes/api/services/CUser.php.

### user.unblock

**Description**

**object** user.unblock(array userids)

This method allows to unblock users.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See **User roles** for more information.

**Parameters**

- (array) IDs of users to unblock.

**Return values**

- (object) Returns an object containing the IDs of the unblocked users under the **userids** property.

**Examples**

Unblocking multiple users

Unblock two users.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "user.unblock",
    "params": [
        "1",
        "5"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

**Response:**

```json
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "1",
            "5"
        ]
    },
    "id": 1
}
```

Source

CUser::unblock() in ui/include/classes/api/services/CUser.php.

### user.update

**Description**

**object** user.update(object/array users)

**1304**

This method allows to update existing users.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

The strength of user password is validated according the password policy rules defined by Authentication API. See Authentication API for more information.

Parameters

(object/array) User properties to be updated.

The userid property must be defined for each user, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard user properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>passwd</td>
<td>string</td>
<td>User’s password. Can be empty string if user belongs to or is moved only to groups that have LDAP access.</td>
</tr>
<tr>
<td>usrgrps</td>
<td>array</td>
<td>User groups to replace existing user groups. The user groups must have the usrgrpid property defined.</td>
</tr>
<tr>
<td>medias</td>
<td>array</td>
<td>User media to replace existing media.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated users under the userids property.

Examples

Renaming a user

Rename a user to John Doe.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "user.update",
    "params": {
        "userid": "1",
        "name": "John",
        "surname": "Doe"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "1"
        ],
        "id": 1
    }
}
```

Changing user role

Change a role of a user.

Request:
```json
{
    "jsonrpc": "2.0",
    "method": "user.update",
    "params": {
        "userid": "12",
        "roleid": "6"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "12"
        ]
    },
    "id": 1
}
```

See also

- Authentication

Source

CUser::update() in ui/include/classes/api/services/CUser.php.

User directory

This class is designed to work with user directories.

Object references:

- User directory

Available methods:

- `userdirectory.create` - create new user directory
- `userdirectory.delete` - delete user directory
- `userdirectory.get` - retrieve user directory
- `userdirectory.update` - update user directory
- `userdirectory.test` - test user directory connection

> User directory object

The following objects are directly related to the userdirectory API.

User directory

The user directory object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userid</td>
<td>string</td>
<td>(readonly) ID of the user directory.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Unique name of the user directory.</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>LDAP server host name, IP or URI. URI should contain schema, host and port.</td>
</tr>
<tr>
<td>port</td>
<td>integer</td>
<td>LDAP server port.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>base_dn</td>
<td>string</td>
<td>LDAP base distinguished name string.</td>
</tr>
<tr>
<td>search_attribute</td>
<td>string</td>
<td>LDAP attribute name to identify user by username in Zabbix database.</td>
</tr>
<tr>
<td>bind_dn</td>
<td>string</td>
<td>LDAP bind distinguished name string. Can be empty for anonymous binding.</td>
</tr>
<tr>
<td>bind_password</td>
<td>string</td>
<td>LDAP bind password. Can be empty for anonymous binding.</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>User directory description.</td>
</tr>
<tr>
<td>search_filter</td>
<td>string</td>
<td>LDAP custom filter string when authenticating user in LDAP.</td>
</tr>
<tr>
<td>start_tls</td>
<td>integer</td>
<td>LDAP startTLS option. It cannot be used with ldaps:// protocol hosts.</td>
</tr>
</tbody>
</table>

**Filter search_filter supported placeholders:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%{attr}</td>
<td>Search attribute name (uid, sAMAccountName).</td>
</tr>
<tr>
<td>%{user}</td>
<td>Username value.</td>
</tr>
</tbody>
</table>

**userdirectory.create**

Description

doctype userdirectory.create(object/array userDirectory)

This method allows to create new user directories.

This method is only available to Super admin user type.

Parameters

(object/array) User directories to create.

The method accepts user directories with the standard user directory properties.

Return values

(object) Returns an object containing the IDs of the created user directories under the userdirectoryids property. The order of the returned IDs matches the order of the passed user directories.

Examples

Creating a user directory

Create a user directory to authenticate users with StartTLS over LDAP.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "userdirectory.create",
    "params": {
        "name": "LDAP API server #1",
        "host": "ldap://local.ldap",
        "port": "389",
        "base_dn": "ou=Users,dc=example,dc=org",
        "bind_dn": "cn=ldap_search,dc=example,dc=org",
        "bind_password": "password"
    }
}
```
"bind_password": "ldapsecretpassword",
"search_attribute": "uid",
"start_tls": "1"
},
"auth": "038e1d7b1735c6a5436ee9eae096879e",
"id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": {
    "userdirectoryids": [
      "2"
    ],
  },
  "id": 1
}

Source
CUserDirectory::create() in ui/include/classes/api/services/CUserDirectory.php.

**userdirectory.delete**

**Description**
object userdirectory.delete(array userDirectoryIds)

This method allows to delete user directories. User directory cannot be deleted when it is directly used for at least one user group. Default LDAP user directory cannot be deleted when authentication.ldap_configured is set to 1 or when there are more user directories left.

This method is only available to Super admin user type.

**Parameters**
(array) IDs of the user directories to delete.

**Return values**
(object) Returns an object containing the IDs of the deleted user directories under the userdirectoryids property.

**Examples**

Deleting multiple user directories

Delete two user directories.

Request:
{
  "jsonrpc": "2.0",
  "method": "userdirectory.delete",
  "params": [
    "2",
    "12"
  ],
  "auth": "3a57200802b24cda67c4e4010b50c065",
  "id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": {
    "userdirectoryids": [
      "2",
      "12"
    ],
    "id": 1
  }
}
Source

CUserDirectory::delete() in ui/include/classes/api/services/CUserDirectory.php.

**userdirectory.get**

**Description**

`integer/array userdirectory.get(object parameters)`

The method allows to retrieve user directories according to the given parameters.

This method is only available to Super admin user types.

**Parameters**

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userdirectoryids</td>
<td>string/array</td>
<td>Return only user directory with the given IDs.</td>
</tr>
<tr>
<td>selectUsrgrps</td>
<td>query</td>
<td>Return a usgrps property with user groups associated with user directory.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Supports count. Sort the result by the given properties.</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td>Return only those results that exactly match the given filter.</td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td>Return results that match the given wildcard search (case-insensitive).</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Return values**

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

**Examples**

**Retrieving user directories**

Retrieve all user directories with additional property with count of user groups where user directory is used.

Request:
See also

- User group

Source

CUserDirectory::get() in ui/include/classes/api/services/CUserDirectory.php.

userdirectory.test

Description

object userdirectory.test(array userDirectory)

This method allows to test user directory connection settings.

This method is only available to Super admin user type.

Parameters

(object) User directory properties.

Since userdirectory.get API does not return bind_password field, userdirectoryid and/or bind_password should be supplied.

Additionally to the standard user directory properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_username</td>
<td>string</td>
<td>Username to test in user directory.</td>
</tr>
<tr>
<td>test_password</td>
<td>string</td>
<td>Username associated password to test in user directory.</td>
</tr>
</tbody>
</table>

Return values

(bool) Returns true on success.
Examples

Test user directory

Test user directory for user "user1".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "userdirectory.test",
    "params": {
        "userdirectoryid": "2",
        "host": "127.0.0.1",
        "port": "3389",
        "base_dn": "ou=Users,dc=example,dc=org",
        "search_attribute": "uid",
        "bind_dn": "cn=ldap_search,dc=example,dc=org",
        "bind_password": "password",
        "test_username": "user1",
        "test_password": "password"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Test user directory

Test non existing user "user2".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "userdirectory.test",
    "params": {
        "userdirectoryid": "2",
        "host": "127.0.0.1",
        "port": "3389",
        "base_dn": "ou=Users,dc=example,dc=org",
        "search_attribute": "uid",
        "bind_dn": "cn=ldap_search,dc=example,dc=org",
        "bind_password": "password",
        "test_username": "user2",
        "test_password": "password"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "error": {
        "code": -32500,
        "message": "Application error."
    },
    "data": "Incorrect user name or password or account is temporarily blocked."
},
    "id": 1
}
```
User directory update

Description

object userdirectory.update(object/array userDirectory)

This method allows to update existing user directories.
This method is only available to Super admin user type.

Parameters

(object/array) User directory properties to be updated.

The userdirectoryid property must be defined for each user directory, all other properties are optional.
Only the passed properties will be updated, all others will remain unchanged.

Return values

(object) Returns an object containing the IDs of the updated user directories under the userdirectoryids property.

Examples

Update bind password for user directory

Set new bind password for a user directory.

Request:

```json
{
  "jsonrpc": "2.0",
  "method": "userdirectory.update",
  "params": {
    "userdirectory": "2",
    "bind_password": "newldappassword"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```json
{
  "jsonrpc": "2.0",
  "result": {
    "userdirectoryids": [2]
  },
  "id": 1
}
```

User group

This class is designed to work with user groups.

Object references:

- User group

Available methods:
- **usergroup.create** - creating new user groups
- **usergroup.delete** - deleting user groups
- **usergroup.get** - retrieving user groups
- **usergroup.update** - updating user groups

> **User group object**

The following objects are directly related to the `usergroup` API.

**User group**

The user group object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>usrgrpid</code></td>
<td>string</td>
<td>(readonly) ID of the user group.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>string</td>
<td>Name of the user group.</td>
</tr>
<tr>
<td><code>debug_mode</code></td>
<td>integer</td>
<td>Whether debug mode is enabled or disabled. Possible values are: 0 - (default) disabled; 1 - enabled.</td>
</tr>
<tr>
<td><code>gui_access</code></td>
<td>integer</td>
<td>Frontend authentication method of the users in the group. Possible values: 0 - (default) use the system default authentication method; 1 - use internal authentication; 2 - use LDAP authentication; 3 - disable access to the frontend.</td>
</tr>
<tr>
<td><code>users_status</code></td>
<td>integer</td>
<td>Whether the user group is enabled or disabled. Possible values are: 0 - (default) enabled; 1 - disabled.</td>
</tr>
<tr>
<td><code>userdirectoryid</code></td>
<td>string</td>
<td>Authentication user directory when <code>gui_access</code> set to LDAP or System default.</td>
</tr>
</tbody>
</table>

**Permission**

The permission object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>id</code></td>
<td>string</td>
<td>ID of the host group to add permission to.</td>
</tr>
<tr>
<td><code>permission</code></td>
<td>integer</td>
<td>Access level to the host group. Possible values: 0 - access denied; 2 - read-only access; 3 - read-write access.</td>
</tr>
</tbody>
</table>

**Tag based permission**

The tag based permission object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>groupid</code></td>
<td>string</td>
<td>ID of the host group to add permission to.</td>
</tr>
<tr>
<td><code>tag</code></td>
<td>string</td>
<td>Tag name.</td>
</tr>
<tr>
<td><code>value</code></td>
<td>string</td>
<td>Tag value.</td>
</tr>
</tbody>
</table>
**usergroup.create**

**Description**

object usergroup.create(object/array userGroups)

This method allows to create new user groups.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

**Parameters**

(object/array) User groups to create.

Additionally to the standard user group properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rights</td>
<td>object/array</td>
<td>Permissions to assign to the group</td>
</tr>
<tr>
<td>tag_filters</td>
<td>array</td>
<td>Tag based permissions to assign to the group</td>
</tr>
<tr>
<td>users</td>
<td>object/array</td>
<td>Users to add to the user group. The user must have the userid property defined.</td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns an object containing the IDs of the created user groups under the usrgrpid property. The order of the returned IDs matches the order of the passed user groups.

**Examples**

Create a user group, which denies access to host group “2”, and add a user to it.

**Request:**

```json
{
   "jsonrpc": "2.0",
   "method": "usergroup.create",
   "params": {
      "name": "Operation managers",
      "rights": {
         "permission": 0,
         "id": "2"
      },
      "users": [
         {"userid": "12"}
      ]
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```

**Response:**

```json
{
   "jsonrpc": "2.0",
   "result": {
      "usrgrpid": [
         "20"
      ]
   },
   "id": 1
}
```

See also

- Permission
usergroup.delete

Description

object usergroup.delete(array userGroupIds)

This method allows to delete user groups.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the user groups to delete.

Return values

(object) Returns an object containing the IDs of the deleted user groups under the usrgpids property.

Examples

Deleting multiple user groups

Delete two user groups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.delete",
    "params": [
        "20",
        "21"
    ],
    "auth": "3a57200802b4d67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "20",
            "21"
        ],
        "id": 1
    }
}
```

usergroup.get

Description

integer/array usergroup.get(object parameters)

The method allows to retrieve user groups according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>integer</td>
<td>Return only user groups with the given status.</td>
</tr>
<tr>
<td>userids</td>
<td>string/array</td>
<td>Return only user groups that contain the given users.</td>
</tr>
<tr>
<td>usrgrpids</td>
<td>string/array</td>
<td>Return only user groups with the given IDs.</td>
</tr>
<tr>
<td>selectTagFilters</td>
<td>query</td>
<td>Return user group tag based permissions in the <code>tag_filters</code> property.</td>
</tr>
</tbody>
</table>

It has the following properties:
- **groupid** - (string) ID of the host group;
- **tag** - (string) tag name;
- **value** - (string) tag value.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>selectUsers</td>
<td>query</td>
<td>Return the users from the user group in the <code>users</code> property.</td>
</tr>
<tr>
<td>selectHostGroupRights</td>
<td>query</td>
<td>Return user group host group rights in the <code>host group rights</code> property.</td>
</tr>
</tbody>
</table>

It has the following properties:
- **permission** - (integer) access level to the host group;
- **id** - (string) ID of the host group.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>selectTemplateGroupRights</td>
<td>query</td>
<td>Return user group template group rights in the <code>template group rights</code> property.</td>
</tr>
</tbody>
</table>

It has the following properties:
- **permission** - (integer) access level to the template group;
- **id** - (string) ID of the template group.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>limitSelects</td>
<td>integer</td>
<td>Limits the number of records returned by subselects.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Possible values are: <code>usrgrpId</code>, <code>name</code>.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>selectRights</td>
<td>query</td>
<td></td>
</tr>
</tbody>
</table>

This parameter is deprecated, please use `selectHostGroupRights` or `selectTemplateGroupRights` instead.

Return user group rights in the `rights` property.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>limitSelects</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>selectRights</td>
<td>query</td>
<td></td>
</tr>
</tbody>
</table>

Refer to the `user group page` for a list of access levels to host groups.

Return values

(integer/array) Returns either:
• an array of objects;
• the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving enabled user groups

Retrieve all enabled user groups.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "usergroup.get",
    "params": {
        "output": "extend",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": [
        {
            "usrgrpid": "7",
            "name": "Zabbix administrators",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "1",
            "userdirectoryid": "0"
        },
        {
            "usrgrpid": "8",
            "name": "Guests",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0",
            "userdirectoryid": "0"
        },
        {
            "usrgrpid": "11",
            "name": "Enabled debug mode",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "1",
            "userdirectoryid": "0"
        },
        {
            "usrgrpid": "12",
            "name": "No access to the frontend",
            "gui_access": "2",
            "users_status": "0",
            "debug_mode": "0",
            "userdirectoryid": "0"
        },
        {
            "usrgrpid": "14",
            "name": "Read only",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0",
            "userdirectoryid": "0"
        }
    ]
}
```
See also

- User

Source

CUserGroup::get() in ui/include/classes/api/services/CUserGroup.php.

**usergroup.update**

Description

**object usergroup.update(object/array userGroups)**

This method allows to update existing user groups.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See **User roles** for more information.

Parameters

(object/array) User group properties to be updated.

The **usrgrpid** property must be defined for each user group, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the **standard user group properties**, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rights</td>
<td>object/array</td>
<td>Permissions to replace the current permissions assigned to the user group.</td>
</tr>
<tr>
<td>tag_filters</td>
<td>array</td>
<td>Tag based permissions to assign to the group.</td>
</tr>
<tr>
<td>users</td>
<td>object/array</td>
<td>Users to add to the user group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The user must have the <strong>userid</strong> property defined.</td>
</tr>
</tbody>
</table>

Return values

(object) Returns an object containing the IDs of the updated user groups under the **usrgrpids** property.

Examples

Disabling a user group

Disable a user group.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "usergroup.update",
    "params": {
        "usrgrpid": "17",
        "users_status": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e"
}
```
Response:
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "17"
        ],
        "id": 1
    },
    "id": 1
}

See also
- Permission

Source
CUserGroup::update() in ui/include/classes/api/services/CUserGroup.php.

User macro

Host macro is deprecated and will be removed in upcoming versions.
This class is designed to work with host and global macros.

Object references:
- Global macro
- Host macro

Available methods:
- usermacro.create - creating new host macros
- usermacro.createglobal - creating new global macros
- usermacro.delete - deleting host macros
- usermacro.deleteglobal - deleting global macros
- usermacro.get - retrieving host and global macros
- usermacro.update - updating host macros
- usermacro.updateglobal - updating global macros

> User macro object

The following objects are directly related to the usermacro API.

Global macro

The global macro object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>globalmacroid</td>
<td>string</td>
<td>(readonly) ID of the global macro.</td>
</tr>
<tr>
<td>macro</td>
<td>string</td>
<td>(required) Macro string.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>(required) Value of the macro.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>(required) Type of macro.</td>
</tr>
</tbody>
</table>

Possible values:
- 0 - (default) Text macro;
- 1 - Secret macro;
- 2 - Vault secret.
Host macro

This functionality is deprecated and will be removed in upcoming versions.

The host macro object defines a macro available on a host, host prototype or template. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>string</td>
<td>Description of the macro.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostmacroid</td>
<td>string</td>
<td>(readonly) ID of the host macro.</td>
</tr>
<tr>
<td>hostid</td>
<td>string</td>
<td>ID of the host that the macro belongs to.</td>
</tr>
<tr>
<td>macro</td>
<td>string</td>
<td>Macro string.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Value of the macro.</td>
</tr>
<tr>
<td>type</td>
<td>integer</td>
<td>Type of macro.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Text macro;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Secret macro;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - Vault secret.</td>
</tr>
<tr>
<td>description</td>
<td>string</td>
<td>Description of the macro.</td>
</tr>
<tr>
<td>automatic</td>
<td>integer</td>
<td>Defines whether the macro is controlled by discovery rule.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) Macro is managed by user;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - Macro is managed by discovery rule.</td>
</tr>
</tbody>
</table>

User is not allowed to create automatic macro. To update automatic macro, it must be converted to manual.

usermacro.create

This functionality is deprecated and will be removed in upcoming versions.

Description

object usermacro.create(object/array hostMacros)

This method allows to create new host macros.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Host macros to create.

The method accepts host macros with the standard host macro properties.

Return values

(object) Returns an object containing the IDs of the created host macros under the hostmacroids property. The order of the returned IDs matches the order of the passed host macros.

Examples

Creating a host macro

Create a host macro "{$SNMP_COMMUNITY}" with the value "public" on host "10198".

Request:
{  "jsonrpc": "2.0",  "method": "usermacro.create",  "params": {    "hostid": "10198",    "macro": "{$SNMP_COMMUNITY}",    "value": "public"  },  "auth": "038e1d7b1735c6a5436ee9eae095879e",  "id": 1}
}

Response:
{
  "jsonrpc": "2.0",  "result": {    "hostmacroids": [11],  },  "id": 1}
}

Source
CUserMacro::create() in ui/include/classes/api/services/CUserMacro.php.

usermacro.createglobal

Description

object usermacro.createglobal(object/array globalMacros)

This method allows to create new global macros.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Global macros to create.

The method accepts global macros with the standard global macro properties.

Return values

(object) Returns an object containing the IDs of the created global macros under the globalmacroids property. The order of the returned IDs matches the order of the passed global macros.

Examples

Creating a global macro

Create a global macro "{$SNMP_COMMUNITY}" with value "public".

Request:
{
  "jsonrpc": "2.0",  "method": "usermacro.createglobal",  "params": {    "macro": "{$SNMP_COMMUNITY}",    "value": "public"  },  "auth": "038e1d7b1735c6a5436ee9eae095879e",  "id": 1
}

Response:
null

usermacro.delete

This functionality is deprecated and will be removed in upcoming versions.

Description

object usermacro.delete(array hostMacroIds)

This method allows to delete host macros.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the host macros to delete.

Return values

(object) Returns an object containing the IDs of the deleted host macros under the hostmacroids property.

Examples

Deleting multiple host macros

Delete two host macros.

Request:

{  "jsonrpc": "2.0",  "method": "usermacro.delete",  "params": [    "32",    "11"  ],  "auth": "3a57200802b24cda67c4e4010b50c065",  "id": 1}  

Response:

{  "jsonrpc": "2.0",  "result": {    "hostmacroids": [      "32",      "11"    ]  },  "id": 1}  

Source

CUserMacro::createGlobal() in ui/include/classes/api/services/CUserMacro.php.

CUserMacro::delete() in ui/include/classes/api/services/CUserMacro.php.
usermacro.deleteglobal

Description

object usermacro.deleteglobal(array globalMacroIds)

This method allows to delete global macros.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the global macros to delete.

Return values

(object) Returns an object containing the IDs of the deleted global macros under the globalmacroids property.

Examples

Deleting multiple global macros

Delete two global macros.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.deleteglobal",
    "params": [
        "32",
        "11"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "globalmacroids": [
            "32",
            "11"
        ]
    },
    "id": 1
}
```

Source

CUserMacro::deleteGlobal() in ui/include/classes/api/services/CUserMacro.php.

usermacro.get

This functionality is deprecated and will be removed in upcoming versions.

Description

integer/array usermacro.get(object parameters)

The method allows to retrieve host and global macros according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>globalmacro</td>
<td>flag</td>
<td>Return global macros instead of host macros.</td>
</tr>
<tr>
<td>globalmacroids</td>
<td>string/array</td>
<td>Return only global macros with the given IDs.</td>
</tr>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only host macros that belong to hosts or templates from the given host groups or template groups.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only macros that belong to the given hosts or templates.</td>
</tr>
<tr>
<td>hostmacroids</td>
<td>string/array</td>
<td>Return only host macros with the given IDs.</td>
</tr>
<tr>
<td>inherited</td>
<td>boolean</td>
<td>If set to true return only host prototype user macros inherited from a template.</td>
</tr>
<tr>
<td>selectHostGroups</td>
<td>query</td>
<td>Return host groups that the host macro belongs to in the host groups property.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return hosts that the host macro belongs to in the hosts property.</td>
</tr>
<tr>
<td>selectTemplateGroups</td>
<td>query</td>
<td>Return template groups that the template macro belongs to in the template groups property.</td>
</tr>
<tr>
<td>selectTemplates</td>
<td>query</td>
<td>Return templates that the host macro belongs to in the templates property.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Possible value: macro. These parameters being common for all get methods are described in detail in the reference commentary page.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td>This parameter is deprecated, please use selectHostGroups or selectTemplateGroups instead. Return host groups and template groups that the host macro belongs to in the groups property.</td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>selectGroups</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>(deprecated)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving host macros for a host

Retrieve all host macros defined for host "10198".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "usermacro.get",
    "params": {
        "hostids": ["10198"]
    }
}
```
"params": {
  "output": "extend",
  "hostids": "10198"
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": [
    {
      "hostmacroid": "9",
      "hostid": "10198",
      "macro": "{$INTERFACE}",
      "value": "eth0",
      "description": "",
      "type": "0",
      "automatic": "0"
    },
    {
      "hostmacroid": "11",
      "hostid": "10198",
      "macro": "{$SNMP_COMMUNITY}",
      "value": "public",
      "description": "",
      "type": "0",
      "automatic": "0"
    }
  ],
  "id": 1
}

Retrieving global macros

Retrieve all global macros.

Request:
{
  "jsonrpc": "2.0",
  "method": "usermacro.get",
  "params": {
    "output": "extend",
    "globalmacro": true
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": [
    {
      "globalmacroid": "6",
      "macro": "{$SNMP_COMMUNITY}",
      "value": "public",
      "description": "",
      "type": "0"
    }
  ],
  "id": 1
}
usermacro.update

This functionality is deprecated and will be removed in upcoming versions.

Description

object usermacro.update(object/array hostMacros)

This method allows to update existing host macros.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Host macro properties to be updated.

The hostmacroid property must be defined for each host macro, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Return values

(object) Returns an object containing the IDs of the updated host macros under the hostmacroids property.

Examples

Change the value of a host macro

Change the value of a host macro to “public”.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "usermacro.update",
   "params": {
      "hostmacroid": "1",
      "value": "public"
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "hostmacroids": ["1"],
   },
   "id": 1
}
```

Change macro value that was created by discovery rule

Convert discovery rule created “automatic” macro to “manual” and change its value to “new-value”.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "usermacro.update",
   "params": {
      "hostmacroid": "1",
      "value": "new-value"
   },
   "auth": "038e1d7b1735c6a5436ee9eae095879e",
   "id": 1
}
```
"value": "new-value",
"automatic": "0"
},
"auth": "038e1d7b1735c6a5436ee9eae096879e",
"id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": {
    "hostmacroids": [
      "1"
    ],
  },
  "id": 1
}

Source
CUserMacro::update() in ui/include/classes/api/services/CUserMacro.php.

usermacro.updateglobal

Description
object usermacro.updateglobal(object/array globalMacros)

This method allows to update existing global macros.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
(object/array) Global macro properties to be updated.

The globalmacroid property must be defined for each global macro, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Return values
(object) Returns an object containing the IDs of the updated global macros under the globalmacroids property.

Examples
Changing the value of a global macro

Change the value of a global macro to "public".

Request:
{
  "jsonrpc": "2.0",
  "method": "usermacro.updateglobal",
  "params": {
    "globalmacroid": "1",
    "value": "public"
  },
  "auth": "038e1d7b1735c6a5436ee9eae096879e",
  "id": 1
}

Response:
{
  "jsonrpc": "2.0",
  "result": {
    "globalmacroids": [
      "1"
    ]
  }
}
Source
CUserMacro::updateGlobal() in ui/include/classes/api/services/CUserMacro.php.

Value map

This class is designed to work with value maps.

Object references:

- Value map

Available methods:

- valuemap.create - creating new value maps
- valuemap.delete - deleting value maps
- valuemap.get - retrieving value maps
- valuemap.update - updating value maps

> Value map object

The following objects are directly related to the valuemap API.

Value map

The value map object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>valuemapid</td>
<td>string</td>
<td>(readonly) ID of the value map.</td>
</tr>
<tr>
<td>hostid</td>
<td>id</td>
<td>Value map host ID.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the value map.</td>
</tr>
<tr>
<td>mappings</td>
<td>array</td>
<td>Value mappings for current value map. The mapping object is described in detail below.</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Universal unique identifier, used for linking imported value maps to already existing ones. Used only for value maps on templates. Auto-generated, if not given.</td>
</tr>
</tbody>
</table>

For update operations this field is readonly.

Value mappings

The value mappings object defines value mappings of the value map. It has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>integer</td>
<td>Mapping match type. For type equal 0,1,2,3,4 value field cannot be empty, for type 5 value field should be empty. Possible values: 0 - (default) exact match; 1 - mapping will be applied if value is greater or equal; 2 - mapping will be applied if value is less or equal; 3 - mapping will be applied if value is in range (ranges are inclusive), allow to define multiple ranges separated by comma character; 4 - mapping will be applied if value match regular expression; 5 - default value, mapping will be applied if no other match were found.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Is not required for mapping of type &quot;default&quot;.</td>
</tr>
<tr>
<td>newvalue</td>
<td>string</td>
<td>Value to which the original value is mapped to.</td>
</tr>
</tbody>
</table>

1 supported only for items having value type "numeric unsigned", "numeric float".
2 supported only for items having value type "character".

**valuemap.create**

Description

**object valuemap.create(object/array valuemaps)**

This method allows to create new value maps.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See [User roles](https://example.com/userroles) for more information.

Parameters

(object/array) Value maps to create.

The method accepts value maps with the [standard value map properties](https://example.com/value_property).

Return values

(object) Returns an object containing the IDs of the created value maps the valuemapids property. The order of the returned IDs matches the order of the passed value maps.

Examples

Creating a value map

Create one value map with two mappings.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "valuemap.create",
    "params": {
        "hostid": "50009",
        "name": "Service state",
        "mappings": [
            {
                "type": "1",
                "value": "1",
                "newvalue": "Up"
            },
            {
                "type": "5",
                "newvalue": "Down"
            }
        ]
    }
}
```
Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "valuemapids": [
         "1",
         "2"
      ],
   },
   "id": 1
}
```

Source

CValueMap::create() in ui/include/classes/api/services/CValueMap.php.

**valuemap.delete**

Description

object valuemap.delete(array valuemapids)

This method allows to delete value maps.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the value maps to delete.

Return values

(object) Returns an object containing the IDs of the deleted value maps under the valuemapids property.

Examples

Deleting multiple value maps

Delete two value maps.

Request:

```json
{
   "jsonrpc": "2.0",
   "method": "valuemap.delete",
   "params": [
      "1",
      "2"
   ],
   "auth": "57562fd409b3b9a4d916d45207bbcb",
   "id": 1
}
```

Response:

```json
{
   "jsonrpc": "2.0",
   "result": {
      "valuemapids": [
         "1",
         "2"
      ]
   }
}
```
valuemap.get

Description

integer/array valuemap.get(object parameters)

The method allows to retrieve value maps according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>valuemapIds</td>
<td>string/array</td>
<td>Return only value maps with the given IDs.</td>
</tr>
<tr>
<td>selectMappings</td>
<td>query</td>
<td>Return the value mappings for current value map in the mappings property.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Possible values are: valuemapid, name. These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving value maps

Retrieve all configured value maps.

Request:

{    "jsonrpc": "2.0",    "method": "valuemap.get",    "params": {        "output": "extend"    }}
Retrieve one value map with its mappings.

Request:
```
{
  "jsonrpc": "2.0",
  "method": "valuemap.get",
  "params": {
    "output": "extend",
    "selectMappings": "extend",
    "valuemapids": ["4"]
  },
  "auth": "57562fd409b3b3b9a4d916d45207bbcb",
  "id": 1
}
```

Response:
```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "valuemapid": "4",
      "name": "APC Battery Replacement Status"
    },
    {
      "valuemapid": "5",
      "name": "APC Battery Status"
    },
    {
      "valuemapid": "7",
      "name": "Dell Open Manage System Status"
    }
  ],
  "id": 1
}
```
CValueMap::get() in ui/include/classes/api/services/CValueMap.php.

valuemap.update

Description

object valuemap.update(object/array valuemaps)

This method allows to update existing value maps.

This method is only available to Super admin user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) Value map properties to be updated.

The valuemapid property must be defined for each value map, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Return values

(object) Returns an object containing the IDs of the updated value maps under the valuemapids property.

Examples

Changing value map name

Change value map name to "Device status".

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "valuemap.update",
    "params": {
        "valuemapid": "2",
        "name": "Device status"
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb"
}
```
Changing mappings for one value map.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "valuemap.update",
    "params": {
        "valuemapid": "2",
        "mappings": [
            {
                "type": "0",
                "value": "0",
                "newvalue": "Online"
            },
            {
                "type": "0",
                "value": "1",
                "newvalue": "Offline"
            }
        ]
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "valuemaps": [ "2" ]
    },
    "id": 1
}
```

Source
CValueMap::update() in ui/include/classes/api/services/CValueMap.php.

**Web scenario**

This class is designed to work with web scenarios.

Object references:

- Web scenario
- Scenario step
Available methods:

- `httptest.create` - creating new web scenarios
- `httptest.delete` - deleting web scenarios
- `httptest.get` - retrieving web scenarios
- `httptest.update` - updating web scenarios

> Web scenario object

The following objects are directly related to the `webcheck` API.

Web scenario

The web scenario object has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>httptestid</code></td>
<td>string</td>
<td>(readonly) ID of the web scenario.</td>
</tr>
<tr>
<td><code>hostid</code></td>
<td>string</td>
<td>ID of the host that the web scenario belongs to.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>string</td>
<td>Name of the web scenario.</td>
</tr>
<tr>
<td><code>agent</code></td>
<td>string</td>
<td>User agent string that will be used by the web scenario. Default: Zabbix</td>
</tr>
<tr>
<td><code>authentication</code></td>
<td>integer</td>
<td>Authentication method that will be used by the web scenario. Possible values: 0 - (default) none; 1 - basic HTTP authentication; 2 - NTLM authentication.</td>
</tr>
<tr>
<td><code>delay</code></td>
<td>string</td>
<td>Execution interval of the web scenario. Accepts seconds, time unit with suffix and user macro. Default: 1m.</td>
</tr>
<tr>
<td><code>headers</code></td>
<td>array of HTTP fields</td>
<td>HTTP headers that will be sent when performing a request.</td>
</tr>
<tr>
<td><code>http_password</code></td>
<td>string</td>
<td>Password used for basic HTTP or NTLM authentication.</td>
</tr>
<tr>
<td><code>http_proxy</code></td>
<td>string</td>
<td>Proxy that will be used by the web scenario given as  http://[username[:password]@]proxy.example.com[:port].</td>
</tr>
<tr>
<td><code>http_user</code></td>
<td>string</td>
<td>User name used for basic HTTP or NTLM authentication.</td>
</tr>
<tr>
<td><code>nextcheck</code></td>
<td>timestamp</td>
<td>(readonly) Time of the next web scenario execution.</td>
</tr>
<tr>
<td><code>retries</code></td>
<td>integer</td>
<td>Number of times a web scenario will try to execute each step before failing. Default: 1.</td>
</tr>
<tr>
<td><code>ssl_cert_file</code></td>
<td>string</td>
<td>Name of the SSL certificate file used for client authentication (must be in PEM format).</td>
</tr>
<tr>
<td><code>ssl_key_file</code></td>
<td>string</td>
<td>Name of the SSL private key file used for client authentication (must be in PEM format).</td>
</tr>
<tr>
<td><code>ssl_key_password</code></td>
<td>string</td>
<td>SSL private key password.</td>
</tr>
<tr>
<td><code>status</code></td>
<td>integer</td>
<td>Whether the web scenario is enabled. Possible values are: 0 - (default) enabled; 1 - disabled.</td>
</tr>
<tr>
<td><code>templateid</code></td>
<td>string</td>
<td>(readonly) ID of the parent template web scenario.</td>
</tr>
<tr>
<td><code>variables</code></td>
<td>array of HTTP fields</td>
<td>Web scenario variables.</td>
</tr>
<tr>
<td><code>verify_host</code></td>
<td>integer</td>
<td>Whether to verify that the host name specified in the SSL certificate matches the one used in the scenario. Possible values are: 0 - (default) skip host verification; 1 - verify host.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>verify_peer</td>
<td>integer</td>
<td>Whether to verify the SSL certificate of the web server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) skip peer verification;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - verify peer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(readonly on already existing web scenarios)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Global unique identifier, used for linking imported web scenarios to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>already existing ones. Used only for web scenarios on templates.</td>
</tr>
<tr>
<td>uuid</td>
<td>string</td>
<td>Global unique identifier, used for linking imported web scenarios to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>already existing ones. Used only for web scenarios on templates.</td>
</tr>
</tbody>
</table>

### Web scenario tag

The web scenario tag object has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>string</td>
<td>Web scenario tag name.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Web scenario tag value.</td>
</tr>
</tbody>
</table>

### Scenario step

The scenario step object defines a specific web scenario check. It has the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>httpstpid</td>
<td>string</td>
<td>(readonly) ID of the scenario step.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Name of the scenario step.</td>
</tr>
<tr>
<td>no</td>
<td>integer</td>
<td>Sequence number of the step in a web scenario.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>URL to be checked.</td>
</tr>
<tr>
<td>follow_redirects</td>
<td>integer</td>
<td>Whether to follow HTTP redirects.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - don’t follow redirects;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - (default) follow redirects.</td>
</tr>
<tr>
<td>headers</td>
<td>array of HTTP fields</td>
<td>HTTP headers that will be sent when performing a request. Scenario step headers will overwrite headers specified for the web scenario.</td>
</tr>
<tr>
<td>httpstestid</td>
<td>string</td>
<td>(readonly) ID of the web scenario that the step belongs to.</td>
</tr>
<tr>
<td>posts</td>
<td>string</td>
<td>HTTP POST variables as a string (raw post data) or as an array of HTTP fields (form field data).</td>
</tr>
<tr>
<td>required</td>
<td>string</td>
<td>Text that must be present in the response.</td>
</tr>
<tr>
<td>retrieve_mode</td>
<td>integer</td>
<td>Part of the HTTP response that the scenario step must retrieve.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - (default) only body;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - only headers;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - headers and body.</td>
</tr>
<tr>
<td>status_codes</td>
<td>string</td>
<td>Ranges of required HTTP status codes separated by commas.</td>
</tr>
<tr>
<td>timeout</td>
<td>string</td>
<td>Request timeout in seconds. Accepts seconds, time unit with suffix and user macro.</td>
</tr>
<tr>
<td>variables</td>
<td>array of HTTP fields</td>
<td>Scenario step variables.</td>
</tr>
<tr>
<td>query_fields</td>
<td>array of HTTP fields</td>
<td>Query fields - array of HTTP fields that will be added to URL when performing a request</td>
</tr>
</tbody>
</table>

### HTTP field

The HTTP field object defines a name and value that is used to specify variable, HTTP header, POST form field data of query field data. It has the following properties.
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>string</td>
<td>Name of header / variable / POST or GET field.</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Value of header / variable / POST or GET field.</td>
</tr>
</tbody>
</table>

**httptest.create**

**Description**

object httptest.create(object/array webScenarios)

This method allows to create new web scenarios.

Creating a web scenario will automatically create a set of web monitoring items.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See [User roles](#) for more information.

**Parameters**

(object/array) Web scenarios to create.

Additionally to the standard web scenario properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>steps</td>
<td>array</td>
<td>Web scenario steps.</td>
</tr>
<tr>
<td>tags</td>
<td>array</td>
<td>Web scenario tags.</td>
</tr>
</tbody>
</table>

**Return values**

(object) Returns an object containing the IDs of the created web scenarios under the httptestids property. The order of the returned IDs matches the order of the passed web scenarios.

**Examples**

**Creating a web scenario**

Create a web scenario to monitor the company home page. The scenario will have two steps, to check the home page and the "About" page and make sure they return the HTTP status code 200.

**Request:**

```json
{
    "jsonrpc": "2.0",
    "method": "httptest.create",
    "params": {
        "name": "Homepage check",
        "hostid": "10085",
        "steps": [
            {
                "name": "Homepage",
                "url": "http://example.com",
                "status_codes": "200",
                "no": 1
            },
            {
                "name": "Homepage / About",
                "url": "http://example.com/about",
                "status_codes": "200",
                "no": 2
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e"
}
```
Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "httptestids": ["5"]
    },
    "id": 1
}
```

See also

- Scenario step

Source

CHttpTest::create() in ui/include/classes/api/services/CHttpTest.php.

**httptest.delete**

Description

object httptest.delete(array webScenarioIds)

This method allows to delete web scenarios.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the web scenarios to delete.

Return values

(object) Returns an object containing the IDs of the deleted web scenarios under the httptestids property.

Examples

Deleting multiple web scenarios

Delete two web scenarios.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "httptest.delete",
    "params": [
        "2",
        "3"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "httptestids": ["2", "3"
    ]
},
```

1338
Source

CHttpTest::delete() in ui/include/classes/api/services/CHttpTest.php.

**httpstest.get**

**Description**

integer/array httpstest.get(object parameters)

The method allows to retrieve web scenarios according to the given parameters.

This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See **User roles** for more information.

**Parameters**

(object) Parameters defining the desired output.

The method supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupids</td>
<td>string/array</td>
<td>Return only web scenarios that belong to the given host groups.</td>
</tr>
<tr>
<td>hostids</td>
<td>string/array</td>
<td>Return only web scenarios that belong to the given hosts.</td>
</tr>
<tr>
<td>httpstestids</td>
<td>string/array</td>
<td>Return only web scenarios with the given IDs.</td>
</tr>
<tr>
<td>inherited</td>
<td>boolean</td>
<td>If set to <code>true</code> return only web scenarios inherited from a template.</td>
</tr>
<tr>
<td>monitored</td>
<td>boolean</td>
<td>If set to <code>true</code> return only enabled web scenarios that belong to monitored hosts.</td>
</tr>
<tr>
<td>templated</td>
<td>boolean</td>
<td>If set to <code>true</code> return only web scenarios that belong to templates.</td>
</tr>
<tr>
<td>templatedids</td>
<td>string/array</td>
<td>Return only web scenarios that belong to the given templates.</td>
</tr>
<tr>
<td>expandName</td>
<td>flag</td>
<td>Expand macros in the name of the web scenario.</td>
</tr>
<tr>
<td>expandStepName</td>
<td>flag</td>
<td>Expand macros in the names of scenario steps.</td>
</tr>
<tr>
<td>evaltype</td>
<td>integer</td>
<td>Rules for tag searching.</td>
</tr>
<tr>
<td>tags</td>
<td>array of objects</td>
<td>Return only web scenarios with given tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value. Format: [{&quot;tag&quot;: &quot;&lt;tag&gt;&quot;, &quot;value&quot;: &quot;&lt;value&gt;&quot;, &quot;operator&quot;: &quot;&lt;operator&gt;&quot;}, ...]. An empty array returns all web scenarios. Possible operator types: 0 - (default) Like; 1 - Equal; 2 - Not like; 3 - Not equal; 4 - Exists; 5 - Not exists.</td>
</tr>
<tr>
<td>selectHosts</td>
<td>query</td>
<td>Return the hosts that the web scenario belongs to as an array in the hosts property.</td>
</tr>
<tr>
<td>selectSteps</td>
<td>query</td>
<td>Return web scenario steps in the steps property. Supports count.</td>
</tr>
<tr>
<td>selectTags</td>
<td>query</td>
<td>Return the web scenario tags in tags property.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties. Possible values are: httpstestid and name.</td>
</tr>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>These parameters being common for all get methods are described in detail in the reference commentary.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td></td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>sortorder</td>
<td>string/array</td>
<td></td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Return values

(integer/array) Returns either:
- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving a web scenario

Retrieve all data about web scenario "4".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.get",
    "params": {
        "output": "extend",
        "selectSteps": "extend",
        "httptestids": [9]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "httptestid": 9,
            "name": "Homepage check",
            "nextcheck": 0,
            "delay": 1m,
            "status": 0,
            "variables": [],
            "agent": "Zabbix",
            "authentication": 0,
            "http_user": "",
            "http_password": "",
            "hostid": 10084,
            "templateid": 0,
            "http_proxy": "",
            "retries": 1,
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": 0,
            "verify_host": 0,
            "headers": []
        }
    ]
}
```
See also

- Host
- Scenario step

Source

CHttpTest::get() in ui/include/classes/api/services/CHttpTest.php.

**httptest.update**

Description

object httptest.update(object/array webScenarios)

This method allows to update existing web scenarios.

This method is only available to Admin and Super admin user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters
Web scenario properties to be updated.

The `httptestid` property must be defined for each web scenario, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard web scenario properties, the method accepts the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>steps</td>
<td>array</td>
<td>Scenario steps to replace existing steps.</td>
</tr>
<tr>
<td>tags</td>
<td>array</td>
<td>Web scenario tags.</td>
</tr>
</tbody>
</table>

Return values

Returns an object containing the IDs of the updated web scenarios under the `httptestid` property.

Examples

Enabling a web scenario

Enable a web scenario, that is, set its status to “0”.

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "httptest.update",
    "params": {
        "httptestid": "5",
        "status": 0
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```json
{
    "jsonrpc": "2.0",
    "result": {
        "httptestids": [
            "5"
        ],
        "id": 1
    }
}
```

See also

- Scenario step

Source

CHttpTest::update() in ui/include/classes/api/services/CHttpTest.php.

Appendix 1. Reference commentary

**Notation**  Data types

The Zabbix API supports the following data types as input:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>A boolean value, accepts either true or false.</td>
</tr>
<tr>
<td>flag</td>
<td>The value is considered to be true if it is passed and not equal to null and false otherwise.</td>
</tr>
<tr>
<td>integer</td>
<td>A whole number.</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>float</td>
<td>A floating point number.</td>
</tr>
<tr>
<td>string</td>
<td>A text string.</td>
</tr>
<tr>
<td>text</td>
<td>A longer text string.</td>
</tr>
<tr>
<td>timestamp</td>
<td>A Unix timestamp.</td>
</tr>
<tr>
<td>array</td>
<td>An ordered sequence of values, that is, a plain array.</td>
</tr>
<tr>
<td>object</td>
<td>An associative array.</td>
</tr>
<tr>
<td>query</td>
<td>A value which defines, what data should be returned. Can be defined as an array of property names to return only specific properties, or as one of the predefined values: extend - returns all object properties; count - returns the number of retrieved records, supported only by certain subselects.</td>
</tr>
</tbody>
</table>

Zabbix API always returns values as strings or arrays only.

Property labels

Some of the objects properties are marked with short labels to describe their behavior. The following labels are used:

- readonly - the value of the property is set automatically and cannot be defined or changed by the client;
- constant - the value of the property can be set when creating an object, but cannot be changed after.

**Reserved ID value "0"**  Reserved ID value "0" can be used to filter elements and to remove referenced objects. For example, to remove a referenced proxy from a host, proxy_hostid should be set to 0 ("proxy_hostid": "0") or to filter hosts monitored by server option proxyids should be set to 0 ("proxyids": "0").

**Common "get" method parameters**  The following parameters are supported by all get methods:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>countOutput</td>
<td>boolean</td>
<td>Return the number of records in the result instead of the actual data.</td>
</tr>
<tr>
<td>editable</td>
<td>boolean</td>
<td>If set to true return only objects that the user has write permissions to. Default: false.</td>
</tr>
<tr>
<td>excludeSearch</td>
<td>boolean</td>
<td>Return results that do not match the criteria given in the search parameter.</td>
</tr>
<tr>
<td>filter</td>
<td>object</td>
<td>Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against.</td>
</tr>
<tr>
<td>limit</td>
<td>integer</td>
<td>Doesn’t work for text fields. Limit the number of records returned.</td>
</tr>
<tr>
<td>output</td>
<td>query</td>
<td>Object properties to be returned. Default: extend.</td>
</tr>
<tr>
<td>preservekeys</td>
<td>boolean</td>
<td>Use IDs as keys in the resulting array.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>search</td>
<td>object</td>
<td>Return results that match the given wildcard search (case-insensitive). Accepts an array, where the keys are property names, and the values are strings to search for. If no additional options are given, this will perform a LIKE &quot;%…%&quot; search. Works only for string and text fields.</td>
</tr>
<tr>
<td>searchByAny</td>
<td>boolean</td>
<td>If set to true return results that match any of the criteria given in the filter or search parameter instead of all of them. Default: false.</td>
</tr>
<tr>
<td>searchWildcardsEnabled</td>
<td>boolean</td>
<td>If set to true enables the use of &quot;*&quot; as a wildcard character in the search parameter. Default: false.</td>
</tr>
<tr>
<td>sortfield</td>
<td>string/array</td>
<td>Sort the result by the given properties. Refer to a specific API get method description for a list of properties that can be used for sorting. Macros are not expanded before sorting. If no value is specified, data will be returned unsorted.</td>
</tr>
<tr>
<td>sortOrder</td>
<td>string/array</td>
<td>Order of sorting. If an array is passed, each value will be matched to the corresponding property given in the sortfield parameter. Possible values are: ASC - (default) ascending; DESC - descending.</td>
</tr>
<tr>
<td>startSearch</td>
<td>boolean</td>
<td>The search parameter will compare the beginning of fields, that is, perform a LIKE &quot;…%&quot; search instead. Ignored if searchWildcardsEnabled is set to true.</td>
</tr>
</tbody>
</table>

**Examples**  
User permission check

Does the user have permission to write to hosts whose names begin with "MySQL" or "Linux"?

Request:

```json
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "countOutput": true,
    }
}```
''search'': {
    ''host'': ['MySQL', 'Linux'],
},
''editable'': true,
''startSearch'': true,
''searchByAny'': true,
},
''auth'': '766b71ee543230a1182ca5c44d353e36',
''id'': 1
}

Response:
{
    "jsonrpc": "2.0",
    "result": "0",
    "id": 1
}

Zero result means no hosts with read/write permissions.

Mismatch counting

Count the number of hosts whose names do not contain the substring "ubuntu"

Request:
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "countOutput": true,
        "search": {
            "host": "ubuntu"
        },
        "excludeSearch": true
    },
    "auth": '766b71ee543230a1182ca5c44d353e36',
    "id": 1
}

Response:
{
    "jsonrpc": "2.0",
    "result": "44",
    "id": 1
}

Searching for hosts using wildcards

Find hosts whose name contains word "server" and have interface ports "10050" or "10071". Sort the result by host name in descending order and limit it to 5 hosts.

Request:
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ['hostid', 'host'],
        "selectInterfaces": ['port'],
        "filter": {
            "port": ['10050', '10071']
        },
        "search": {
            "host": '*server*'
        }
    },
    "auth": '1345'
}
"searchWildcardsEnabled": true,
"searchByAny": true,
"sortfield": "host",
"sortorder": "DESC",
"limit": 5
},
"auth": "766b71ee543230a1182ca5c44d353e36",
"id": 1}

Response:
{
  "jsonrpc": "2.0",
  "result": [
    {
      "hostid": "50003",
      "host": "WebServer-Tomcat02",
      "interfaces": [
        {
          "port": "10071"
        }
      ]
    },
    {
      "hostid": "50005",
      "host": "WebServer-Tomcat01",
      "interfaces": [
        {
          "port": "10071"
        }
      ]
    },
    {
      "hostid": "50004",
      "host": "WebServer-Nginx",
      "interfaces": [
        {
          "port": "10071"
        }
      ]
    },
    {
      "hostid": "99032",
      "host": "MySQL server 01",
      "interfaces": [
        {
          "port": "10050"
        }
      ]
    },
    {
      "hostid": "99061",
      "host": "Linux server 01",
      "interfaces": [
        {
          "port": "10050"
        }
      ]
    }
  ],
  "id": 1
}
Searching for hosts using wildcards with "preservekeys"

If you add the parameter "preservekeys" to the previous request, the result is returned as an associative array, where the keys are the id of the objects.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid", "host"],
        "selectInterfaces": ["port"],
        "filter": {
            "port": ["10050", "10071"]
        },
        "search": {
            "host": "*server*"
        },
        "searchWildcardsEnabled": true,
        "searchByAny": true,
        "sortfield": "host",
        "sortorder": "DESC",
        "limit": 5,
        "preservekeys": true
    },
    "auth": "766b71ee543230a1182ca5c44d353e36",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "50003": {
            "hostid": "50003",
            "host": "WebServer-Tomcat02",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        "50005": {
            "hostid": "50005",
            "host": "WebServer-Tomcat01",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        "50004": {
            "hostid": "50004",
            "host": "WebServer-Nginx",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        "99032": {
            "hostid": "99032",
            "host": "MySQL server 01",
            ...
        }
    }
}
```
Appendix 2. Changes from 6.0 to 6.2

**Backward incompatible changes**  authentication

Changes:
ZBXNEXT-2289 authentication.get, authentication.create, authentication.update: removed properties ldap_host, ldap_port, ldap_base_dn, ldap_search_attribute, ldap_bind_dn, ldap_bind_password.

configuration

Changes:
ZBXNEXT-2592 configuration.export: option groups is not supported, instead new options introduced host_groups and template_groups.

hostgroup

Changes:
ZBXNEXT-2592 hostgroup object does not have property internal anymore.
ZBXNEXT-2592 hostgroup.get: removed options templated_hosts, with_hosts_and_templates, selectTemplates.
ZBXNEXT-2592 hostgroup.massadd, hostgroup.massupdate: does not accept templates parameter anymore.
ZBXNEXT-2592 hostgroup.massremove: does not accept templateids parameter anymore.

template

Changes:
ZBXNEXT-2592 template.create, template.massadd, template.massupdate, template.update: parameter groups now accepts only template groups and must contain template group groupid.
ZBXNEXT-2592 template.massremove: parameter groupids is now template groups groupids.

trigger

Changes:
ZBX-20613 trigger.adddependencies, trigger.deletedeprendencies: support for the methods dropped.

**Other changes and bug fixes**  auditlog

ZBXNEXT-2592 added new resourcetype (50 - Template group).
ZBXNEXT-2289 added new resourcetype (49 - LDAP user directory).
ZBXNEXT-1580 added new action (11 - Config refresh).
authentication

ZBXNEXT-2289 authentication.get, authentication.create, authentication.update: added property ldap_directory_id.

event

ZBXNEXT-721 event.acknowledge: added support of new acknowledge actions 32 - suppress and 64 - unsuppress.
ZBXNEXT-721 event.acknowledge: added new property suppress_until to specify time for suppress action.
ZBXNEXT-721 event.get: added new parameter user_id to suppression_data property.

graph

Changes:
ZBXNEXT-2592 graph.get: deprecated option selectGroups, new options selectHostGroups, selectTemplateGroups.

graphprototype

Changes:
ZBXNEXT-2592 graphprototype.get: deprecated option selectGroups, new options selectHostGroups, selectTemplateGroups.

host

Changes:
ZBXNEXT-2592 host.get: deprecated option selectGroups, new option selectHostGroups.
ZBXNEXT-5517 host.update: added new field automatic to macros property with possible value "0" to change state of existing discovered host macro to manually user created macro.
ZBXNEXT-5517 host.update: added possibility to add user macros to discovered hosts.
ZBXNEXT-5517 host.get: added new field automatic to macros property that determines if macro on discovered host was created by discovery rule or manually created by user (0 - user created macro, 1 - macro created by discovery rule).
ZBXNEXT-7591 host.get: added new field automatic to tags property that determines if tag on discovered host was created by discovery rule or manually created by user (0 - user created tag, 1 - tag created by discovery rule).
ZBXNEXT-5088 host.get: added new property for host active interface availability status active_available (0 - unknown, 1 - available, 2 - not available).
ZBXNEXT-7523 host.get: option selectParentTemplates returns a new property link_type (0 - manually linked, 1 - linked by LLD) and allows only to select template related fields "templateid", "host", "name", "description", "uuid" and "link_type". As well as "extend" or "count".
ZBXNEXT-7523 host.massremove: properties templateids and templateids_clear allow to remove only templates that are manually linked. Automatically linked templates by LLD are skipped.
ZBXNEXT-7523 host.massupdate, host.update: properties templates and templates_clear are now allowed for discovered hosts. Only manually linked templates can be removed.

hostgroup

Changes:
ZBXNEXT-2592 hostgroup.get: deprecated options monitored_hosts, real_hosts, new options with_hosts, with_monitored_hosts.
ZBXNEXT-2592 hostgroup.propagate: new method.

maintenance

Changes:
ZBXNEXT-2592 maintenance.get: deprecated option selectGroups, new option selectHostGroups.

problem

Changes:
ZBXNEXT-721 problem.get: added new parameter user_id to suppression_data property.

role

Changes:
ZBXNEXT-4768 rules -> actions property name accepts new value "invoke_execute_now" and property status has possible values: 0 - (default) user cannot execute item check if user has only read permissions to host, 1 - user may execute item check
even though user has only read permissions to host. This value is ignored for super admins.

**script**

Changes:

**ZBXNEXT-2592** script.get: deprecated option selectGroups, new option selectHostGroups.

**settings**

Changes:

**ZBXNEXT-7402** settings.get: new property vault_provider (0 - HashiCorp Vault, 1 - CyberArk Vault).

**task**

Changes:

**ZBXNEXT-1580** added new type (2 - Refresh proxy configuration) and new field proxy_hostids in request property.

**ZBXNEXT-4768** task.create: tasks with type "6" (Execute now) now accepts dependent items, but only if top level master item is of allowed type: (0 - Zabbix agent, 3 - Simple check, 5 - Zabbix internal, 10 - External check, 11 - Database monitor, 12 - IPMI agent, 13 - SSH agent, 14 - TELNET agent, 15 - Calculated, 16 - JMX agent, 19 - HTTP agent, 20 - SNMP agent, 21 - Script).

**template**

Changes:

**ZBXNEXT-2592** template.get: deprecated option selectGroups, new option selectTemplateGroups.

**ZBXNEXT-7523** template.get: option selectParentTemplates returns a new property “link_type” (0 - manually linked, 1 - linked by LLD) and allows only to select template related fields “templateid”, “host”, “name”, “description” and “uuid”. As well as “extend” or “count”.

**templategroup**

Changes:

**ZBXNEXT-2592** added new templategroup API introducing new methods: templategroup.create, templategroup.delete, templategroup.get, templategroup.massadd, templategroup.massremove, templategroup.massupdate, templategroup.propagate, templategroup.update.

**trigger**

Changes:

**ZBXNEXT-2592** trigger.get: deprecated option selectGroups, new options selectHostGroups, selectTemplateGroups.

**triggerprototype**

Changes:

**ZBXNEXT-2592** triggerprototype.get: deprecated option selectGroups, new option selectHostGroups, selectTemplateGroups.

**user**

Changes:

**ZBXNEXT-2289** user.checkAuthentication: returns a new property userdirectoryid.

**ZBXNEXT-2289** user.login: option userData returns a new property userdirectoryid.

**userdirectory**

Changes:

**ZBXNEXT-2289** added new userdirectory API introducing new methods userdirectory.get, userdirectory.create, userdirectory.update, userdirectory.delete, userdirectory.test.

**usergroup**

Changes:

**ZBXNEXT-2592** usergroup.get: deprecated option selectRights, new options selectHostGroupRights, selectTemplateGroupRights.

**ZBXNEXT-2289** usergroup.create, usergroup.update, usergroup.get: added property userdirectoryid.
usermacro

Changes:

ZBXNEXT-2592 usermacro.get: deprecated option selectGroups, new options selectHostGroups, selectTemplateGroups.
ZBXNEXT-5517 usermacro.update: added new field automatic to macros property with possible value "0" to change state of existing discovered host macro to manually user created macro.
ZBXNEXT-5517 usermacro.update: added possibility to add user macros to discovered hosts.
ZBXNEXT-5517 usermacro.get: added new field automatic to macros property that determines if macro on discovered host was created by discovery rule or manually created by user (0 - user created macro, 1 - macro created by discovery rule).

Zabbix API changes in 6.2

6.2.1 graph

Changes:

ZBX-7706 graph.get: Graph availability doesn't depend on permissions to items specified in graph "ymin_itemid" and "ymax_itemid" fields. Graph having MIN or MAX Y axis linked to inaccessible items will still be accessible but MIN/MAX Y axis works the same way as if specified calculation method is "Calculated".

graphprototype

Changes:

ZBX-7706 graphprototype.get: Graph prototype availability doesn't depend on permissions to items specified in graph prototype "ymin_itemid" and "ymax_itemid" fields.

20. Modules

Overview  It is possible to enhance Zabbix frontend functionality by adding 3rd party modules or by developing your own modules without the need to change the source code of Zabbix.

Note that the module code will run with the same privileges as Zabbix source code. This means:

- 3rd party modules can be harmful. You must trust the modules you are installing;
- Errors in a 3rd party module code may crash the frontend. If this happens, just remove the module code from the frontend. As soon as you reload Zabbix frontend, you'll see a note saying that some modules are absent. Go to Module administration (in Administration → General → Modules) and click Scan directory again to remove non-existent modules from the database.

Installation  Please always read the installation manual for a particular module. It is recommended to install new modules one by one to catch failures easily.

Just before you install a module:

- Make sure you have downloaded the module from a trusted source. Installation of harmful code may lead to consequences, such as data loss
- Different versions of the same module (same ID) can be installed in parallel, but only a single version can be enabled at once

Steps to install a module:

- Unpack your module within its own folder in the modules folder of the Zabbix frontend
- Ensure that your module folder contains at least the manifest.json file
- Navigate to Module administration and click the Scan directory button
- New module will appear in the list along with its version, author, description and status
- Enable module by clicking on its status

Troubleshooting:
### Problem Solution

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module did not appear in the list</td>
<td>Make sure that the manifest.json file exists in modules/your-module/ folder of the Zabbix frontend. If it does that means the module does not suit the current Zabbix version. If manifest.json file does not exist, you have probably unpacked in the wrong directory.</td>
</tr>
<tr>
<td>Frontend crashed</td>
<td>The module code is not compatible with the current Zabbix version or server configuration. Please delete module files and reload the frontend. You’ll see a notice that some modules are absent. Go to Module administration and click Scan directory again to remove non-existent modules from the database.</td>
</tr>
<tr>
<td>Error message about identical namespace, ID or actions appears</td>
<td>New module tried to register a namespace, ID or actions which are already registered by other enabled modules. Disable the conflicting module (mentioned in error message) prior to enabling the new one.</td>
</tr>
<tr>
<td>Technical error messages appear</td>
<td>Report errors to the developer of the module.</td>
</tr>
</tbody>
</table>

### Developing modules

Modules are written in PHP language. Model-view-controller (MVC) software pattern design is preferred, as it is also used in Zabbix frontend and will ease the development. PHP strict typing is also welcome but not mandatory.

Please note that with modules you can easily add new menu items and respective views and actions to Zabbix frontend. Currently it is not possible to register new API or create new database tables through modules.

### Module structure

Each module is a directory (placed within the modules directory) with sub-directories containing controllers, views and any other code:

- **example_module_directory/** (required)
  - **manifest.json** (required) Metadata and action definition.
  - **Module.php** Module initialization and event handling.
  - **actions/** Action controller files.
    - **SomethingView.php**
    - **SomethingCreate.php**
    - **SomethingDelete.php**
    - **data_export/**
      - **ExportAsXml.php**
      - **ExportAsExcel.php**
  - **views/** View files.
    - **example.something.view.php**
    - **example.something.delete.php**
  - **js/**
    - **example.something.view.js.php**
  - **partials/** View partial files.
    - **example.something.reusable.php**
  - **js/**
    - **example.something.reusable.js.php**

As you can see, the only mandatory file within the custom module directory is manifest.json. The module will not register without this file. Module.php is responsible for registering menu items and processing events such as ‘onBeforeAction’ and ‘onTerminate’. The actions, views and partials directories contain PHP and JavaScript code needed for module actions.

### Naming convention

Before you create a module, it is important to agree on the naming convention for different module items such as directories and files so that we could keep things well organized. You can also find examples above, in the Module structure section.

<table>
<thead>
<tr>
<th>Item</th>
<th>Naming rules</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module directory</td>
<td>Lowercase [a-z], underscore and decimal digits</td>
<td>example_v2</td>
</tr>
<tr>
<td>Action subdirectories</td>
<td>Lowercase [a-z] and underscore character</td>
<td>data_export</td>
</tr>
<tr>
<td>Action files</td>
<td>CamelCase, ending with action type</td>
<td>SomethingView.php</td>
</tr>
</tbody>
</table>
### Item Naming rules

| View and partial files | Lowercase [a-z] | module.example.something.view.php |
| Javascript files      | The same rules apply as for view and partial files, except the .js.php file extension. | module.example.something.view.js.php |

Note that the ‘module’ prefix and name inclusion is mandatory for view and partial file names, unless you need to override Zabbix core views or partials. This rule, however, does not apply to action file names.

### Manifest preparation

Each module is expected to have a manifest.json file with the following fields in JSON format:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>manifest_version</td>
<td>Yes</td>
<td>Double</td>
<td>-</td>
<td>Manifest version of the module. Currently supported version is 1.</td>
</tr>
<tr>
<td>id</td>
<td>Yes</td>
<td>String</td>
<td>-</td>
<td>Module ID. Only one module with given ID can be enabled at the same time.</td>
</tr>
<tr>
<td>name</td>
<td>Yes</td>
<td>String</td>
<td>-</td>
<td>Module name as displayed in the Administration section.</td>
</tr>
<tr>
<td>version</td>
<td>Yes</td>
<td>String</td>
<td>-</td>
<td>Module version as displayed in the Administration section.</td>
</tr>
<tr>
<td>namespace</td>
<td>Yes</td>
<td>String</td>
<td>-</td>
<td>PHP namespace for Module.php and action classes.</td>
</tr>
<tr>
<td>author</td>
<td>No</td>
<td>String</td>
<td>&quot;&quot;</td>
<td>Module author as displayed in the Administration section.</td>
</tr>
<tr>
<td>url</td>
<td>No</td>
<td>String</td>
<td>&quot;&quot;</td>
<td>Module URL as displayed in the Administration section.</td>
</tr>
<tr>
<td>description</td>
<td>No</td>
<td>String</td>
<td>&quot;&quot;</td>
<td>Module description as displayed in the Administration section.</td>
</tr>
<tr>
<td>actions</td>
<td>No</td>
<td>Object</td>
<td>{}</td>
<td>Actions to register with this module. See Actions.</td>
</tr>
<tr>
<td>config</td>
<td>No</td>
<td>Object</td>
<td>{}</td>
<td>Module configuration.</td>
</tr>
</tbody>
</table>

For reference, please see an example of manifest.json in the Reference section.

### Actions

The module will have control over frontend actions defined within the actions object in the manifest.json file. This way new actions are defined. In the same way you may redefine existing actions. Each key of actions should represent the action name and the corresponding value should contain class and optionally layout and view keys.

One action is defined by four counterparts: name, controller, view and layout. Data validation and preparation is typically done in the controller, output formatting is done in the view or partials, and the layout is responsible for decorating the page with elements such as menu, header, footer and others.

Module actions must be defined in the manifest.json file as actions object:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>key</em></td>
<td>Yes</td>
<td>String</td>
<td>-</td>
<td>Action name, in lowercase [a-z], separating words with dot.</td>
</tr>
<tr>
<td>class</td>
<td>Yes</td>
<td>String</td>
<td>-</td>
<td>Action class name, including subdirectory path (if used) within the actions directory.</td>
</tr>
<tr>
<td>view</td>
<td>No</td>
<td>String</td>
<td>null</td>
<td>Action view.</td>
</tr>
</tbody>
</table>

1353
There are several predefined layouts, like layout.json or layout.xml. These are intended for actions which produce different result than an HTML. You may explore predefined layouts in the app/views/ directory or even create your own.

Sometimes it is necessary to only redefine the view part of some action leaving the controller intact. In such case just place the necessary view and/or partial files inside the views directory of the module.

For reference, please see an example action controller file in the Reference section. Please do not hesitate to explore current actions of Zabbix source code, located in the app/ directory.

**Module.php**

This optional PHP file is responsible for module initialization as well as event handling. Class ‘Module’ is expected to be defined in this file, extending base class \Core\CModule. The Module class must be defined within the namespace specified in the manifest.json file.

```php
<?php
namespace Modules\Example;
use \Core\CModule as BaseModule;

class Module extends BaseModule {
    ...
}
```

For reference, please see an example of Module.php in the Reference section.

**Reference**  This section contains basic versions of different module elements introduced in the previous sections.

**manifest.json**

```json
{
    "manifest_version": 1.0,
    "id": "example_module",
    "name": "Example module",
    "version": "1.0",
    "namespace": "Example",
    "author": "John Smith",
    "url": "http://module.example.com",
    "description": "Short description of the module.",
    "actions": {
        "example.something.view": {
            "class": "SomethingView",
            "view": "module.example.something.view"
        },
        "example.something.create": {
            "class": "SomethingCreate",
            "layout": null
        },
        "example.something.delete": {
            "class": "SomethingDelete",
            "layout": null
        },
        "example.something.export.xml": {
            "class": "data_export/ExportAsXml",
            "layout": null
        },
        "example.something.export.excel": {
            "class": "data_export/ExportAsExcel",
            "layout": null
        }
    },
    "config": {
        "username": "john_smith"
    }
}
```

**Module.php**
<?php declare(strict_types = 1);

namespace Modules\Example;

use APP;
use CController as CAction;

/**
 * Please see Core\CModule class for additional reference.
 */
class Module extends \Core\CModule {

 /**
  * Initialize module.
  */
  public function init(): void {
      // Initialize main menu (CMenu class instance).
      APP::Component() ->get('menu.main')
          ->findOrAdd(_('Reports'))
          ->getSubmenu()
              ->add((new \CMenuItem(_('Example wide report')))
                  ->setAction('example.report.wide.php'))
              ->add((new \CMenuItem(_('Example narrow report')))
                  ->setAction('example.report.narrow.php'));
  }

 /**
  * Event handler, triggered before executing the action.
  *
  * @param CAction $action Action instance responsible for current request.
  */
  public function onBeforeAction(CAction $action): void {
  }

 /**
  * Event handler, triggered on application exit.
  *
  * @param CAction $action Action instance responsible for current request.
  */
  public function onTerminate(CAction $action): void {
  }
}

Action controller
<?php declare(strict_types = 1);

namespace Modules\Example\Actions;

use CControllerResponseData;
use CControllerResponseFatal;
use CController as CAction;

/**
 * Example module action.
 */
class SomethingView extends CAction {

 /**
  * Initialize action. Method called by Zabbix core.
  */
  public function init(): void {
  }
}
* @return void
*/

public function init(): void {
    /**
     * Disable SID (Session ID) validation. Session ID validation should only be used for actions which modify,
     * such as update or delete actions. In such case Session ID must be presented in the URL, so that the URL would expire as soon as the session expired.
     */
    $this→disableSIDvalidation();
}
/**
 * Check and sanitize user input parameters. Method called by Zabbix core. Execution stops if false is returned.
 * @return bool true on success, false on error.
 */
protected function checkInput(): bool {
    $fields = [
        'name' => 'required|string',
        'email' => 'required|string',
        'phone' => 'string'
    ];

    // Only validated data will further be available using $this->hasInput() and $this->getInput().
    $ret = $this->validateInput($fields);
    if (!$ret) {
        $this->setResponse(new CControllerResponseFatal());
    }
    return $ret;
}
/**
 * Check if the user has permission to execute this action. Method called by Zabbix core.
 * Execution stops if false is returned.
 * @return bool
 */
protected function checkPermissions(): bool {
    $permit_user_types = [USER_TYPE_ZABBIX_ADMIN, USER_TYPE_SUPER_ADMIN];

    return in_array($this->getUserType(), $permit_user_types);
}
/**
 * Prepare the response object for the view. Method called by Zabbix core.
 * @return void
 */
protected function doAction(): void {
    $contacts = $this->getInput('email');

    if ($this->hasInput('phone')) {
        $contacts .= ', ', $this->getInput('phone');
    }

    $data = [
        'name' => $this->getInput('name'),
        'contacts' => $contacts
    ];
$response = new CControllerResponseData($data);

$this->setResponse($response);
}

Action view

<?php declare(strict_types = 1);
/**
 * @var CView $this
 */
$this->includeJsFile('example.something.view.js.php');

(new CWidget())
  ->setTitle(_('Something view'))
  ->addItem(new CDiv($data['name']))
  ->addItem(new CPartial('module.example.something.reusable', [
    'contacts' => $data['contacts']
  ])
)->show();

21. Appendixes

Please use the sidebar to access content in the Appendixes section.

1 Frequently asked questions / Troubleshooting

Frequently asked questions or FAQ.

1. Q: Can I flush/clear the queue (as depicted in Administration → Queue)?
   A: No.
2. Q: How do I migrate from one database to another?
   A: Dump data only (for MySQL, use flag -t or --no-create-info), create the new database using schema files from Zabbix and import the data.
3. Q: I would like to replace all spaces with underscores in my item keys because they worked in older versions but space is not a valid symbol for an item key in 3.0 (or any other reason to mass-modify item keys). How should I do it and what should i beware of?
   A: You may use a database query to replace all occurrences of spaces in item keys with underscores:
   update items set key_=replace(key_,'','_');
   Triggers will be able to use these items without any additional modifications, but you might have to change any item references in these locations:
   * Notifications (actions)
   * Map element and link labels
   * Calculated item formulas
4. Q: My graphs have dots instead of lines or empty areas. Why so?
   A: Data is missing. This can happen for a variety of reasons - performance problems on Zabbix database, Zabbix server, network, monitored devices...
5. Q: Zabbix daemons fail to start up with a message Listener failed with error: socket() for [|-]:10050 failed with error 22: invalid argument.
   A: This error arises at attempt to run Zabbix agent compiled on version 2.6.27 or above on a platform with a kernel 2.6.26 and lower. Note that static linking will not help in this case because it is the socket() system call that does not support SOCK_CLOEXEC flag on earlier kernels. ZBX-3395
6. Q: I try to set up a flexible user parameter (one that accepts parameters) with a command that uses a positional parameter like $1, but it doesn’t work (uses item parameter instead). How to solve this?
   A: Use a double dollar sign like $$1
7. Q: All dropdowns have a scrollbar and look ugly in Opera 11. Why so?
   A: It’s a known bug in Opera 11.00 and 11.01; see Zabbix issue tracker for more information.
8. Q: How can I change graph background color in a custom theme?
   A: See graph_theme table in the database and theming guide.
9. Q: With DebugLevel 4 I’m seeing messages “Trapper got [] len 0” in server/proxy log - what’s that?
   A: Most likely that is frontend, connecting and checking whether server is still running.
10. Q: My system had the time set in the future and now no data is coming in. How could this be solved?
    A: Clear values of database fields hostsdisable_until*, drules.nextcheck, httptest.nextcheck and restart the server/proxy.
11. Q: Text item values in frontend (when using {ITEM.VALUE} macro and in other cases) are cut/trimmed to 20 symbols. Is that normal?
    A: Yes, there is a hardcoded limit in include/items.inc.php currently.

If you haven’t found answer to your question try Zabbix forum

2 Installation and setup

1 Database creation

Overview

A Zabbix database must be created during the installation of Zabbix server or proxy.

This section provides instructions for creating a Zabbix database. A separate set of instructions is available for each supported database.

UTF-8 is the only encoding supported by Zabbix. It is known to work without any security flaws. Users should be aware that there are known security issues if using some of the other encodings.

If installing from Zabbix Git repository, you need to run:

\$ make dbschema

prior to proceeding to the next steps.

MySQL

Character sets utf8 (aka utf8mb3) and utf8mb4 are supported (with utf8_bin and utf8mb4_bin collation respectively) for Zabbix server/proxy to work properly with MySQL database. It is recommended to use utf8mb4 for new installations.

Deterministic triggers need to be created during the import of schema. On MySQL and MariaDB, this requires GLOBAL log_bin_trust_function_creators = 1 to be set if binary logging is enabled and there is no superuser privileges and log_bin_trust_functionCreators = 1 is not set in MySQL configuration file.

If you are installing from Zabbix packages, proceed to the instructions for your platform.

If you are installing Zabbix from sources:

- Create and configure a database and a user.

  shell> mysql -uroot -p<password>
  mysql> create database zabbix character set utf8mb4 collate utf8mb4_bin;
  mysql> create user 'zabbix'@'localhost' identified by '<password>';
  mysql> grant all privileges on zabbix.* to 'zabbix'@'localhost';
  mysql> SET GLOBAL log_bin_trust_function_creators = 1;
  mysql> quit;

- Import the data into the database. For a Zabbix proxy database, only schema.sql should be imported (no images.sql nor data.sql).

  shell> cd database/mysql
  shell> mysql -uzabbix -p<password> zabbix < schema.sql
  #### stop here if you are creating database for Zabbix proxy
  shell> mysql -uzabbix -p<password> zabbix < images.sql
  shell> mysql -uzabbix -p<password> zabbix < data.sql

log_bin_trust_function_creators can be disabled after the schema has been successfully imported:
shell> mysql -uroot -p<password>
mysql> SET GLOBAL log_bin_trust_function_creators = 0;
mysql> quit;

PostgreSQL

You need to have database user with permissions to create database objects.

If you are installing from Zabbix packages, proceed to the instructions for your platform.

If you are installing Zabbix from sources:

• Create a database user.

  The following shell command will create user zabbix. Specify a password when prompted and repeat the password (note, you may first be asked for sudo password):

  shell> sudo -u postgres createuser --pwprompt zabbix

• Create a database.

  The following shell command will create the database zabbix (last parameter) with the previously created user as the owner (-O zabbix).

  shell> sudo -u postgres createdb -O zabbix -E Unicode -T template0 zabbix

• Import the initial schema and data (assuming you are in the root directory of Zabbix sources).

  shell> cd database/postgresql
  shell> cat schema.sql | sudo -u zabbix psql zabbix

  #### stop here if you are creating database for Zabbix proxy
  shell> cat images.sql | sudo -u zabbix psql zabbix
  shell> cat data.sql | sudo -u zabbix psql zabbix

  The above commands are provided as an example that will work in most of GNU/Linux installations. You can use different commands, e.g. "psql -U <username>" depending on how your system/database are configured. If you have troubles setting up the database please consult your Database administrator.

TimescaleDB

Instructions for creating and configuring TimescaleDB are provided in a separate section.

Oracle

Instructions for creating and configuring Oracle database are provided in a separate section.

SQLite

Using SQLite is supported for Zabbix proxy only!

The database will be automatically created if it does not exist.

Return to the installation section.

2 Repairing Zabbix database character set and collation

MySQL/MariaDB

Historically, MySQL and derivatives used ‘utf8’ as an alias for utf8mb3 - MySQL’s own 3-byte implementation of the standard UTF8, which is 4-byte. Starting from MySQL 8.0.28 and MariaDB 10.6.1, ‘utf8mb3’ character set is deprecated and at some point its support will be dropped while ‘utf8’ will become a reference to ‘utf8mb4’. Since Zabbix 6.0, ‘utf8mb4’ is supported. To avoid future problems, it is highly recommended to use ‘utf8mb4’. Another advantage of switching to ‘utf8mb4’ is support of supplementary Unicode characters.

As versions before Zabbix 6.0 are not aware of utf8mb4, make sure to first upgrade Zabbix server and DB schema to 6.0.x or later before executing utf8mb4 conversion.

1. Check the database character set and collation.

For example:
As we see, the character set here is not 'utf8mb4' and collation is not 'utf8mb4_bin', so we need to fix them.

2. Stop Zabbix.

3. Create a backup copy of the database!

4. Fix the character set and collation on database level:
   alter database <your DB name> character set utf8mb4 collate utf8mb4_bin;

Fixed values:

5. Load the script to fix character set and collation on table and column level:
   mysql <your DB name> < utf8mb4_convert.sql

6. Execute the script:

   SET @ZABBIX_DATABASE = '<your DB name>';
   If MariaDB → set innodb_strict_mode = OFF;
   CALL zbx_convert_utf8();
   If MariaDB → set innodb_strict_mode = ON;
   drop procedure zbx_convert_utf8;

Please note that 'utf8mb4' is expected to consume slightly more disk space.

7. If no errors - you may want to create a database backup copy with the fixed database.

8. Start Zabbix.

3 Database upgrade to primary keys

Overview

Since Zabbix 6.0, primary keys are used for all tables in new installations.

This section provides instructions for manually upgrading the history tables in existing installations to primary keys.

Instructions are available for:

- MySQL
- PostgreSQL
- TimescaleDB v1
- TimescaleDB v2
- Oracle

Important notes

- Make sure to back up the database before the upgrade
- If your database uses partitions, contact your DB administrator or Zabbix support team for help
• The CSV files can be removed after a successful upgrade to primary keys
• Zabbix must not be running during upgrade
• Optionally, Zabbix frontend may be switched to maintenance mode
• Upgrading to primary keys should be done only after upgrading Zabbix server to 6.0
• On proxy, history_pk_prepare.sql can be executed to upgrade history tables (that are not used) to primary keys.

MySQL
Export and import must be performed in tmux/screen, so that the session isn’t dropped.

See also: Important notes

MySQL 8.0+ (with mysqlsh)
• Rename old tables, create new tables by running history_pk_prepare.sql.
  mysql -uzabbix -p<password> zabbix < /usr/share/doc/zabbix-sql-scripts/mysql/history_pk_prepare.sql
• Export and import data
  mysqlish should be installed. mysqlish should be able to connect to the DB. If connection is done through socket, it could be needed to explicitly state a path to it.
  Connect via mysqlish:
  sudo mysqlsh -uroot -S /run/mysqld/mysqld.sock --no-password -Dzabbix
  Run (CSVPATH should/could be tweaked according to needs):
  CSVPATH="/var/lib/mysql-files";
  util.exportTable("history_old", CSVPATH + "/history.csv", {dialect: "csv"});
  util.importTable(CSVPATH + "/history.csv", {"dialect": "csv", "table": "history" });
  ...
  • Verify that everything works as supposed
  • Drop old tables
    DROP TABLE history_old; DROP TABLE history_uint_old; DROP TABLE history_str_old; DROP TABLE history_log_old; DROP TABLE history_text_old;
MySQL 8.0+ (without mysqlsh) or MariaDB
This option is slower and more time consuming, use only if there is a reason not to use mysqlsh.
You must log in as a root (recommended), or with any user that has FILE privileges.
MySQL should be started with local_infile variable enabled.
• Rename old tables, create new tables by running history_pk_prepare.sql.
  mysql -uzabbix -p<password> zabbix < /usr/share/doc/zabbix-sql-scripts/mysql/history_pk_prepare.sql
• Export and import data
  Check if import/export is enabled only for files in the specific path:
  mysql> SELECT @@secure_file_priv;
  +-----------------------+
  | /var/lib/mysql-files/ |
  +-----------------------+
  | @secure_file_priv |
If the value is a path to directory, export/import could be performed for files in that directory. In this case paths to files in queries
should be edited accordingly. Alternatively, secure_file_priv could be disabled (set to empty string) during upgrade. If the value
is empty, export/import could be performed to/from files that could be located anywhere. If secure_file_priv is NULL, set it to the
path where exported table data will be stored (in the example above, it is ’/var/lib/mysql-files/’). For more details, see MySQL
documentation).
max_execution_time should be disabled before exporting data to avoid timeout during export.

SET @@max_execution_time=0;

SELECT * INTO OUTFILE '/var/lib/mysql-files/history.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINES TER
LOAD DATA INFILE '/var/lib/mysql-files/history.csv' IGNORE INTO TABLE history FIELDS TERMINATED BY ',' ESC

SELECT * INTO OUTFILE '/var/lib/mysql-files/history_uint.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINE
LOAD DATA INFILE '/var/lib/mysql-files/history_uint.csv' IGNORE INTO TABLE history_uint FIELDS TERMINATED

SELECT * INTO OUTFILE '/var/lib/mysql-files/history_str.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINES
LOAD DATA INFILE '/var/lib/mysql-files/history_str.csv' IGNORE INTO TABLE history_str FIELDS TERMINATED BY

SELECT * INTO OUTFILE '/var/lib/mysql-files/history_log.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINES
LOAD DATA INFILE '/var/lib/mysql-files/history_log.csv' IGNORE INTO TABLE history_log FIELDS TERMINATED BY

SELECT * INTO OUTFILE '/var/lib/mysql-files/history_text.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINE
LOAD DATA INFILE '/var/lib/mysql-files/history_text.csv' IGNORE INTO TABLE history_text FIELDS TERMINATED
• Verify that everything works as supposed
• Drop old tables
DROP TABLE history_old; DROP TABLE history_uint_old; DROP TABLE history_str_old; DROP TABLE history_log_old; DROP
TABLE history_text_old;
PostgreSQL
Export and import must be performed in tmux/screen, so that the session isn’t dropped.
See also: Important notes
Upgrading tables
• Rename tables using

history_pk_prepare.sql.

sudo -u zabbix psql zabbix < /usr/share/doc/zabbix-sql-scripts/postgresql/history_pk_prepare.sql
• Export current history, import it to the temp table, and insert it into new tables while ignoring duplicates

\copy history_old TO '/tmp/history.csv' DELIMITER ',' CSV
CREATE TEMP TABLE temp_history (
itemid
bigint
NOT NULL,
clock
integer
DEFAULT '0'
NOT NULL,
value
DOUBLE PRECISION DEFAULT '0.0000'
NOT NULL,
ns
integer
DEFAULT '0'
NOT NULL
);
\copy temp_history FROM '/tmp/history.csv' DELIMITER ',' CSV
INSERT INTO history SELECT * FROM temp_history ON CONFLICT (itemid,clock,ns) DO NOTHING;
\copy history_uint_old TO '/tmp/history_uint.csv' DELIMITER ',' CSV
CREATE TEMP TABLE temp_history_uint (
itemid
bigint
NOT NULL,
clock
integer
DEFAULT '0'
NOT NULL,
value
numeric(20)
DEFAULT '0'
NOT NULL,
ns
integer
DEFAULT '0'
NOT NULL
);
\copy temp_history_uint FROM '/tmp/history_uint.csv' DELIMITER ',' CSV
INSERT INTO history_uint SELECT * FROM temp_history_uint ON CONFLICT (itemid,clock,ns) DO NOTHING;
\copy history_str_old TO '/tmp/history_str.csv' DELIMITER ',' CSV
CREATE TEMP TABLE temp_history_str (
itemid
bigint

1362

NOT NULL,


\copy temp_history_str FROM '/tmp/history_str.csv' DELIMITER ',' CSV
INSERT INTO history_str (itemid,clock,value,ns) SELECT * FROM temp_history_str ON CONFLICT (itemid,clock,ns) DO NOTHING;

\copy temp_history_log FROM '/tmp/history_log.csv' DELIMITER ',' CSV
INSERT INTO history_log SELECT * FROM temp_history_log ON CONFLICT (itemid,clock,ns) DO NOTHING;

\copy temp_history_text FROM '/tmp/history_text.csv' DELIMITER ',' CSV
INSERT INTO history_text SELECT * FROM temp_history_text ON CONFLICT (itemid,clock,ns) DO NOTHING;

• Verify that everything works as supposed
• Drop old tables
  DROP TABLE history_old; DROP TABLE history_uint_old; DROP TABLE history_str_old; DROP TABLE history_log_old; DROP TABLE history_text_old;

Consider using the following tips to improve insert performance:
• PostgreSQL: Bulk Loading Huge Amounts of Data
• Checkpoint Distance and Amount of WAL

TimescaleDB v1.x

Export and import must be performed in tmux/screen, so that the session isn’t dropped.

See also: Important notes

Upgrading tables
• Rename tables using history_pk_prepare.sql.
  sudo -u zabbix psql zabbix < /usr/share/doc/zabbix-sql-scripts/postgresql/history_pk_prepare.sql
  • Example of upgrading for one table:
    -- Verify that there is enough space to allow export of uncompressed data
    select sum(before_compression_total_bytes)/1024/1024 as before_compression_total_mbytes, sum(after_compression_total_bytes)/1024/1024 as after_compression_total_mbytes FROM chunk_compression_stats('history_uint_old');

    -- Export data
    \copy (select * from history_uint_old) TO '/tmp/history_uint.csv' DELIMITER ',' CSV
-- Import data
\copy temp_history_uint FROM '/tmp/history_uint.csv' DELIMITER ',' CSV

-- Create hypertable and populate it
select create_hypertable('history_uint', 'clock', chunk_time_interval => 86400, migrate_data => true);
INSERT INTO history_uint SELECT * FROM temp_history_uint ON CONFLICT (itemid,clock,ns) DO NOTHING;

-- Enable compression
select set_integer_now_func('history_uint', 'zbx_ts_unix_now', true);
alter table history_uint set (timescaledb.compress,timescaledb.compress_segmentby='itemid',timescaledb.compress_orderby='clock,ns');

-- Job id will returned, it should be passed to run_job
select add_compress_chunks_policy('history_uint', (select (p.older_than).integer_interval from _timescaledb_config.bgw_policy_compress_chunks p inner join _timescaledb_catalog.hypertable h on (h.id=p.hypertable_id) where h.table_name='history_uint' )::integer);

select alter_job((select job_id from timescaledb_information.jobs where hypertable_schema='public' and hypertable_name='history_uint'), scheduled => true);

-- Run compression job
call run_job(<JOB_ID>);
-- May show 'NOTICE: no chunks for hypertable public.history_uint that satisfy compress chunk policy', it

• Verify that everything works as supposed
• Drop old tables

DROP TABLE history_old; DROP TABLE history_uint_old; DROP TABLE history_str_old; DROP TABLE history_log_old; DROP TABLE history_text_old;

See also: Tips for improving PostgreSQL insert performance

TimescaleDB v2.x
Export and import must be performed in tmux/screen, so that the session isn’t dropped.

See also: Important notes

Upgrading tables

• Rename tables using history_pk_prepare.sql.
sudo -u zabbix psql zabbix < /usr/share/doc/zabbix-sql-scripts/postgresql/history_pk_prepare.sql

• Example of upgrading for one table:

-- Verify that there is enough space to allow export of uncompressed data
select sum(before_compression_total_bytes)/1024/1024 as before_compression_total_mbytes, sum(after_compression_total_bytes)/1024/1024 as after_compression_total_mbytes FROM chunk_compression_stats('history_uint_old');

-- Export data
\copy (select * from history_uint_old) TO '/tmp/history_uint.csv' DELIMITER ',' CSV

CREATE TEMP TABLE temp_history_uint (itemid bigint NOT NULL,
clock integer DEFAULT '0' NOT NULL,
value numeric(20) DEFAULT '0' NOT NULL,
ns integer DEFAULT '0' NOT NULL);
-- Import data
\copy temp_history_uint FROM '/tmp/history_uint.csv' DELIMITER ',' CSV

-- Create hypertable and populate it
select create_hypertable('history_uint', 'clock', chunk_time_interval => 86400, migrate_data => true);
INSERT INTO history_uint SELECT * FROM temp_history_uint ON CONFLICT (itemid,clock,ns) DO NOTHING;

-- Enable compression
select set_integer_now_func('history_uint', 'zbx_ts_unix_now', true);
alter table history_uint set (timescaledb.compress,timescaledb.compress_segmentby='itemid',timescaledb.compress_orderby='clock,ns');
-- Substitute your schema in hypertable_schema
-- Job id will returned, it should be passed to run_job
select add_compression_policy('history_uint', ( 
    select extract(epoch from (config::json->>'compress_after')::interval) from timescaledb_information.jobs where application_name like 'Compression%%' and hypertable_schema='public' and hypertable_name='history_uint_old' 
) ::integer 
);

select alter_job((select job_id from timescaledb_information.jobs where hypertable_schema='public' and hypertable_name='history_uint'), scheduled => true);

-- Run compression job
call run_job(<JOB_ID>);
-- May show 'NOTICE: no chunks for hypertable public.history_uint that satisfy compress chunk policy', it is fine.

• Verify that everything works as supposed
• Drop old tables
  DROP TABLE history_old; DROP TABLE history_uint_old; DROP TABLE history_str_old; DROP TABLE history_log_old; DROP TABLE history_text_old;

See also: Tips for improving PostgreSQL insert performance

Oracle
Export and import must be performed in tmux/screen, so that the session isn’t dropped.

See also: Important notes

Importing/exporting history tables in one attempt
• Install Oracle Data Pump (Instant Client Tools package contains it).

Additionally, consider performance tips for Oracle Data Pump.

• Rename tables using history_pk_prepare.sql.
  shell> cd /path/to/zabbix-sources/database/oracle
  shell> sqlplus zabbix/password@oracle_host/ORCL
  sqlplus> @history_pk_prepare.sql
  • Prepare directories for datapump

Example:

#### mkdir -pv /export/history
#### chown -R oracle:oracle /export
  • Create a directory object, grant permissions to it. Run the following under sysdba role:

create directory history as '/export/history';
grant read,write on directory history to zabbix;

• Export tables. Replace N with your desired thread count.

expdp zabbix/password@172.20.0.3:1521/z \
  DIRECTORY=history \
  TABLES=history_old,history_uint_old,history_str_old,history_log_old,history_text_old \
  PARALLEL=N
  • Import tables. Replace N with your desired thread count.

impdp zabbix/password@172.20.0.3:1521/z \
  DIRECTORY=history \
  TABLES=history_uint_old \
  _REMAP_TABLE=history_old:history,history_uint_old:history_uint,history_str_old:history_str,history_log_old:history_log,history_text_old:history_text \
  data_options=SKIP_CONSTRAINT_ERRORS table_exists_action=APPEND PARALLEL=N CONTENT=data_only
  • Verify that everything works as supposed
• Drop old tables
  DROP TABLE history_old; DROP TABLE history_uint_old; DROP TABLE history_str_old; DROP TABLE history_log_old; DROP TABLE history_text_old;
Importing/exporting history tables individually

- Install Oracle Data Pump *(Instant Client Tools package contains it)*.

Additionally, consider performance tips for Oracle Data Pump.

- Rename tables using *history_pk_prepare.sql*.

  - `shell> cd /path/to/zabbix-sources/database/oracle
    shell> sqlplus zabbix/password@oracle_host/ORCL
    sqlplus> @history_pk_prepare.sql`

- Prepare directories for datapump

  Example:

  ```
  # mkdir -pv /export/history /export/history_uint /export/history_str /export/history_log /export/history_text
  # chown -R oracle:oracle /export
  ```

- Create a directory object, grant permissions to it. Run the following under sysdba role:

  ```
  create directory history as '/export/history';
  grant read,write on directory history to zabbix;
  
  create directory history_uint as '/export/history_uint';
  grant read,write on directory history_uint to zabbix;
  
  create directory history_str as '/export/history_str';
  grant read,write on directory history_str to zabbix;
  
  create directory history_log as '/export/history_log';
  grant read,write on directory history_log to zabbix;
  
  create directory history_text as '/export/history_text';
  grant read,write on directory history_text to zabbix;
  ```

- Export and import each table. Replace `N` with your desired thread count.

  ```
  expdp zabbix/password@172.20.0.3:1521/xe DIRECTORY=history TABLES=history_old PARALLEL=N
  impdp zabbix/password@172.20.0.3:1521/xe DIRECTORY=history TABLES=history_old REMAP_TABLE=history_old:history data_options=SKIP_CONSTRAINT_ERRORS table_exists_action=APPEND PARALLEL=N CONTENT=data_only
  expdp zabbix/password@172.20.0.3:1521/xe DIRECTORY=history_uint TABLES=history_uint_old PARALLEL=N
  impdp zabbix/password@172.20.0.3:1521/xe DIRECTORY=history_uint TABLES=history_uint_old REMAP_TABLE=history_uint_old:history_uint data_options=SKIP_CONSTRAINT_ERRORS table_exists_action=APPEND PARALLEL=N CONTENT=data_only
  expdp zabbix/password@172.20.0.3:1521/xe DIRECTORY=history_str TABLES=history_str_old PARALLEL=N
  impdp zabbix/password@172.20.0.3:1521/xe DIRECTORY=history_str TABLES=history_str_old REMAP_TABLE=history_str_old:history_str data_options=SKIP_CONSTRAINT_ERRORS table_exists_action=APPEND PARALLEL=N CONTENT=data_only
  expdp zabbix/password@172.20.0.3:1521/xe DIRECTORY=history_log TABLES=history_log_old PARALLEL=N
  impdp zabbix/password@172.20.0.3:1521/xe DIRECTORY=history_log TABLES=history_log_old REMAP_TABLE=history_log_old:history_log data_options=SKIP_CONSTRAINT_ERRORS table_exists_action=APPEND PARALLEL=N CONTENT=data_only
  expdp zabbix/password@172.20.0.3:1521/xe DIRECTORY=history_text TABLES=history_text_old PARALLEL=N
  impdp zabbix/password@172.20.0.3:1521/xe DIRECTORY=history_text TABLES=history_text_old REMAP_TABLE=history_text_old:history_text data_options=SKIP_CONSTRAINT_ERRORS table_exists_action=APPEND PARALLEL=N CONTENT=data_only
  ```

- Verify that everything works as supposed

- Drop old tables

  ```
  DROP TABLE history_old;
  DROP TABLE history_uint_old;
  DROP TABLE history_str_old;
  DROP TABLE history_log_old;
  DROP TABLE history_text_old;
  ```

4 Secure connection to the database

Overview

This section provides Zabbix setup steps and configuration examples for secure TLS connections between:
<table>
<thead>
<tr>
<th>Database</th>
<th>Zabbix components</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>Zabbix frontend, Zabbix server, Zabbix proxy</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>Zabbix frontend, Zabbix server, Zabbix proxy</td>
</tr>
</tbody>
</table>

To set up connection encryption within the DBMS, see official vendor documentation for details:

- **MySQL**: source and replica replication database servers.
- **MySQL**: group replication, etc. database servers.
- **PostgreSQL** encryption options.

All examples are based on the GA releases of MySQL CE (8.0) and PgSQL (13) available through official repositories using CentOS 8.

Requirements

The following is required to set up encryption:

- Developer-supported operating system with OpenSSL >=1.1.X or alternative.

It is recommended to avoid OS in the end-of-life status, especially in the case of new installations

- Database engine (RDBMS) installed and maintained from the official repository provided by developer. Operating systems often shipped with outdated database software versions for which encryption support is not implemented, for example RHEL 7 based systems and PostgreSQL 9.2, MariaDB 5.5 without encryption support.

Terminology

Setting this option enforces to use TLS connection to database from Zabbix server/proxy and frontend to database:

- required - connect using TLS as transport mode without identity checks;
- verify_ca - connect using TLS and verify certificate;
- verify_full - connect using TLS, verify certificate and verify that database identity (CN) specified by DBHost matches its certificate;

Zabbix configuration

Frontend to the database

A secure connection to the database can be configured during frontend installation:

- Mark the Database TLS encryption checkbox in the Configure DB connection step to enable transport encryption.
- Mark the Verify database certificate checkbox that appears when TLS encryption field is checked to enable encryption with certificates.

For MySQL, the Database TLS encryption checkbox is disabled, if Database host is set to localhost, because connection that uses a socket file (on Unix) or shared memory (on Windows) cannot be encrypted.

For PostgreSQL, the TLS encryption checkbox is disabled, if the value of the Database host field begins with a slash or the field is empty.

The following parameters become available in the TLS encryption in certificates mode (if both checkboxes are marked):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database TLS CA file</td>
<td>Specify the full path to a valid TLS certificate authority (CA) file.</td>
</tr>
<tr>
<td>Database TLS key file</td>
<td>Specify the full path to a valid TLS key file.</td>
</tr>
<tr>
<td>Database TLS certificate</td>
<td>Specify the full path to a valid TLS certificate file.</td>
</tr>
<tr>
<td>file</td>
<td></td>
</tr>
<tr>
<td>Database host</td>
<td>Mark this checkbox to activate host verification.</td>
</tr>
<tr>
<td>verification</td>
<td>Disabled for MYSQL, because PHP MySQL library does not allow to skip the peer certificate validation step.</td>
</tr>
<tr>
<td>Database TLS cipher list</td>
<td>Specify a custom list of valid ciphers. The format of the cipher list must conform to the OpenSSL standard.</td>
</tr>
<tr>
<td></td>
<td>Available for MySQL only.</td>
</tr>
</tbody>
</table>

TLS parameters must point to valid files. If they point to non-existent or invalid files, it will lead to the authorization error.

If certificate files are writable, the frontend generates a warning in the System information report that “TLS certificate files must be read-only.” (displayed only if the PHP user is the owner of the certificate).

Certificates protected by passwords are not supported.
Use cases

Zabbix frontend uses GUI interface to define possible options: required, verify_ca, verify_full. Specify required options in the installation wizard step Configure DB connections. These options are mapped to the configuration file (zabbix.conf.php) in the following manner:

<table>
<thead>
<tr>
<th>GUI settings</th>
<th>Configuration file</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>// Used for TLS connection. $DB[‘ENCRYPTION’] = true; $DB[‘KEY_FILE’] = “”; $DB[‘CERT_FILE’] = “”; $DB[‘CA_FILE’] = “”; $DB[‘VERIFY_HOST’] = false; $DB[‘CIPHER_LIST’] = “”;</td>
<td>Check Database TLS encryption Leave Verify database certificate unchecked</td>
<td>Enable ‘required’ mode.</td>
<td></td>
</tr>
<tr>
<td>$DB[‘ENCRYPTION’] = true; $DB[‘KEY_FILE’] = “”; $DB[‘CERT_FILE’] = “”; $DB[‘CA_FILE’] = “”; $DB[‘VERIFY_HOST’] = false; $DB[‘CIPHER_LIST’] = “”;</td>
<td>1. Check Database TLS encryption and Verify database certificate 2. Specify path to Database TLS CA file</td>
<td>Enable ‘verify_ca’ mode.</td>
<td></td>
</tr>
</tbody>
</table>
GUI settings | Configuration file | Description | Result
---|---|---|---
... // Used for TLS connection with strictly defined Cipher list. $DB['ENCRYPTION'] = true; $DB['KEY_FILE'] = '<key_file_path>'; $DB['CERT_FILE'] = '<key_file_path>'; $DB['CA_FILE'] = '<key_file_path>'; $DB['VERIFY_HOST'] = true; $DB['CIPHER_LIST'] = '<cipher_list>'; ...

Or:

... // Used for TLS connection without Cipher list defined - selected by MySQL server $DB['ENCRYPTION'] = true; $DB['KEY_FILE'] = '<key_file_path>'; $DB['CERT_FILE'] = '<key_file_path>'; $DB['CA_FILE'] = '<key_file_path>'; $DB['VERIFY_HOST'] = true; $DB['CIPHER_LIST'] = ''; ...

1. Check Database TLS encryption and Verify database certificate
2. Specify path to Database TLS key file
3. Specify path to Database TLS CA file
4. Specify path to Database TLS certificate file
5. Specify TLS cipher list (optional)
6. Specify TLS cipher list (optional)

Enable ‘verify_full’ mode for MySQL.

Enable ‘verify_full’ mode for PostgreSQL.

See also: Encryption configuration examples for MySQL, Encryption configuration examples for PostgreSQL.

Zabbix server/proxy configuration

Secure connections to the database can be configured with the respective parameters in the Zabbix server and/or proxy configuration file.
### 1 MySQL encryption configuration

**Overview**

This section provides several encryption configuration examples for CentOS 8.2 and MySQL 8.0.21 and can be used as a quickstart guide for encrypting the connection to the database.

If MySQL host is set to localhost, encryption options will not be available. In this case a connection between Zabbix frontend and the database uses a socket file (on Unix) or shared memory (on Windows) and cannot be encrypted.

List of encryption combinations is not limited to the ones listed on this page. There are a lot more combinations available.

**Pre-requisites**

Install MySQL database from the official repository.

See MySQL documentation for details on how to use MySQL repo.

MySQL server is ready to accept secure connections using a self-signed certificate.

To see, which users are using an encrypted connection, run the following query (Performance Schema should be turned ON):

```sql
mysql> SELECT sbt.variable_value AS tls_version, t2.variable_value AS cipher, processlist_user AS user, processlist_host AS host
    FROM performance_schema.status_by_thread AS sbt
    JOIN performance_schema.threads AS t ON t.thread_id = sbt.thread_id
    JOIN performance_schema.status_by_thread AS t2 ON t2.thread_id = t.thread_id
WHERE sbt.variable_name = 'Ssl_version' and t2.variable_name = 'Ssl_cipher'
ORDER BY tls_version;
```

**Required mode**

MySQL configuration

Modern versions of the database are ready out-of-the-box for ‘required’ encryption mode. A server-side certificate will be created after initial setup and launch.

Create users and roles for the main components:

```sql
mysql> CREATE USER 'zbx_srv'@'%' IDENTIFIED WITH mysql_native_password BY '<strong_password>',
    'zbx_web'@'%' IDENTIFIED WITH mysql_native_password BY '<strong_password>'
REQUIRE SSL
PASSWORD HISTORY 5;
```

```sql
mysql> CREATE ROLE 'zbx_srv_role', 'zbx_web_role';
```

```sql
mysql> GRANT SELECT, UPDATE, DELETE, INSERT, CREATE, DROP, ALTER, INDEX, REFERENCES ON zabbix.* TO 'zbx_srv_role';
mysql> GRANT SELECT, UPDATE, DELETE, INSERT ON zabbix.* TO 'zbx_web_role';
```
mysql> GRANT 'zbx_srv_role' TO 'zbx_srv'@'%';
mysql> GRANT 'zbx_web_role' TO 'zbx_web'@'%';

mysql> SET DEFAULT ROLE 'zbx_srv_role' TO 'zbx_srv'@'%';
mysql> SET DEFAULT ROLE 'zbx_web_role' TO 'zbx_web'@'%';

Note, that the X.509 protocol is not used to check identity, but the user is configured to use only encrypted connections. See MySQL documentation for more details about configuring users.

Run to check connection (socket connection cannot be used to test secure connections):

```bash
$ mysql -u zbx_srv -p -h 10.211.55.9 --ssl-mode=REQUIRED
```

Check current status and available cipher suites:

```sql
mysql> status
--------------
mysql Ver 8.0.21 for Linux on x86_64 (MySQL Community Server - GPL)
Connection id: 62
Current database: 
Current user: zbx_srv@bfdb.local
SSL: Cipher in use is TLS_AES_256_GCM_SHA384

mysql> SHOW SESSION STATUS LIKE 'Ssl_cipher_list'\G;
*************************** 1. row ***************************
Variable_name: Ssl_cipher_list
1 row in set (0.00 sec)
```

ERROR:
No query specified

Frontend
To enable transport-only encryption for connections between Zabbix frontend and the database:

- Check Database TLS encryption
- Leave Verify database certificate unchecked

Server
To enable transport-only encryption for connections between server and the database, configure /etc/zabbix/zabbix_server.conf:
Verify CA mode

Copy required MySQL CA to the Zabbix frontend server, assign proper permissions to allow the webserver to read this file.

Verify CA mode doesn’t work on SLES 12 and RHEL 7 due to older MySQL libraries.

Frontend

To enable encryption with certificate verification for connections between Zabbix frontend and the database:

- Check Database TLS encryption and Verify database certificate
- Specify path to Database TLS CA file

Alternatively, this can be set in /etc/zabbix/web/zabbix.conf.php:

```php
...$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '';
$DB['CERT_FILE'] = '';
$DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem';
$DB['VERIFY_HOST'] = false;
$DB['CIPHER_LIST'] = '';
...```

Troubleshoot user using command-line tool to check if connection is possible for required user:

```
$ mysql -u zbx_web -p -h 10.211.55.9 --ssl-mode=REQUIRED --ssl-ca=/var/lib/mysql/ca.pem
```

Server

To enable encryption with certificate verification for connections between Zabbix server and the database, configure /etc/zabbix/zabbix_server.conf:

```ini
...DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
```
DBPassword=<strong_password>
DBTLSConnect=verify_ca
DBTLSCAFile=/etc/ssl/mysql/ca.pem
...

Verify Full mode

MySQL configuration

Set MySQL CE server configuration option (/etc/my.cnf.d/server-tls.cnf) to:

```
[mysqld]
...
# in this examples keys are located in the MySQL CE datadir directory
ssl_ca=ca.pem
ssl_cert=server-cert.pem
ssl_key=server-key.pem
```

```
require_secure_transport=ON
tls_version=TLSv1.3
...
```

Keys for the MySQL CE server and client (Zabbix frontend) should be created manually according to the MySQL CE documentation:

- Creating SSL and RSA certificates and keys using MySQL
- Creating SSL certificates and keys using openssl

MySQL server certificate should contain the Common Name field set to the FQDN name as Zabbix frontend will use the DNS name to communicate with the database or IP address of the database host.

Create MySQL user:

```
mysql> CREATE USER
    
  'zbx_srv'@'%' IDENTIFIED WITH mysql_native_password BY '<strong_password>',
  'zbx_web'@'%' IDENTIFIED WITH mysql_native_password BY '<strong_password>'
    
  REQUIRE X509
  PASSWORD HISTORY 5;
```

Check if it is possible to log in with that user:

```
$ mysql -u zbx_web -p -h 10.211.55.9 --ssl-mode=VERIFY_IDENTITY --ssl-ca=/var/lib/mysql/ca.pem --ssl-cert=/var/lib/mysql/client-cert.pem --ssl-key=/var/lib/mysql/client-key.pem
```

To enable encryption with full verification for connections between Zabbix frontend and the database:

- Check Database TLS encryption and Verify database certificate
- Specify path to Database TLS key file
- Specify path to Database TLS CA file
- Specify path to Database TLS certificate file

Note, that Database host verification is checked and grayed out - this step cannot be skipped for MySQL.

Cipher list should be empty, so that frontend and server can negotiate required one from the supported by both ends.
Alternatively, this can be set in /etc/zabbix/web/zabbix.conf.php:

```php
// Used for TLS connection with strictly defined Cipher list.
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '/etc/ssl/mysql/client-key.pem';
$DB['CERT_FILE'] = '/etc/ssl/mysql/client-cert.pem';
$DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem';
$DB['VERIFY_HOST'] = true;
$DB['CIPHER_LIST'] = 'TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256';

// or

// Used for TLS connection without Cipher list defined - selected by MySQL server
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '/etc/ssl/mysql/client-key.pem';
$DB['CERT_FILE'] = '/etc/ssl/mysql/client-cert.pem';
$DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem';
$DB['VERIFY_HOST'] = true;
$DB['CIPHER_LIST'] = '';

Server

To enable encryption with full verification for connections between Zabbix server and the database, configure /etc/zabbix/zabbix_server.conf:

```conf
[db]
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=verify_full
DBTLSCertFile=/etc/ssl/mysql/ca.pem
DBTLSKeyFile=/etc/ssl/mysql/client-key.pem

2 PostgreSQL encryption configuration

Overview

This section provides several encryption configuration examples for CentOS 8.2 and PostgreSQL 13.
Connection between Zabbix frontend and PostgreSQL cannot be encrypted (parameters in GUI are disabled), if the value of Database host field begins with a slash or the field is empty.

Pre-requisites
Install the PostgreSQL database using the official repository.

PostgreSQL is not configured to accept TLS connections out-of-the-box. Please follow instructions from PostgreSQL documentation for certificate preparation with postgresql.conf and also for user access control through ph_hba.conf.

By default, the PostgreSQL socket is binded to the localhost, for the network remote connections allow to listen on the real network interface.

PostgreSQL settings for all modes can look like this:

/var/lib/pgsql/13/data/postgresql.conf:

```plaintext
...  
ssl = on  
ssl_ca_file = 'root.crt'  
ssl_cert_file = 'server.crt'  
ssl_key_file = 'server.key'  
ssl_ciphers = 'HIGH: MEDIUM:+3DES:!aNULL'  
ssl_prefer_server_ciphers = on  
ssl_min_protocol_version = 'TLSv1.3'  
  ...
```

For access control adjust /var/lib/pgsql/13/data/pg_hba.conf:

```plaintext
...  
### require  
hostssl all all 0.0.0.0/0 md5  
### verify CA  
hostssl all all 0.0.0.0/0 md5 clientcert=verify-ca  
### verify full  
hostssl all all 0.0.0.0/0 md5 clientcert=verify-full  
  ...
```

Required mode

Frontend

To enable transport-only encryption for connections between Zabbix frontend and the database:

- Check Database TLS encryption
- Leave Verify database certificate unchecked
To enable transport-only encryption for connections between server and the database, configure /etc/zabbix/zabbix_server.conf:

- DBHost=10.211.55.9
- DBName=zabbix
- DBUser=zbx_srv
- DBPassword=<strong_password>
- DBTLSConnect=required

To enable encryption with certificate authority verification for connections between Zabbix frontend and the database:

- Check Database TLS encryption and Verify database certificate
- Specify path to Database TLS key file
- Specify path to Database TLS CA file
- Specify path to Database TLS certificate file
Alternatively, this can be set in `/etc/zabbix/web/zabbix.conf.php`:

```php
...
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = ''; 
$DB['CERT_FILE'] = ''; 
$DB['CA_FILE'] = '/etc/ssl/pgsql/root.crt'; 
$DB['VERIFY_HOST'] = false;
$DB['CIPHER_LIST'] = ''; 
...
```

### Server

To enable encryption with certificate verification for connections between Zabbix server and the database, configure `/etc/zabbix/zabbix_server.conf`:

```ini
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=*strong_password*
DBTLSConnect=verify_ca
DBTLSAFile=/etc/ssl/pgsql/root.crt
...
```

### Frontend

To enable encryption with certificate and database host identity verification for connections between Zabbix frontend and the database:

- Check Database TLS encryption and Verify database certificate
- Specify path to Database TLS key file
- Specify path to Database TLS CA file
- Specify path to Database TLS certificate file
- Check Database host verification
Alternatively, this can be set in `/etc/zabbix/web/zabbix.conf.php`:

```
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '';
$DB['CERT_FILE'] = '';
$DB['CA_FILE'] = '/etc/ssl/pgsql/root.crt';
$DB['VERIFY_HOST'] = true;
$DB['CIPHER_LIST'] = '';
...
```

Server

To enable encryption with certificate and database host identity verification for connections between Zabbix server and the database, configure `/etc/zabbix/zabbix_server.conf`:

```
... 
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=verify_full
DBTLSCAFFile=/etc/ssl/pgsql/root.crt
DBTLSCertFile=/etc/ssl/pgsql/client.crt
DBTLSKeyFile=/etc/ssl/pgsql/client.key
...
```

5 TimescaleDB setup

Overview

Zabbix supports TimescaleDB, a PostgreSQL-based database solution of automatically partitioning data into time-based chunks to support faster performance at scale.

Currently TimescaleDB is not supported by Zabbix proxy.

Instructions on this page can be used for creating TimescaleDB database or migrating from existing PostgreSQL tables to TimescaleDB.

Configuration

We assume that TimescaleDB extension has been already installed on the database server (see installation instructions). TimescaleDB extension must also be enabled for the specific DB by executing:

```
echo "CREATE EXTENSION IF NOT EXISTS timescaledb CASCADE;" | sudo -u postgres psql zabbix
```
Running this command requires database administrator privileges.

If you use a database schema other than ‘public’ you need to add a SCHEMA clause to the command above. E.g.:

```
echo "CREATE EXTENSION IF NOT EXISTS timescaledb SCHEMA yourschema CASCADE;" | sudo -u postgres psql
```

Then run the `timescaledb.sql` script located in `database/postgresql`. For new installations the script must be run after the regular PostgreSQL database has been created with initial schema/data (see [database creation](#)):

```
cat /usr/share/doc/zabbix-sql-scripts/postgresql/timescaledb.sql | sudo -u zabbix psql zabbix
```

The migration of existing history and trend data may take a lot of time. Zabbix server and frontend must be down for the period of migration.

The `timescaledb.sql` script sets the following housekeeping parameters:

- Override item history period
- Override item trend period

In order to use partitioned housekeeping for history and trends, both these options must be on. It’s possible to use TimescaleDB partitioning only for trends (by setting Override item trend period) or only for history (Override item history period).

For PostgreSQL version 10.2 or higher and TimescaleDB version 1.5 or higher, the `timescaledb.sql` script sets two additional parameters:

- Enable compression
- Compress records older than 7 days

All of these parameters can be changed in Administration → General → Housekeeping after the installation.

You may want to run the `timescaledb-tune` tool provided by TimescaleDB to optimize PostgreSQL configuration parameters in your `postgresql.conf`.

TimescaleDB compression

Native TimescaleDB compression is supported starting from Zabbix 5.0 for PostgreSQL version 10.2 or higher and TimescaleDB version 1.5 or higher for all Zabbix tables that are managed by TimescaleDB. During the upgrade or migration to TimescaleDB, initial compression of the large tables may take a lot of time.

Note that compression is supported under the “timescale” Timescale Community license and it is not supported under “apache” Apache 2.0 license. Starting with Zabbix 6.2.1, Zabbix detects if compression is supported. If it is not supported a warning message is written into the Zabbix server log and users cannot enable compression in the frontend.

Users are encouraged to get familiar with TimescaleDB compression documentation before using compression.

Note, that there are certain limitations imposed by compression, specifically:

- Compressed chunk modifications (inserts, deletes, updates) are not allowed
- Schema changes for compressed tables are not allowed.

Compression settings can be changed in the History and trends compression block in Administration → General → Housekeeping section of Zabbix frontend.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable compression</td>
<td>Enabled</td>
<td>Checking or unchecking the checkbox does not activate/deactivate compression immediately. Because compression is handled by the Housekeeper, the changes will take effect in up to 2 times HousekeepingFrequency hours (set in <code>zabbix_server.conf</code>). After disabling compression, new chunks that fall into the compression period will not be compressed. However, all previously compressed data will stay compressed. To uncompress previously compressed chunks, follow instructions in TimescaleDB documentation. When upgrading from older versions of Zabbix with TimescaleDB support, compression will not be enabled by default. This parameter cannot be less than 7 days. Due to immutability of compressed chunks all late data (e.g. data delayed by a proxy) that is older than this value will be discarded.</td>
</tr>
<tr>
<td>Compress records older than</td>
<td>7d</td>
<td></td>
</tr>
</tbody>
</table>
6 Elasticsearch setup

Elasticsearch support is experimental!

Zabbix supports the storage of historical data by means of Elasticsearch instead of a database. Users can choose the storage place for historical data between a compatible database and Elasticsearch. The setup procedure described in this section is applicable to Elasticsearch version 7.X. In case an earlier or later version of Elasticsearch is used, some functionality may not work as intended.

If all history data is stored in Elasticsearch, trends are not calculated nor stored in the database. With no trends calculated and stored, the history storage period may need to be extended.

Configuration

To ensure proper communication between all elements involved make sure server configuration file and frontend configuration file parameters are properly configured.

Zabbix server and frontend

Zabbix server configuration file draft with parameters to be updated:

```plaintext
### Option: HistoryStorageURL
# History storage HTTP[S] URL.
#
# Mandatory: no
# Default:
# HistoryStorageURL=
### Option: HistoryStorageTypes
# Comma separated list of value types to be sent to the history storage.
#
# Mandatory: no
# Default:
# HistoryStorageTypes=uint,dbl,str,log,text
```

Example parameter values to fill the Zabbix server configuration file with:

```
HistoryStorageURL=http://test.elasticsearch.lan:9200
HistoryStorageTypes=str,log,text
```

This configuration forces Zabbix Server to store history values of numeric types in the corresponding database and textual history data in Elasticsearch.

Elasticsearch supports the following item types:

```
uint,dbl,str,log,text
```

Supported item type explanation:

<table>
<thead>
<tr>
<th>Item value type</th>
<th>Database table</th>
<th>Elasticsearch type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric (unsigned)</td>
<td>history_uint</td>
<td>uint</td>
</tr>
<tr>
<td>Numeric (float)</td>
<td>history</td>
<td>dbl</td>
</tr>
<tr>
<td>Character</td>
<td>history_str</td>
<td>str</td>
</tr>
<tr>
<td>Log</td>
<td>history_log</td>
<td>log</td>
</tr>
<tr>
<td>Text</td>
<td>history_text</td>
<td>text</td>
</tr>
</tbody>
</table>

Zabbix frontend configuration file (conf/zabbix.conf.php) draft with parameters to be updated:

```
// Elasticsearch url (can be string if same url is used for all types).
$HISTORY['url'] = [
    'uint' => 'http://localhost:9200',
    'text' => 'http://localhost:9200'
];
// Value types stored in Elasticsearch.
$HISTORY['types'] = ['uint', 'text'];
```

Example parameter values to fill the Zabbix frontend configuration file with:

```
$HISTORY['url'] = 'http://test.elasticsearch.lan:9200';
$HISTORY['types'] = ['str', 'text', 'log'];
```
This configuration forces to store Text, Character and Log history values in Elasticsearch.

It is also required to make $HISTORY global in conf/zabbix.conf.php to ensure everything is working properly (see conf/zabbix.conf.php.example for how to do it):

```
// Zabbix GUI configuration file.
global $DB, $HISTORY;
```

Installing Elasticsearch and creating mapping

Final two steps of making things work are installing Elasticsearch itself and creating mapping process.

To install Elasticsearch please refer to Elasticsearch installation guide.

Mapping is a data structure in Elasticsearch (similar to a table in a database). Mapping for all history data types is available here: database/elasticsearch/elasticsearch.map.

Creating mapping is mandatory. Some functionality will be broken if mapping is not created according to the instruction.

To create mapping for text type send the following request to Elasticsearch:

```
curl -X PUT http://your-elasticsearch.here:9200/text 
-H 'content-type:application/json' 
-d '{
  "settings": {
    "index": {
      "number_of_replicas": 1,
      "number_of_shards": 5
    }
  },
  "mappings": {
    "properties": {
      "itemid": {
        "type": "long"
      },
      "clock": {
        "format": "epoch_second",
        "type": "date"
      },
      "value": {
        "fields": {
          "analyzed": {
            "index": true,
            "type": "text",
            "analyzer": "standard"
          }
        },
        "index": false,
        "type": "text"
      }
    }
  }
}'
```

Similar request is required to be executed for Character and Log history values mapping creation with corresponding type correction.

To work with Elasticsearch please refer to Requirement page for additional information.

Housekeeper is not deleting any data from Elasticsearch.

Storing history data in multiple date-based indices

This section describes additional steps required to work with pipelines and ingest nodes.

To begin with, you must create templates for indices.

The following example shows a request for creating uint template:
curl -X PUT \
http://your-elasticsearch.here:9200/_template/uint_template \
-H 'content-type:application/json' \
-d '{
  "index_patterns": [
    "uint*"
  ],
  "settings": {
    "index": {
      "number_of_replicas": 1,
      "number_of_shards": 5
    }
  },
  "mappings": {
    "properties": {
      "itemid": {
        "type": "long"
      },
      "clock": {
        "format": "epoch_second",
        "type": "date"
      },
      "value": {
        "type": "long"
      }
    }
  }
}'

To create other templates, user should change the URL (last part is the name of template), change "index_patterns" field to match index name and to set valid mapping, which can be taken from database/elasticsearch/elasticsearch.map.

For example, the following command can be used to create a template for text index:

curl -X PUT \
http://your-elasticsearch.here:9200/_template/text_template \
-H 'content-type:application/json' \
-d '{
  "index_patterns": [
    "text*"
  ],
  "settings": {
    "index": {
      "number_of_replicas": 1,
      "number_of_shards": 5
    }
  },
  "mappings": {
    "properties": {
      "itemid": {
        "type": "long"
      },
      "clock": {
        "format": "epoch_second",
        "type": "date"
      },
      "value": {
        "fields": {
          "analyzed": {
            "index": true,
            "type": "text",
            "analyzer": "standard"
          }
        }
      }
    }
  }
}'
This is required to allow Elasticsearch to set valid mapping for indices created automatically. Then it is required to create the pipeline definition. Pipeline is some sort of preprocessing of data before putting data in indices. The following command can be used to create pipeline for uint index:

```bash
  "description": "daily uint index naming",
  "processors": [
    {
      "date_index_name": {
        "field": "clock",
        "date_formats": ["UNIX"],
        "index_name_prefix": "uint-",
        "date_rounding": "d"
      }
    }
  ]
}
```

User can change the rounding parameter ("date_rounding") to set a specific index rotation period. To create other pipelines, user should change the URL (last part is the name of pipeline) and change "index_name_prefix" field to match index name.

See also Elasticsearch documentation.

Additionally, storing history data in multiple date-based indices should also be enabled in the new parameter in Zabbix server configuration:

```plaintext
### Option: HistoryStorageDateIndex
# Enable preprocessing of history values in history storage to store values in different indices based on date.
# 0 - disable
# 1 - enable
# Mandatory: no
# Default: HistoryStorageDateIndex=0
```

Troubleshooting

The following steps may help you troubleshoot problems with Elasticsearch setup:

1. Check if the mapping is correct (GET request to required index URL like http://localhost:9200/uint).
2. Check if shards are not in failed state (restart of Elasticsearch should help).
3. Check the configuration of Elasticsearch. Configuration should allow access from the Zabbix frontend host and the Zabbix server host.
4. Check Elasticsearch logs.

If you are still experiencing problems with your installation then please create a bug report with all the information from this list (mapping, error logs, configuration, version, etc.)

### 7 Real-time export of events, item values, trends

Overview

It is possible to configure real-time exporting of trigger events, item values and trends in a newline-delimited JSON format. Exporting is done into files, where each line of the export file is a JSON object. Value mappings are not applied.
In case of errors (data cannot be written to the export file or the export file cannot be renamed or a new one cannot be created after renaming it), the data item is dropped and never written to the export file. It is written only in the Zabbix database. Writing data to the export file is resumed when the writing problem is resolved.

For precise details on what information is exported, see the export protocol page.

Note that host/item can have no metadata (host groups, host name, item name) if the host/item was removed after the data was received, but before server exported data.

Configuration

Real-time export of trigger events, item values and trends is configured by specifying a directory for the export files - see the ExportDir parameter in server configuration.

Two other parameters are available:

- ExportFileSize may be used to set the maximum allowed size of an individual export file. When a process needs to write to a file it checks the size of the file first. If it exceeds the configured size limit, the file is renamed by appending .old to its name and a new file with the original name is created.

A file will be created per each process that will write data (i.e. approximately 4-30 files). As the default size per export file is 1G, keeping large export files may drain the disk space fast.

- ExportType allows to specify which entity types (events, history, trends) will be exported.

8 Distribution-specific notes on setting up Nginx for Zabbix

RHEL

Nginx is available only in EPEL:

# yum -y install epel-release

SLES 12

In SUSE Linux Enterprise Server 12 you need to add the Nginx repository, before installing Nginx:

zypper addrepo -G -t yum -c 'http://nginx.org/packages/sles/12' nginx

You also need to configure php-fpm:

cp /etc/php5/fpm/php-fpm.conf{.default,}
sed -i 's/user = nobody/user = wwwrun/; s/group = nobody/group = www/' /etc/php5/fpm/php-fpm.conf

SLES 15

In SUSE Linux Enterprise Server 15 you need to configure php-fpm:

cp /etc/php7/fpm/php-fpm.conf{.default,}
cp /etc/php7/fpm/php-fpm.d/www.conf{.default,}
sed -i 's/user = nobody/user = wwwrun/; s/group = nobody/group = www/' /etc/php7/fpm/php-fpm.d/www.conf

9 Running agent as root

Starting with version 5.0.0 the systemd service file for Zabbix agent in official packages was updated to explicitly include directives for User and Group. Both are set to zabbix.

This means that the old functionality of configuring which user Zabbix agent runs as via zabbix_agentd.conf file is bypassed and agent will always run as the user specified in the systemd service file.

To override this new behavior create a /etc/systemd/system/zabbix-agent.service.d/override.conf file with the following content:

[Service]
User=root
Group=root

Reload daemons and restart the zabbix-agent service:

systemctl daemon-reload
systemctl restart zabbix-agent

1384
For **Zabbix agent 2** this completely determines the user that it runs as.

For old agent this only re-enables the functionality of configuring user in the `zabbix_agentd.conf` file. Therefore in order to run zabbix agent as root you still have to edit the agent configuration file and specify `User=root` as well as `AllowRoot=1` options.

### 10 Zabbix agent on Microsoft Windows

Configuring agent

Both generations of Zabbix agents run as a Windows service. For Zabbix agent 2, replace `agentd` with `agent2` in the instructions below.

You can run a single instance of Zabbix agent or multiple instances of the agent on a Microsoft Windows host. A single instance can use the default configuration file `C:\zabbix_agentd.conf` or a configuration file specified in the command line. In case of multiple instances each agent instance must have its own configuration file (one of the instances can use the default configuration file).

An example configuration file is available in Zabbix source archive as `conf/zabbix_agentd.win.conf`.

See the configuration file options for details on configuring Zabbix Windows agent.

Hostname parameter

To perform active checks on a host Zabbix agent needs to have the hostname defined. Moreover, the hostname value set on the agent side should exactly match the "Host name" configured for the host in the frontend.

The hostname value on the agent side can be defined by either the `Hostname` or `HostnameItem` parameter in the agent configuration file - or the default values are used if any of these parameters are not specified.

The default value for `HostnameItem` parameter is the value returned by the "system.hostname" agent key. For Windows, it returns result of the `gethostname()` function, which queries namespace providers to determine the local host name. If no namespace provider responds, the NetBIOS name is returned.

The default value for `Hostname` is the value returned by the `HostnameItem` parameter. So, in effect, if both these parameters are unspecified the actual hostname will be the host NetBIOS name; Zabbix agent will use NetBIOS host name to retrieve the list of active checks from Zabbix server and send results to it.

The default value for `Hostname` is the value returned by the `HostnameItem` parameter. So, in effect, if both these parameters are unspecified the actual hostname will be the host NetBIOS name; Zabbix agent will use NetBIOS host name to retrieve the list of active checks from Zabbix server and send results to it.

The "system.hostname" key supports two optional parameters - type and transform.

Type parameter determines the type of the name the item should return. Supported values:

- netbios (default) - returns the NetBIOS host name which is limited to 15 symbols and is in the UPPERCASE only;
- host - case-sensitive, returns the full, real Windows host name (without a domain);
- shorthost (supported since Zabbix 5.4.7) - returns part of the hostname before the first dot. It will return a full string if the name does not contain a dot.

Transform parameter is supported since Zabbix 5.4.7 and allows to specify additional transformation rule for the hostname. Supported values:

- none (default) - use the original letter case;
- lower - convert the text into lowercase.

So, to simplify the configuration of `zabbix_agentd.conf` file and make it unified, two different approaches could be used.

1. leave `Hostname` or `HostnameItem` parameters undefined and Zabbix agent will use NetBIOS host name as the hostname;
2. leave `Hostname` parameter undefined and define `HostnameItem` like this:
   - `HostnameItem=system.hostname[host]` - for Zabbix agent to use the full, real (case sensitive) Windows host name as the hostname
   - `HostnameItem=system.hostname[shorthost,lower]` - for Zabbix agent to use only part of the hostname before the first dot, converted into lowercase.

Host name is also used as part of Windows service name which is used for installing, starting, stopping and uninstalling the Windows service. For example, if Zabbix agent configuration file specifies `Hostname=Windows_db_server`, then the agent will be installed as a Windows service "Zabbix Agent  [Windows_db_server]". Therefore, to have a different Windows service name for each Zabbix agent instance, each instance must use a different host name.

Installing agent as Windows service

To install a single instance of Zabbix agent with the default configuration file `c:\zabbix_agentd.conf`:
zabbix_agentd.exe --install

On a 64-bit system, a 64-bit Zabbix agent version is required for all checks related to running 64-bit processes to work correctly.

If you wish to use a configuration file other than c:\zabbix_agentd.conf, you should use the following command for service installation:

```
zabbix_agentd.exe --config <your_configuration_file> --install
```

A full path to the configuration file should be specified.

Multiple instances of Zabbix agent can be installed as services like this:

```
zabbix_agentd.exe --config <configuration_file_for_instance_1> --install --multiple-agents
zabbix_agentd.exe --config <configuration_file_for_instance_2> --install --multiple-agents
...
zabbix_agentd.exe --config <configuration_file_for_instance_N> --install --multiple-agents
```

The installed service should now be visible in Control Panel.

Starting agent

To start the agent service, you can use Control Panel or do it from command line.

To start a single instance of Zabbix agent with the default configuration file:

```
zabbix_agentd.exe --start
```

To start a single instance of Zabbix agent with another configuration file:

```
zabbix_agentd.exe --config <your_configuration_file> --start
```

To start one of multiple instances of Zabbix agent:

```
zabbix_agentd.exe --config <configuration_file_for_this_instance> --start --multiple-agents
```

Stopping agent

To stop the agent service, you can use Control Panel or do it from command line.

To stop a single instance of Zabbix agent started with the default configuration file:

```
zabbix_agentd.exe --stop
```

To stop a single instance of Zabbix agent started with another configuration file:

```
zabbix_agentd.exe --config <your_configuration_file> --stop
```

To stop one of multiple instances of Zabbix agent:

```
zabbix_agentd.exe --config <configuration_file_for_this_instance> --stop --multiple-agents
```

Uninstalling agent Windows service

To uninstall a single instance of Zabbix agent using the default configuration file:

```
zabbix_agentd.exe --uninstall
```

To uninstall a single instance of Zabbix agent using a non-default configuration file:

```
zabbix_agentd.exe --config <your_configuration_file> --uninstall
```

To uninstall multiple instances of Zabbix agent from Windows services:

```
zabbix_agentd.exe --config <configuration_file_for_instance_1> --uninstall --multiple-agents
zabbix_agentd.exe --config <configuration_file_for_instance_2> --uninstall --multiple-agents
...
zabbix_agentd.exe --config <configuration_file_for_instance_N> --uninstall --multiple-agents
```

Limitations

Zabbix agent for Windows does not support non-standard Windows configurations where CPUs are distributed non-uniformly across NUMA nodes. If logical CPUs are distributed non-uniformly, then CPU performance metrics may not be available for some CPUs. For example, if there are 72 logical CPUs with 2 NUMA nodes, both nodes must have 36 CPUs each.
11 SAML setup with Okta

This section describes how to configure Okta to enable SAML 2.0 authentication for Zabbix.

Okta configuration

1. Go to https://okta.com and register or sign in to your account.

2. In the Okta web interface navigate to Applications → Applications and press “Add Application” button (Add Application).


4. Fill in the fields in the General settings tab (the first tab that appears) according to your preferences and press “Next”.

5. In the Configure SAML tab enter the values provided below, then press "Next".

   - In the **GENERAL** section:
     - Single sign on URL: https://<your-zabbix-url>/ui/index_sso.php?acs
       The checkbox Use this for Recipient URL and Destination URL should be marked)
     - Audience URI (SP Entity ID): zabbix
       Note, that this value will be used within the SAML assertion as a unique service provider identifier (if not matching, the operation will be rejected). It is possible to specify a URL or any string of data in this field.
     - Default RelayState:
       Leave this field blank; if a custom redirect is required, it can be added in Zabbix in the Administration → Users settings.
     - Fill in other fields according to your preferences.
If planning to use encrypted connection, generate private and public encryption certificates, then upload public certificate to Okta. Certificate upload form appears when Assertion Encryption is set to Encrypted (click Show Advanced Settings to find this parameter).

- In the **ATTRIBUTE STATEMENTS (OPTIONAL)** section add an attribute statement with:
  - Name: usrEmail
  - Name format: Unspecified
  - Value: user.email

6. At the next tab, select “I’m a software vendor. I’d like to integrate my app with Okta” and press “Finish”.

7. Now, navigate to Assignments tab and press the "Assign" button, then select Assign to People from the drop-down.
8. In a popup that appears, assign created app to people that will use SAML 2.0 to authenticate with Zabbix, then press “Save and go back”.

9. Navigate to the Sign On tab and press the ”View Setup Instructions” button. Setup instructions will be displayed in a new tab; keep this tab open while configuring Zabbix.

Zabbix configuration

1. In Zabbix, go to SAML settings in the Administration → Authentication section and copy information from Okta setup instructions into corresponding fields:
   - Identity Provider Single Sign-On URL → SSO service URL
   - Identity Provider Issuer → IdP entity ID
   - Username attribute → Attribute name (usrEmail)
   - SP entity ID → Audience URI

2. Download the certificate provided in the Okta setup instructions page into ui/conf/certs folder as idp.crt, and set permission 644 by running:
chmod 644 idp.crt

Note, that if you have upgraded to Zabbix 5.0 from an older version, you will also need to manually add these lines to zabbix.conf.php file (located in the //ui/conf// directory):

```php
// Used for SAML authentication.
$SSO['SP_KEY'] = 'conf/certs/sp.key'; // Path to your private key.
$SSO['SP_CERT'] = 'conf/certs/sp.crt'; // Path to your public key.
$SSO['IDP_CERT'] = 'conf/certs/idp.crt'; // Path to IdP public key.
$SSO['SETTINGS'] = [];
```

See generic SAML Authentication instructions for more details.

3. If Assertion Encryption has been set to Encrypted in Okta, a checkbox “Assertions” of the Encrypt parameter should be marked in Zabbix as well.

4. Press the “Update” button to save these settings.

To sign in with SAML, the username in Zabbix should match the Okta e-mail. These settings can be changed in the Administration → Users section of Zabbix web interface.

12 Oracle database setup

Overview

This section contains instructions for creating Oracle database and configuring connections between the database and Zabbix server, proxy, and frontend.

Database creation

We assume that a zabbix database user with password password exists and has permissions to create database objects in ORCL service located on the host Oracle database server. Zabbix requires a Unicode database character set and a UTF8 national character set. Check current settings:

```
sqlplus> select parameter,value from v$nls_parameters where parameter='NLS_CHARACTERSET' or parameter='NLS_NCHAR_CHARACTERSET'
```

43x770

SAMLAuthentication

Now prepare the database:

```
shell> cd /path/to/zabbix-sources/database/oracle
shell> sqlplus zabbix/password@oracle_host/ORCL
sqlplus> @schema.sql
# stop here if you are creating database for Zabbix proxy
sqlplus> @images.sql
sqlplus> @data.sql
```

Please set the initialization parameter CURSOR_SHARING=FORCE for best performance.

Connection set up

Zabbix supports two types of connect identifiers (connection methods):

- Easy Connect
- Net Service Name

Connection configuration parameters for Zabbix server and Zabbix proxy can be set in the configuration files. Important parameters for the server and proxy are DBHost, DBUser, DBName and DBPassword. The same parameters are important for the frontend: $DB["SERVER"], $DB["PORT"], $DB["DATABASE"], $DB["USER"], $DB["PASSWORD"].

Zabbix uses the following connection string syntax:

```
{DBUser/DBPassword[@<connect_identifier>]}
```

<connect_identifier> can be specified either in the form of "Net Service Name" or "Easy Connect".

```
@[[//]Host[:Port]/<service_name> | <net_service_name>]
```

**Easy Connect**

Easy Connect uses the following parameters to connect to the database:

- Host - the host name or IP address of the database server computer (DBHost parameter in the configuration file).
- Port - the listening port on the database server (DBPort parameter in the configuration file; if not set the default 1521 port will be used).
- <service_name> - the service name of the database you want to access (DBName parameter in the configuration file).

**Example:**

Database parameters set in the server or proxy configuration file (zabbix_server.conf and zabbix_proxy.conf):

```
DBHost=localhost
DBPort=1521
DBUser=myusername
DBName=ORCL
DBPassword=mypassword
```

Connection string used by Zabbix to establish connection:

```
DBUser/DBPassword@DBHost:DBPort/DBName
```

During Zabbix frontend installation, set the corresponding parameters in the Configure DB connection step of the setup wizard:

- Database host: localhost
- Database port: 1521
- Database name: ORCL
- User: myusername
- Password: mypassword
Alternatively, these parameters can be set in the frontend configuration file (zabbix.conf.php):

```
$DB["TYPE"] = 'ORACLE';
$DB["SERVER"] = 'localhost';
$DB["PORT"] = '1521';
$DB["DATABASE"] = 'ORCL';
$DB["USER"] = 'myusername';
$DB["PASSWORD"] = 'mypassword';
```

Net service name

Since Zabbix 5.4.0 it is possible to connect to Oracle by using net service name.

<net_service_name> is a simple name for a service that resolves to a connect descriptor.

In order to use the service name for creating a connection, this service name has to be defined in the tnsnames.ora file located on both the database server and the client systems. The easiest way to make sure that the connection will succeed is to define the location of tnsnames.ora file in the TNS_ADMIN environment variable. The default location of the tnsnames.ora file is:

```
$ORACLE_HOME/network/admin/
```

A simple tnsnames.ora file example:

```
ORCL =
 (DESCRIPTION =
 (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
 (CONNECT_DATA =
 (SERVER = DEDICATED)
 (SERVICE_NAME = ORCL)
 )
)
```

To set configuration parameters for the "Net Service Name" connection method, use one of the following options:

- Set an empty parameter DBHost and set DBName as usual:
  ```
  DBHost=
  DBName=ORCL
  ```
- Set both parameters and leave both empty:
  ```
  DBHost=
  DBName=
  ```

In the second case, the TWO_TAKS environment variable has to be set. It specifies the default remote Oracle service (service name). When this variable is defined, the connector connects to the specified database by using an Oracle listener that accepts connection requests. This variable is for use on Linux and UNIX only. Use the LOCAL environment variable for Microsoft Windows.

**Example:**

Connect to a database using Net Service Name set as ORCL and the default port. Database parameters set in the server or proxy configuration file (zabbix_server.conf and zabbix_proxy.conf):
During Zabbix frontend installation, set the corresponding parameters in the Configure DB connection step of the setup wizard:

- Database host:
- Database port: 0
- Database name: ORCL
- User: myusername
- Password: mypassword

Alternatively, these parameters can be set in the frontend configuration file (zabbix.conf.php):

```php
$DB["TYPE"] = 'ORACLE';
$DB["SERVER"] = '';
$DB["PORT"] = '0';
$DB["DATABASE"] = 'ORCL';
$DB["USER"] = 'myusername';
$DB["PASSWORD"] = 'mypassword';
```

Connection string used by Zabbix to establish connection:

```
DBUser/DBPassword@ORCL
```

13 Setting up scheduled reports

Overview

This section provides instructions on installing Zabbix web service and configuring Zabbix to enable generation of scheduled reports.

Currently the support of scheduled reports is experimental.

Installation

A new Zabbix web service process and Google Chrome browser should be installed to enable generation of scheduled reports. The web service may be installed on the same machine where the Zabbix server is installed or on a different machine. Google Chrome browser should be installed on the same machine, where the web service is installed.

The official zabbix-web-service package is available in the Zabbix repository. Google Chrome browser is not included into these packages and has to be installed separately.

To compile Zabbix web service from sources, see Installing Zabbix web service.

After the installation, run zabbix_web_service on the machine, where the web service is installed:

```
shell> zabbix_web_service
```
Configuration

To ensure proper communication between all elements involved make sure server configuration file and frontend configuration parameters are properly configured.

Zabbix server

The following parameters in Zabbix server configuration file need to be updated: WebServiceURL and StartReportWriters.

**WebServiceURL**

This parameter is required to enable communication with the web service. The URL should be in the format `<host:port>/report`.

- By default, the web service listens on port 10053. A different port can be specified in the web service configuration file.
- Specifying the `/report` path is mandatory (the path is hardcoded and cannot be changed).

Example:

```
WebServiceURL=http://localhost:10053/report
```

**StartReportWriters**

This parameter determines how many report writer processes should be started. If it is not set or equals 0, report generation is disabled. Based on the number and frequency of reports required, it is possible to enable from 1 to 100 report writer processes.

Example:

```
StartReportWriters=3
```

Zabbix frontend

A Frontend URL parameter should be set to enable communication between Zabbix frontend and Zabbix web service:

- Proceed to the Administration → General → Other parameters frontend menu section
- Specify the full URL of the Zabbix web interface in the Frontend URL parameter.

Once the setup procedure is completed, you may want to configure and send a test report to make sure everything works correctly.

**14 Additional frontend languages**

Overview

In order to use any other language than English in Zabbix web interface, its locale should be installed on the web server. Additionally, the PHP gettext extension is required for the translations to work.

Installing locales

To list all installed languages, run:

```
locale -a
```

If some languages that are needed are not listed, open the `/etc/locale.gen` file and uncomment the required locales. Since Zabbix uses UTF-8 encoding, you need to select locales with UTF-8 charset.

Now, run:

```
locale-gen
```

Restart the web server.

The locales should now be installed. It may be required to reload Zabbix frontend page in browser using Ctrl + F5 for new languages to appear.

Installing Zabbix
If installing Zabbix directly from Zabbix git repository, translation files should be generated manually. To generate translation files, run:

```
make gettext
locale/make_mo.sh
```

This step is not needed when installing Zabbix from packages or source tar.gz files.

Selecting a language

There are several ways to select a language in Zabbix web interface:

- When installing web interface - in the frontend installation wizard. Selected language will be set as system default.
- After the installation, system default language can be changed in the Administration→General→GUI menu section.
- Language for a particular user can be changed in the user profile.

If a locale for a language is not installed on the machine, this language will be greyed out in Zabbix language selector. A red icon is displayed next to the language selector if at least one locale is missing. Upon pressing on this icon the following message will be displayed: "You are not able to choose some of the languages, because locales for them are not installed on the web server.”

### 3 Process configuration

#### 1 Zabbix server

**Overview**

This section lists parameters supported in a Zabbix server configuration file (zabbix_server.conf).

**Note that:**

- The default values reflect daemon defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported in the beginning of the line.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlertScriptsPath</td>
<td>no</td>
<td>/usr/local/share/zabbix/alertscripts</td>
<td></td>
<td>Location of custom alert scripts (depends on compile-time installation variable datadir).</td>
</tr>
<tr>
<td>AllowRoot</td>
<td>no</td>
<td>0</td>
<td>0</td>
<td>Allow the server to run as 'root'. If disabled and the server is started by 'root', the server will try to switch to the 'zabbix' user instead. Has no effect if started under a regular user. 0 - do not allow 1 - allow</td>
</tr>
<tr>
<td>AllowUnsupportedDBVersions</td>
<td>no</td>
<td>0</td>
<td>0</td>
<td>Allow the server to work with unsupported database versions. 0 - do not allow 1 - allow</td>
</tr>
<tr>
<td>CacheSize</td>
<td>no</td>
<td>128K-64G</td>
<td>32M</td>
<td>Size of configuration cache, in bytes. Shared memory size for storing host, item and trigger data.</td>
</tr>
<tr>
<td>CacheUpdateFrequency</td>
<td></td>
<td>1-3600</td>
<td>60</td>
<td>How often Zabbix will perform update of configuration cache, in seconds. See also runtime control options.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>--------------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DBHost</td>
<td>no</td>
<td>localhost</td>
<td></td>
<td>Database host name. In case of MySQL localhost or empty string results in using a socket. In case of PostgreSQL only empty string results in attempt to use socket. In case of Oracle empty string results in using the Net Service Name connection method; in this case consider using the TNS_ADMIN environment variable to specify the directory of the tnsnames.ora file.</td>
</tr>
<tr>
<td>DBName</td>
<td>yes</td>
<td></td>
<td></td>
<td>Database name. In case of Oracle if the Net Service Name connection method is used, specify the service name from tnsnames.ora or set to empty string; set the TWO_TASK environment variable if DBName is set to empty string.</td>
</tr>
<tr>
<td>DBPassword</td>
<td>no</td>
<td></td>
<td></td>
<td>Database password. Comment this line if no password is used.</td>
</tr>
<tr>
<td>DBPort</td>
<td>no</td>
<td>1024-65535</td>
<td></td>
<td>Database port when not using local socket. In case of Oracle if the Net Service Name connection method is used this parameter will be ignored; the port number from the tnsnames.ora file will be used instead.</td>
</tr>
<tr>
<td>DBSchema</td>
<td>no</td>
<td></td>
<td></td>
<td>Schema name. Used for PostgreSQL.</td>
</tr>
<tr>
<td>DBSocket</td>
<td>no</td>
<td></td>
<td></td>
<td>Path to MySQL socket file.</td>
</tr>
<tr>
<td>DBUser</td>
<td>no</td>
<td></td>
<td></td>
<td>Database user.</td>
</tr>
<tr>
<td>DBTLSConnect</td>
<td>no</td>
<td></td>
<td></td>
<td>Setting this option enforces to use TLS connection to database: required - connect using TLS verify_ca - connect using TLS and verify certificate verify_full - connect using TLS, verify certificate and verify that database identity specified by DBHost matches its certificate. On MySQL starting from 5.7.11 and PostgreSQL the following values are supported: &quot;required&quot;, &quot;verify_ca&quot;, &quot;verify_full&quot;. On MariaDB starting from version 10.2.6 &quot;required&quot; and &quot;verify_full&quot; values are supported. By default not set to any option and the behavior depends on database configuration.</td>
</tr>
<tr>
<td>DBTLSCAFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the top-level CA(s) certificates for database certificate verification. This parameter is supported since Zabbix 5.0.0.</td>
</tr>
<tr>
<td>DBTLSCertFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of file containing Zabbix server certificate for authenticating to database. This parameter is supported since Zabbix 5.0.0.</td>
</tr>
<tr>
<td>DBTLSKeyFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of file containing the private key for authenticating to database. This parameter is supported since Zabbix 5.0.0.</td>
</tr>
<tr>
<td>DBTLSCipher</td>
<td>no</td>
<td></td>
<td></td>
<td>The list of encryption ciphers that Zabbix server permits for TLS protocols up through TLSv1.2. Supported only for MySQL. This parameter is supported since Zabbix 5.0.0.</td>
</tr>
<tr>
<td>DBTLSCipher13</td>
<td>no</td>
<td></td>
<td></td>
<td>The list of encryption ciphersuites that Zabbix server permits for TLSv1.3 protocol. Supported only for MySQL, starting from version 8.0.16. This parameter is supported since Zabbix 5.0.0.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------</td>
<td>-----------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| DebugLevel         | no        | 0-5       | 3       | Specifies debug level:  
0 - basic information about starting and stopping of Zabbix processes  
1 - critical information  
2 - error information  
3 - warnings  
4 - for debugging (produces lots of information)  
5 - extended debugging (produces even more information)  
See also runtime control options. |
| ExportDir          | no        | Directory |         | Directory for real-time export of events, history and trends in newline-delimited JSON format. If set, enables real-time export.  
This parameter is supported since Zabbix 4.0.0. |
| ExportFileSize     | no        | 1M-1G     | 1G      | Maximum size per export file in bytes. Only used for rotation if ExportDir is set.  
This parameter is supported since Zabbix 4.0.0. |
| ExportType         | no        | List of comma-delimited entity types (events, history, trends) for real-time export (all types by default). Valid only if ExportDir is set.  
Note that if ExportType is specified, but ExportDir is not, then this is a configuration error and the server will not start.  
e.g.: ExportType=history,trends - export history and trends only  
ExportType=events - export events only |
| ExternalScripts    | no        | /usr/local/share/zabbix/externalscripts | Location of externalscripts (depends on compile-time installation variable datadir). |
| Fping6Location     | no        | /usr/sbin/fping6 | Location of fping6.  
Make sure that fping6 binary has root ownership and SUID flag set.  
Make empty ("Fping6Location=") if your fping utility is capable to process IPv6 addresses. |
| FpingLocation      | no        | /usr/sbin/fping | Location of fping.  
Make sure that fping binary has root ownership and SUID flag set! |
| HANodeName         | no        | The high availability cluster node name.  
When empty the server is working in standalone mode and a node with empty name is created. |
| HistoryIndexCacheSize |       | 128K-2G   | 4M      | Size of history index cache, in bytes.  
Shared memory size for indexing history data stored in history cache.  
The index cache size needs roughly 100 bytes to cache one item.  
This parameter is supported since Zabbix 3.0.0. |
| HistoryStorageDateIndex |   | 0         |         | Enable preprocessing of history values in history storage to store values in different indices based on date:  
0 - disable  
1 - enable |
| HistoryStorageURL  |           |           |         | History storage HTTP[S] URL.  
This parameter is used for Elasticsearch setup. |
| HistoryStorageTypes|           | uint,dbl,str,log | Comma separated list of value types to be sent to the history storage.  
This parameter is used for Elasticsearch setup. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HousekeepingFrequency</td>
<td>no</td>
<td>0-24</td>
<td>1</td>
<td>How often Zabbix will perform housekeeping procedure (in hours). Housekeeping is removing outdated information from the database. Note: To prevent housekeeper from being overloaded (for example, when history and trend periods are greatly reduced), no more than 4 times HousekeepingFrequency hours of outdated information are deleted in one housekeeping cycle, for each item. Thus, if HousekeepingFrequency is 1, no more than 4 hours of outdated information (starting from the oldest entry) will be deleted per cycle. Note: To lower load on server startup housekeeping is postponed for 30 minutes after server start. Thus, if HousekeepingFrequency is 1, the very first housekeeping procedure after server start will run after 30 minutes, and will repeat with one hour delay thereafter. Since Zabbix 3.0.0 it is possible to disable automatic housekeeping by setting HousekeepingFrequency to 0. In this case the housekeeping procedure can only be started by housekeeper_execute runtime control option and the period of outdated information deleted in one housekeeping cycle is 4 times the period since the last housekeeping cycle, but not less than 4 hours and not greater than 4 days. See also runtime control options.</td>
</tr>
<tr>
<td>Include</td>
<td>no</td>
<td></td>
<td></td>
<td>You may include individual files or all files in a directory in the configuration file. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. For example: /absolute/path/to/config/files/*.conf. See special notes about limitations.</td>
</tr>
<tr>
<td>JavaGateway</td>
<td>no</td>
<td></td>
<td></td>
<td>IP address (or hostname) of Zabbix Java gateway. Only required if Java pollers are started.</td>
</tr>
<tr>
<td>JavaGatewayPort</td>
<td></td>
<td>1024-32767</td>
<td>10052</td>
<td>Port that Zabbix Java gateway listens on.</td>
</tr>
<tr>
<td>ListenBacklog</td>
<td>no</td>
<td>0 - INT_MAX</td>
<td>SOMAXCONN</td>
<td>The maximum number of pending connections in the TCP queue. Default value is a hard-coded constant, which depends on the system. Maximum supported value depends on the system, too high values may be silently truncated to the 'implementation-specified maximum'.</td>
</tr>
<tr>
<td>ListenIP</td>
<td>no</td>
<td>0.0.0.0</td>
<td></td>
<td>List of comma delimited IP addresses that the trapper should listen on. Trapper will listen on all network interfaces if this parameter is missing.</td>
</tr>
<tr>
<td>ListenPort</td>
<td>no</td>
<td>1024-32767</td>
<td>10051</td>
<td>Listen port for trapper.</td>
</tr>
<tr>
<td>LoadModule</td>
<td>no</td>
<td></td>
<td></td>
<td>Module to load at server startup. Modules are used to extend functionality of the server. Formats: LoadModule=&lt;module.so&gt; LoadModule=&lt;path/module.so&gt; LoadModule=&lt;/abs_path/module.so&gt; Either the module must be located in directory specified by LoadModulePath or the path must precede the module name. If the preceding path is absolute (starts with '/') then LoadModulePath is ignored. It is allowed to include multiple LoadModule parameters.</td>
</tr>
<tr>
<td>LoadModulePath</td>
<td>no</td>
<td></td>
<td></td>
<td>Full path to location of server modules. Default depends on compilation options.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------</td>
<td>-----------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LogFile</td>
<td>yes, if LogType is set to file, otherwise no</td>
<td>0-1024</td>
<td>1</td>
<td>Name of log file.</td>
</tr>
<tr>
<td>LogFileSize</td>
<td>no</td>
<td>0-1024</td>
<td>1</td>
<td>Maximum size of log file in MB. 0 - disable automatic log rotation. Note: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.</td>
</tr>
<tr>
<td>LogType</td>
<td>no</td>
<td>file</td>
<td></td>
<td>Log output type: file - write log to file specified by LogFile parameter, system - write log to syslog, console - write log to standard output. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>LogSlowQueries</td>
<td>no</td>
<td>0-3600000</td>
<td>0</td>
<td>How long a database query may take before being logged (in milliseconds). 0 - don't log slow queries. This option becomes enabled starting with DebugLevel=3.</td>
</tr>
</tbody>
</table>
| MaxHousekeeperDelete      | no        | 0-1000000 | 5000    | No more than ‘MaxHousekeeperDelete’ rows (corresponding to [tablename], [field], [value]) will be deleted per one task in one housekeeping cycle. If set to 0 then no limit is used at all. In this case you must know what you are doing, so as not to overload the database! 2  
This parameter applies only to deleting history and trends of already deleted items. |
| NodeAddress               | no        |          |         | IP or hostname with optional port to override how the frontend should connect to the server. Format: <address>::[port] The priority of addresses used by the frontend to specify the server address is: NodeAddress (1); ListenIP (if not 0.0.0.0 or ::) (2); localhost (default) (3) See also: HANodeName parameter |
| PidFile                   | no        | /tmp     |         | Name of PID file.                                                          |
| ProxyConfigFrequency      | 1-604800  | 3600     |         | How often Zabbix server sends configuration data to a Zabbix proxy in seconds. Used only for proxies in a passive mode.  |
| ProxyDataFrequency        | 1-3600    | 1        |         | How often Zabbix server requests history data from a Zabbix proxy in seconds. Used only for proxies in a passive mode. |
| SNMPTrapperFile           | no        | /tmp     |         | Temporary file used for passing data from SNMP trap daemon to the server. Must be the same as in zabbix_trap_receiver.pl or SNMP TT configuration file.  |
| SocketDir                 | no        | /tmp     |         | Directory to store IPC sockets used by internal Zabbix services.  
This parameter is supported since Zabbix 3.4.0. |
| SourceIP                  | no        |          |         | Source IP address for:  
- outgoing connections to Zabbix proxy and Zabbix agent;  
- agentless connections (VMware, SSH, JMX, SNMP, Telnet and simple checks);  
- HTTP agent connections;  
- script item JavaScript HTTP requests;  
- preprocessing JavaScript HTTP requests;  
- sending notification emails (connections to SMTP server);  
- webhook notifications (JavaScript HTTP connections);  
- connections to the Vault  |
<p>| SSHKeyLocation            | no        |          |         | Location of public and private keys for SSH checks and actions  |
| SSLCertLocation           | no        |          |         | Location of SSL client certificate files for client authentication. This parameter is used in web monitoring only.  |
| SSLKeyLocation            | no        |          |         | Location of SSL private key files for client authentication. This parameter is used in web monitoring only.  |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSLCALocation</td>
<td>no</td>
<td></td>
<td></td>
<td>Override the location of certificate authority (CA) files for SSL server</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>certificate verification. If not set, system-wide directory will be used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note that the value of this parameter will be set as libcurl option</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CURLOPT_CAPATH. For libcurl versions before 7.42.0, this only has effect if</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>libcurl was compiled to use OpenSSL. For more information see cURL web</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>page.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This parameter is used in web monitoring since Zabbix 2.4.0 and in SMTP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>authentication since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>StartAlerters</td>
<td>no</td>
<td>1-100</td>
<td>3</td>
<td>Number of pre-forked instances of alerters.</td>
</tr>
<tr>
<td>StartDBSyncers</td>
<td>no</td>
<td>1-100</td>
<td>4</td>
<td>Number of pre-forked instances of history syncers.</td>
</tr>
<tr>
<td>StartDiscoverers</td>
<td>no</td>
<td>0-250</td>
<td>1</td>
<td>Number of pre-forked instances of discoverers.</td>
</tr>
<tr>
<td>StartEscalators</td>
<td>no</td>
<td>1-100</td>
<td>1</td>
<td>Number of pre-forked instances of escalators.</td>
</tr>
<tr>
<td>StartHistoryPollers</td>
<td>no</td>
<td>0-1000</td>
<td>5</td>
<td>Number of pre-forked instances of history pollers.</td>
</tr>
<tr>
<td>StartHTTPPollers</td>
<td>no</td>
<td>0-1000</td>
<td>1</td>
<td>Number of pre-forked instances of HTTP pollers.</td>
</tr>
<tr>
<td>StartIPMIPollers</td>
<td>no</td>
<td>0-1000</td>
<td>0</td>
<td>Number of pre-forked instances of IPMI pollers.</td>
</tr>
<tr>
<td>StartJavaPollers</td>
<td>no</td>
<td>0-1000</td>
<td>0</td>
<td>Number of pre-forked instances of Java pollers.</td>
</tr>
<tr>
<td>StartLLDProcessors</td>
<td>no</td>
<td>1-100</td>
<td>2</td>
<td>Number of pre-forked instances of low-level discovery (LLD) workers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The LLD manager process is automatically started when an LLD worker is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>started.</td>
</tr>
<tr>
<td>StartODBCPollers</td>
<td>no</td>
<td>0-1000</td>
<td>1</td>
<td>Number of pre-forked instances of ODBC pollers.</td>
</tr>
<tr>
<td>StartPingers</td>
<td>no</td>
<td>0-1000</td>
<td>1</td>
<td>Number of pre-forked instances of ICMP pingers.</td>
</tr>
<tr>
<td>StartPollersUnreachable</td>
<td>no</td>
<td>0-1000</td>
<td>1</td>
<td>Number of pre-forked instances of pollers for unreachable hosts (including</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IPMI and Java)².</td>
</tr>
<tr>
<td>StartPollers</td>
<td>no</td>
<td>0-1000</td>
<td>5</td>
<td>Number of pre-forked instances of pollers.</td>
</tr>
<tr>
<td>StartPreprocessors</td>
<td>no</td>
<td>1-1000</td>
<td>3</td>
<td>Number of pre-forked instances of preprocessing workers.²</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The preprocessing manager process is automatically started when a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>preprocessor worker is started.</td>
</tr>
<tr>
<td>StartProxyPollers</td>
<td>no</td>
<td>0-250</td>
<td>1</td>
<td>Number of pre-forked instances of pollers for passive proxies.¹</td>
</tr>
<tr>
<td>StartReportWriters</td>
<td>no</td>
<td>0-100</td>
<td>0</td>
<td>Number of pre-forked instances of report writers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If set to 0, scheduled report generation is disabled. The report manager</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>process is automatically started when a report writer is started.</td>
</tr>
<tr>
<td>StartSNMPTrappers</td>
<td>no</td>
<td>0-1</td>
<td>0</td>
<td>If set to 1, SNMP trapper process will be started.</td>
</tr>
<tr>
<td>StartTimers</td>
<td>no</td>
<td>1-1000</td>
<td>1</td>
<td>Number of pre-forked instances of timers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timers process maintenance periods.</td>
</tr>
<tr>
<td>StartTrappers</td>
<td>no</td>
<td>0-1000</td>
<td>5</td>
<td>Number of pre-forked instances of trappers¹.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Trappers accept incoming connections from Zabbix sender, active agents and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>active proxies.</td>
</tr>
<tr>
<td>StartVMwareCollectors</td>
<td>no</td>
<td>0-250</td>
<td>0</td>
<td>Number of pre-forked VMware collector instances.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>--------</td>
<td>---------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>StatsAllowedIP</td>
<td>no</td>
<td>List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of external Zabbix instances. Stats request will be accepted only from the addresses listed here. If this parameter is not set no stats requests will be accepted. If IPv6 support is enabled then ‘127.0.0.1’, ‘::127.0.0.1’, ‘::ffff:127.0.0.1’ are treated equally and ‘::/0’ will allow any IPv4 or IPv6 address. ‘0.0.0.0/0’ can be used to allow any IPv4 address. Example: StatsAllowedIP=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32.zabbix.example.com This parameter is supported since Zabbix 4.2.0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timeout</td>
<td>no</td>
<td>1-30</td>
<td>3</td>
<td>Specifies how long we wait for agent, SNMP device or external check (in seconds).</td>
</tr>
<tr>
<td>TLSCAFile</td>
<td>no</td>
<td>Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSCertFile</td>
<td>no</td>
<td>Full pathname of a file containing the server certificate or certificate chain, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSCipherAll</td>
<td>no</td>
<td>GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption. Example: TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA:TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA:TLS_RSA_WITH_AES_128_CBC_SHA:TLS_RSA_WITH_AES_256_CBC_SHA This parameter is supported since Zabbix 4.4.7.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSCipherCert13</td>
<td>no</td>
<td>Cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate-based encryption. This parameter is supported since Zabbix 4.4.7.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TLSCipherPSK13</td>
<td>no</td>
<td></td>
<td></td>
<td>Cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ciphersuite selection criteria for PSK-based encryption. Example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This parameter is supported since Zabbix 4.4.7.</td>
</tr>
<tr>
<td>TLSCRLFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing revoked certificates. This parameter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>is used for encrypted communications between Zabbix components. This</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSKeyFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the server private key, used for</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>encrypted communications between Zabbix components. This parameter is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TmpDir</td>
<td>no</td>
<td>/tmp</td>
<td></td>
<td>Temporary directory.</td>
</tr>
<tr>
<td>TrapperTimeout</td>
<td>no</td>
<td>1-300</td>
<td>300</td>
<td>Specifies how many seconds trapper may spend processing new data.</td>
</tr>
<tr>
<td>TrendFunctionCacheSize</td>
<td>no</td>
<td>128K-2G</td>
<td>4M</td>
<td>Size of trend function cache, in bytes. Shared memory size for caching</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>calculated trend function data.</td>
</tr>
<tr>
<td>UnavailableDelay</td>
<td>no</td>
<td>1-3600</td>
<td>60</td>
<td>How often host is checked for availability during the unavailability period,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>in seconds.</td>
</tr>
<tr>
<td>UnreachableDelay</td>
<td>no</td>
<td>1-3600</td>
<td>15</td>
<td>How often host is checked for availability during the unreachability period,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>in seconds.</td>
</tr>
<tr>
<td>UnreachablePeriod</td>
<td>no</td>
<td>1-3600</td>
<td>45</td>
<td>After how many seconds of unreachability treat a host as unavailable.</td>
</tr>
<tr>
<td>User</td>
<td>no</td>
<td>zabbix</td>
<td></td>
<td>Drop privileges to a specific, existing user on the system. Only has effect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>if run as ‘root’ and AllowRoot is disabled.</td>
</tr>
<tr>
<td>ValueCacheSize</td>
<td>no</td>
<td>0,128K-64G</td>
<td>8M</td>
<td>Size of history value cache, in bytes. Shared memory size for caching item</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>history data requests. Setting to 0 disables value cache (not recommended).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When value cache runs out of the shared memory a warning message is written</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>to the server log every 5 minutes.</td>
</tr>
<tr>
<td>Vault</td>
<td>no</td>
<td>HashiCorp</td>
<td></td>
<td>Specifies the vault provider:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HashiCorp - HashiCorp KV Secrets Engine version 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CyberArk - CyberArk Central Credential Provider</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Must match the vault provider set in the frontend.</td>
</tr>
<tr>
<td>VaultDBPath</td>
<td>no</td>
<td></td>
<td></td>
<td>Specifies a location, from where database credentials should be retrieved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>by keys. Depending on the Vault, can be vault path or query. Keys used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>for HashiCorp are ‘password’ and ‘username’. Example: secret/zabbix/database</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Keys used for CyberArk are ‘Content’ and ‘UserName’. Example: Ap-PID=zabbix</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>server&amp;Query=Safe=passwordSafe;Object=zabbix_proxy_database</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This option can only be used if DBUser and DBPassword are not specified.</td>
</tr>
<tr>
<td>VaultTLSCertFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Name of the SSL certificate file used for client authentication. The</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>certificate file must be in PEM1 format. If the certificate file contains</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>also the private key, leave the SSL key file field empty. The directory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>containing this file is specified by the configuration parameter SSLCertLocation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This option can be omitted but is recommended for CyberArkCCP vault.</td>
</tr>
<tr>
<td>VaultTLSKeyFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Name of the SSL private key file used for client authentication. The</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>private key file must be in PEM1 format. The directory containing this file</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>is specified by the configuration parameter SSLKeyLocation. This option</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>can be omitted but is recommended for CyberArkCCP vault.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>---------------</td>
<td>---------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VaultToken</td>
<td>yes, if Vault is set to HashiCorp, otherwise no</td>
<td></td>
<td></td>
<td>HashiCorp Vault authentication token that should have been generated exclusively for Zabbix server with read-only permission to the paths specified in Vault macros and read-only permission to the path specified in the optional VaultDBPath configuration parameter. It is an error if VaultToken and VAULT_TOKEN environment variable are defined at the same time.</td>
</tr>
<tr>
<td>VaultURL</td>
<td>no</td>
<td><a href="https://127.0.0.1:8200">https://127.0.0.1:8200</a></td>
<td></td>
<td>URL of HashiCorp Vault server HTTP[S] URL. System-wide CA certificates directory will be used if SSLCALocation is not specified.</td>
</tr>
<tr>
<td>VMwareCacheSize</td>
<td>no</td>
<td>256K-2G</td>
<td>8M</td>
<td>Shared memory size for storing VMware data. A VMware internal check zabbix{vmware,buffer,...} can be used to monitor the VMware cache usage (see Internal checks).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note that shared memory is not allocated if there are no VMware collector instances configured to start.</td>
</tr>
<tr>
<td>VMwareFrequency</td>
<td>no</td>
<td>10-86400</td>
<td>60</td>
<td>Delay in seconds between data gathering from a single VMware service. This delay should be set to the least update interval of any VMware monitoring item.</td>
</tr>
<tr>
<td>VMwarePerfFrequency</td>
<td>no</td>
<td>10-86400</td>
<td>60</td>
<td>Delay in seconds between performance counter statistics retrieval from a single VMware service. This delay should be set to the least update interval of any VMware monitoring item that uses VMware performance counters.</td>
</tr>
<tr>
<td>VMwareTimeout</td>
<td>no</td>
<td>1-300</td>
<td>10</td>
<td>The maximum number of seconds vmware collector will wait for a response from VMware service (vCenter or ESX hypervisor).</td>
</tr>
<tr>
<td>WebServiceURL</td>
<td>no</td>
<td></td>
<td></td>
<td>HTTP[S] URL to Zabbix web service in the format <a href="">host:port</a>/report. For example: <a href="http://localhost:10053/report">http://localhost:10053/report</a>. This parameter is supported since Zabbix 5.4.0.</td>
</tr>
</tbody>
</table>

Footnotes

1  Note that too many data gathering processes (pollers, unreachable pollers, ODBC pollers, HTTP pollers, Java pollers, pingers, trappers, proxypollers) together with IPMI manager, SNMP trapper and preprocessing workers can exhaust the per-process file descriptor limit for the preprocessing manager.

This will cause Zabbix server to stop (usually shortly after the start, but sometimes it can take more time). The configuration file should be revised or the limit should be raised to avoid this situation.

2  When a lot of items are deleted it increases the load to the database, because the housekeeper will need to remove all the history data that these items had. For example, if we only have to remove 1 item prototype, but this prototype is linked to 50 hosts and for every host the prototype is expanded to 100 real items, 5000 items in total have to be removed (1*50*100). If 500 is set for MaxHousekeeperDelete (MaxHousekeeperDelete=500), the housekeeper process will have to remove up to 2500000 values (5000*500) for the deleted items from history and trends tables in one cycle.

2 Zabbix proxy

Overview

This section lists parameters supported in a Zabbix proxy configuration file (zabbix_proxy.conf).

Note that:

- The default values reflect daemon defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported in the beginning of the line.

Parameters
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowRoot</td>
<td>no</td>
<td>0</td>
<td>0</td>
<td>Allow the proxy to run as 'root'. If disabled and the proxy is started by 'root', the proxy will try to switch to the 'zabbix' user instead. Has no effect if started under a regular user.</td>
</tr>
<tr>
<td>AllowUnsupportedDBVersions</td>
<td>no</td>
<td>0</td>
<td>1</td>
<td>Allow the proxy to work with unsupported database versions.</td>
</tr>
<tr>
<td>CacheSize</td>
<td>no</td>
<td>128K-64G</td>
<td>32M</td>
<td>Size of configuration cache, in bytes.</td>
</tr>
<tr>
<td>ConfigFrequency</td>
<td>no</td>
<td>1-604800</td>
<td>3600</td>
<td>How often proxy retrieves configuration data from Zabbix server in seconds. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).</td>
</tr>
<tr>
<td>DataSenderFrequency</td>
<td>1-3600</td>
<td>1</td>
<td></td>
<td>Proxy will send collected data to the server every N seconds. Note that active proxy will still poll Zabbix server every second for remote command tasks. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).</td>
</tr>
<tr>
<td>DBHost</td>
<td>no</td>
<td>localhost</td>
<td></td>
<td>Database host name. In case of MySQL localhost or empty string results in using a socket. In case of PostgreSQL only empty string results in attempt to use socket. In case of Oracle empty string results in using the Net Service Name connection method; in this case consider using the TNS_ADMIN environment variable to specify the directory of the tnsnames.ora file.</td>
</tr>
<tr>
<td>DBName</td>
<td>yes</td>
<td></td>
<td></td>
<td>Database name or path to database file for SQLite3 (multi-process architecture of Zabbix does not allow to use in-memory database, e.g. :memory:, file::memory:?cache=shared or file:membdb1?mode=memory&amp;cache=shared).</td>
</tr>
<tr>
<td>DBPassword</td>
<td>no</td>
<td></td>
<td></td>
<td>Database password. Ignored for SQLite. Comment this line if no password is used.</td>
</tr>
<tr>
<td>DBSchema</td>
<td>no</td>
<td></td>
<td></td>
<td>Schema name. Used for PostgreSQL.</td>
</tr>
<tr>
<td>DBSocket</td>
<td>no</td>
<td>3306</td>
<td></td>
<td>Path to MySQL socket. Database port when not using local socket. Ignored for SQLite.</td>
</tr>
<tr>
<td>DBUser</td>
<td></td>
<td></td>
<td></td>
<td>Database user. Ignored for SQLite.</td>
</tr>
<tr>
<td>DBTLSConnect</td>
<td>no</td>
<td></td>
<td></td>
<td>Setting this option enforces to use TLS connection to database: required - connect using TLS verify_ca - connect using TLS and verify certificate verify_full - connect using TLS, verify certificate and verify that database identity specified by DBHost matches its certificate</td>
</tr>
</tbody>
</table>

On MySQL starting from 5.7.11 and PostgreSQL the following values are supported: “required”, “verify”, “verify_full”. On MariaDB starting from version 10.2.6 “required” and “verify_full” values are supported. By default not set to any option and the behavior depends on database configuration.

This parameter is supported since Zabbix 5.0.0.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBTLSCAFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the top-level CA(s) certificates for database certificate verification. This parameter is supported since Zabbix 5.0.0.</td>
</tr>
<tr>
<td>DBTLSCertFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of file containing Zabbix server certificate for authenticating to database. This parameter is supported since Zabbix 5.0.0.</td>
</tr>
<tr>
<td>DBTLSKeyFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of file containing the private key for authenticating to database. This parameter is supported since Zabbix 5.0.0.</td>
</tr>
<tr>
<td>DBTLSCipher</td>
<td>no</td>
<td></td>
<td></td>
<td>The list of encryption ciphers that Zabbix server permits for TLS protocols up through TLSv1.2. Supported only for MySQL. This parameter is supported since Zabbix 5.0.0.</td>
</tr>
<tr>
<td>DBTLSCipher13</td>
<td>no</td>
<td></td>
<td></td>
<td>The list of encryption ciphersuites that Zabbix server permits for TLSv1.3 protocol. Supported only for MySQL, starting from version 8.0.16. This parameter is supported since Zabbix 5.0.0.</td>
</tr>
<tr>
<td>DebugLevel</td>
<td>no</td>
<td>0-5</td>
<td>3</td>
<td>Specifies debug level: 0 - basic information about starting and stopping of Zabbix processes 1 - critical information 2 - error information 3 - warnings 4 - for debugging (produces lots of information) 5 - extended debugging (produces even more information)</td>
</tr>
<tr>
<td>EnableRemoteCommands</td>
<td></td>
<td>0</td>
<td></td>
<td>Whether remote commands from Zabbix server are allowed. 0 - not allowed 1 - allowed This parameter is supported since Zabbix 3.4.0.</td>
</tr>
<tr>
<td>ExternalScripts</td>
<td>no</td>
<td></td>
<td></td>
<td>Location of externalscripts (depends on compile-time installation variable datadir).</td>
</tr>
<tr>
<td>Fping6Location</td>
<td>no</td>
<td></td>
<td></td>
<td>Location of fping6. Make sure that fping6 binary has root ownership and SUID flag set. Make empty (&quot;Fping6Location=&quot;) if your fping utility is capable to process IPv6 addresses.</td>
</tr>
<tr>
<td>FpingLocation</td>
<td>no</td>
<td>/usr/sbin/fping</td>
<td></td>
<td>Location of fping. Make sure that fping binary has root ownership and SUID flag set!</td>
</tr>
<tr>
<td>HeartbeatFrequency</td>
<td></td>
<td>0-3600</td>
<td>60</td>
<td>Frequency of heartbeat messages in seconds. Used for monitoring availability of proxy on server side. 0 - heartbeat messages disabled. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).</td>
</tr>
<tr>
<td>HistoryIndexCacheSize</td>
<td></td>
<td>128K-2G</td>
<td>4M</td>
<td>Size of history index cache, in bytes. Shared memory size for storing history data. Size of history index cache, in bytes. Shared memory size for indexing history data stored in history cache. The index cache size needs roughly 100 bytes to cache one item.</td>
</tr>
<tr>
<td>Hostname</td>
<td>no</td>
<td>Set by HostnameItem</td>
<td></td>
<td>Unique, case sensitive Proxy name. Make sure the proxy name is known to the server! Allowed characters: alphanumeric, '.', '-', '_' and '-' Maximum length: 128</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------</td>
<td>-------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HostnameItem</td>
<td>no</td>
<td>system.hostname</td>
<td>used for setting Hostname if it is undefined (this will be run on the proxy similarly as on an agent). Does not support UserParameters, performance counters or aliases, but does support system.run[].</td>
<td></td>
</tr>
<tr>
<td>HousekeepingFrequency</td>
<td>0-24</td>
<td>1</td>
<td></td>
<td>Ignored if Hostname is set. How often Zabbix will perform housekeeping procedure (in hours). Housekeeping is removing outdated information from the database. Note: To prevent housekeeper from being overloaded (for example, when configuration parameters ProxyLocalBuffer or ProxyOfflineBuffer are greatly reduced), no more than 4 times HousekeepingFrequency hours of outdated information are deleted in one housekeeping cycle. Thus, if HousekeepingFrequency is 1, no more than 4 hours of outdated information (starting from the oldest entry) will be deleted per cycle. Note: To lower load on proxy startup housekeeping is postponed for 30 minutes after proxy start. Thus, if HousekeepingFrequency is 1, the very first housekeeping procedure after proxy start will run after 30 minutes, and will repeat every hour thereafter. Since Zabbix 3.0.0 it is possible to disable automatic housekeeping by setting HousekeepingFrequency to 0. In this case the housekeeping procedure can only be started by housekeeper_execute runtime control option and the period of outdated information deleted in one housekeeping cycle is 4 times the period since the last housekeeping cycle, but not less than 4 hours and not greater than 4 days.</td>
</tr>
<tr>
<td>Include</td>
<td>no</td>
<td></td>
<td></td>
<td>You may include individual files or all files in a directory in the configuration file. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. For example: /absolute/path/to/config/files/*.conf. See special notes about limitations.</td>
</tr>
<tr>
<td>JavaGateway</td>
<td>no</td>
<td>IP address (or hostname) of Zabbix Java gateway.</td>
<td>Only required if Java pollers are started.</td>
<td></td>
</tr>
<tr>
<td>JavaGatewayPort</td>
<td>1024-32767</td>
<td>10052</td>
<td></td>
<td>Port that Zabbix Java gateway listens on.</td>
</tr>
<tr>
<td>ListenBacklog</td>
<td>no</td>
<td>0 - INT_MAX</td>
<td>SOMAXCONN</td>
<td>The maximum number of pending connections in the TCP queue. Default value is a hard-coded constant, which depends on the system. Maximum supported value depends on the system, too high values may be silently truncated to the 'implementation-specified maximum'.</td>
</tr>
<tr>
<td>ListenIP</td>
<td>no</td>
<td>0.0.0.0</td>
<td></td>
<td>List of comma delimited IP addresses that the trapper should listen on. Trapper will listen on all network interfaces if this parameter is missing.</td>
</tr>
<tr>
<td>ListenPort</td>
<td>no</td>
<td>1024-32767</td>
<td>10051</td>
<td>Listen port for trapper.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------</td>
<td>-------------</td>
<td>---------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LoadModule</td>
<td>no</td>
<td></td>
<td></td>
<td>Module to load at proxy startup. Modules are used to extend functionality of the proxy.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Formats:&lt;br&gt;LoadModule=&lt;module.so&gt;&lt;br&gt;LoadModule=&lt;path/module.so&gt;&lt;br&gt;LoadModule=&lt;/abs_path/module.so&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Either the module must be located in directory specified by LoadModulePath or the path must precede the module name. If the preceding path is absolute (starts with '/') then LoadModulePath is ignored.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>It is allowed to include multiple LoadModule parameters.</td>
</tr>
<tr>
<td>LoadModulePath</td>
<td>no</td>
<td></td>
<td></td>
<td>Full path to location of proxy modules.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Default depends on compilation options.</td>
</tr>
<tr>
<td>LogFile</td>
<td>yes, if LogType is set to file, otherwise no</td>
<td></td>
<td></td>
<td>Name of log file.</td>
</tr>
<tr>
<td>LogFileSize</td>
<td>no</td>
<td>0-1024</td>
<td>1</td>
<td>Maximum size of log file in MB. 0 - disable automatic log rotation. Note: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.</td>
</tr>
<tr>
<td>LogRemoteCommands</td>
<td>no</td>
<td></td>
<td>0</td>
<td>Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled This parameter is supported since Zabbix 3.4.0.</td>
</tr>
<tr>
<td>LogType</td>
<td>no</td>
<td>file</td>
<td></td>
<td>Log output type: file - write log to file specified by LogFile parameter, system - write log to syslog, console - write log to standard output. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>LogSlowQueries</td>
<td>no</td>
<td>0-3600000</td>
<td>0</td>
<td>How long a database query may take before being logged (in milliseconds). 0 - don’t log slow queries. This option becomes enabled starting with DebugLevel=3.</td>
</tr>
<tr>
<td>PidFile</td>
<td>no</td>
<td>/tmp/zabbix_proxy.pid</td>
<td></td>
<td>Name of PID file.</td>
</tr>
<tr>
<td>ProxyLocalBuffen</td>
<td>no</td>
<td>0-720</td>
<td>0</td>
<td>Proxy will keep data locally for N hours, even if the data have already been synced with the server. This parameter may be used if local data will be used by third party applications.</td>
</tr>
<tr>
<td>ProxyMode</td>
<td>no</td>
<td>0-1</td>
<td>0</td>
<td>Proxy operating mode. 0 - proxy in the active mode 1 - proxy in the passive mode. Note that (sensitive) proxy configuration data may become available to parties having access to the Zabbix server trapper port when using an active proxy. This is possible because anyone may pretend to be an active proxy and request configuration data; authentication does not take place.</td>
</tr>
<tr>
<td>ProxyOfflineBuffen</td>
<td>no</td>
<td>1-720</td>
<td>1</td>
<td>Proxy will keep data for N hours in case of no connectivity with Zabbix server. Older data will be lost.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------</td>
<td>----------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Server</td>
<td>yes</td>
<td></td>
<td></td>
<td>If ProxyMode is set to active mode: Zabbix server IP address or DNS name (address:port) or cluster (address:port;address2:port) to get configuration data from and send data to. If port is not specified, the default port is used. Cluster nodes must be separated by a semicolon.</td>
</tr>
<tr>
<td>SNMPTrapperFile</td>
<td>no</td>
<td>/tmp</td>
<td></td>
<td>Temporary file used for passing data from SNMP trap daemon to the proxy. Must be the same as in zabbix_trap_receiver.pl or SNMP TT configuration file.</td>
</tr>
<tr>
<td>SocketDir</td>
<td>no</td>
<td>/tmp</td>
<td></td>
<td>Directory to store IPC sockets used by internal Zabbix services. This parameter is supported since Zabbix 3.4.0.</td>
</tr>
<tr>
<td>SourceIP</td>
<td>no</td>
<td></td>
<td></td>
<td>Source IP address for: - outgoing connections to Zabbix server; - agentless connections (VMware, SSH, JMX, SNMP, Telnet and simple checks); - HTTP agent connections; - script item JavaScript HTTP requests; - preprocessing JavaScript HTTP requests; - connections to the Vault</td>
</tr>
<tr>
<td>SSHKeyLocation</td>
<td>no</td>
<td></td>
<td></td>
<td>Location of public and private keys for SSH checks and actions. This parameter is used in web monitoring only.</td>
</tr>
<tr>
<td>SSLCertLocation</td>
<td>no</td>
<td></td>
<td></td>
<td>Location of SSL client certificate files for client authentication. This parameter is used in web monitoring only.</td>
</tr>
<tr>
<td>SSLKeyLocation</td>
<td>no</td>
<td></td>
<td></td>
<td>Location of SSL private key files for client authentication. This parameter is used in web monitoring only.</td>
</tr>
<tr>
<td>SSLCA Location</td>
<td>no</td>
<td></td>
<td></td>
<td>Location of certificate authority (CA) files for SSL server certificate verification. Note that the value of this parameter will be set as libcurl option CURLOPT_CAPATH. For libcurl versions before 7.42.0, this only has effect if libcurl was compiled to use OpenSSL. For more information see cURL web page. This parameter is used in web monitoring since Zabbix 2.4.0 and in SMTP authentication since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>StartDBSyncers</td>
<td>no</td>
<td>1-100</td>
<td>4</td>
<td>Number of pre-forked instances of history syncers. Note: Be careful when changing this value, increasing it may do more harm than good.</td>
</tr>
<tr>
<td>StartDiscoverers</td>
<td>no</td>
<td>0-250</td>
<td>1</td>
<td>Number of pre-forked instances of discoverers.</td>
</tr>
<tr>
<td>StartHTTPPollers</td>
<td>no</td>
<td>0-1000</td>
<td>1</td>
<td>Number of pre-forked instances of HTTP pollers.</td>
</tr>
<tr>
<td>StartIPMIPollers</td>
<td>no</td>
<td>0-1000</td>
<td>0</td>
<td>Number of pre-forked instances of IPMI pollers.</td>
</tr>
<tr>
<td>StartJavaPollers</td>
<td>no</td>
<td>0-1000</td>
<td>0</td>
<td>Number of pre-forked instances of Java pollers.</td>
</tr>
<tr>
<td>StartODBCPollers</td>
<td>no</td>
<td>0-1000</td>
<td>1</td>
<td>Number of pre-forked instances of ODBC pollers.</td>
</tr>
<tr>
<td>StartPingers</td>
<td>no</td>
<td>0-1000</td>
<td>1</td>
<td>Number of pre-forked instances of ICMP pingers.</td>
</tr>
<tr>
<td>StartPollersUnreachable</td>
<td>no</td>
<td>0-1000</td>
<td>1</td>
<td>Number of pre-forked instances of pollers for unreachable hosts (including IPMI and Java). At least one poller for unreachable hosts must be running if regular, IPMI or Java pollers are started.</td>
</tr>
<tr>
<td>StartPollers</td>
<td>no</td>
<td>0-1000</td>
<td>5</td>
<td>Number of pre-forked instances of pollers.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------</td>
<td>---------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>StartPreprocessors</td>
<td>no</td>
<td>1-1000</td>
<td>3</td>
<td>Number of pre-forked instances of preprocessing <strong>workers</strong>(^1). The preprocessing manager process is automatically started when a preprocessor worker is started. This parameter is supported since Zabbix 4.2.0.</td>
</tr>
<tr>
<td>StartSNMPTrappers</td>
<td>no</td>
<td>0-1</td>
<td>0</td>
<td>If set to 1, <strong>SNMP trap</strong> process will be started. *<em>()*</em></td>
</tr>
<tr>
<td>StartTrappers</td>
<td>no</td>
<td>0-1000</td>
<td>5</td>
<td>Number of pre-forked instances of <strong>trappers</strong>. <strong>Trappers</strong> accept incoming connections from Zabbix sender and active agents.</td>
</tr>
<tr>
<td>StartVMwareCollectors</td>
<td>no</td>
<td>0-250</td>
<td>0</td>
<td>Number of pre-forked <strong>VMware collector</strong> instances. *<em>()*</em></td>
</tr>
<tr>
<td>StatsAllowedIP</td>
<td>no</td>
<td></td>
<td></td>
<td>List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of external Zabbix instances. Stats request will be accepted only from the addresses listed here. If this parameter is not set no stats requests will be accepted. If IPv6 support is enabled then ‘127.0.0.1’, ‘::127.0.0.1’, ‘::ffff:127.0.0.1’ are treated equally and ‘::/0’ will allow any IPv4 or IPv6 address. ‘0.0.0.0/0’ can be used to allow any IPv4 address. Example: StatsAllowedIP=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com This parameter is supported since Zabbix 4.2.0.</td>
</tr>
<tr>
<td>Timeout</td>
<td>no</td>
<td>1-30</td>
<td>3</td>
<td>Specifies how long we wait for agent, SNMP device or external check (in seconds). *<em>()*</em></td>
</tr>
<tr>
<td>TLSAccept</td>
<td>yes for passive proxy, if TLS certificate or PSK parameters are defined (even for unencrypted connection), otherwise no</td>
<td></td>
<td></td>
<td>What incoming connections to accept from Zabbix server. Used for a passive proxy, ignored on an active proxy. Multiple values can be specified, separated by comma: unencrypted - accept connections without encryption (default) psk - accept connections with TLS and a pre-shared key (PSK) cert - accept connections with TLS and a certificate This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSCAFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSCertFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the proxy certificate or certificate chain, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSCipherAll</td>
<td>no</td>
<td></td>
<td></td>
<td>GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption. Example: TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256 This parameter is supported since Zabbix 4.4.7.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
<td>-------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>TLSCipherCert13</td>
<td>no</td>
<td></td>
<td></td>
<td>Cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate-based encryption. This parameter is supported since Zabbix 4.4.7.</td>
</tr>
<tr>
<td>TLSCipherPSK13</td>
<td>no</td>
<td></td>
<td></td>
<td>Cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for PSK-based encryption. Example: TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256 This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSConnect</td>
<td>yes for active proxy, if TLS certificate or PSK parameters are defined (even for unencrypted connection), otherwise no</td>
<td></td>
<td></td>
<td>How the proxy should connect to Zabbix server. Used for an active proxy, ignored on a passive proxy. Only one value can be specified: unencrypted - connect without encryption (default) psk - connect using TLS and a pre-shared key (PSK) cert - connect using TLS and a certificate This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSCRLFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSKeyFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the proxy private key, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSPSKFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the proxy pre-shared key, used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSPSKIdentity</td>
<td>no</td>
<td></td>
<td></td>
<td>Pre-shared key identity string, used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSServerCertIssuer</td>
<td></td>
<td></td>
<td></td>
<td>Allowed server certificate issuer. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSServerCertSubject</td>
<td></td>
<td></td>
<td></td>
<td>Allowed server certificate subject. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TmpDir</td>
<td>no</td>
<td>/tmp</td>
<td></td>
<td>Temporary directory.</td>
</tr>
<tr>
<td>TrapperTimeout</td>
<td>no</td>
<td>1-300</td>
<td>300</td>
<td>Specifies how many seconds trapper may spend processing new data.</td>
</tr>
<tr>
<td>User</td>
<td>no</td>
<td></td>
<td>zabbix</td>
<td>Drop privileges to a specific, existing user on the system. Only has effect if run as ‘root’ and AllowRoot is disabled.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UnavailableDelay</td>
<td>no</td>
<td>1-3600</td>
<td>60</td>
<td>How often host is checked for availability during the unavailable period, in seconds.</td>
</tr>
<tr>
<td>UnreachableDelay</td>
<td>no</td>
<td>1-3600</td>
<td>15</td>
<td>How often host is checked for availability during the unreachability period, in seconds.</td>
</tr>
<tr>
<td>UnreachablePeriod</td>
<td>no</td>
<td>1-3600</td>
<td>45</td>
<td>After how many seconds of unreachability treat a host as unavailable.</td>
</tr>
<tr>
<td>Vault</td>
<td>no</td>
<td>HashiCorp</td>
<td></td>
<td>Specifies secret management tool:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HashiCorp - HashiCorp KV Secrets Engine version 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CyberArk - CyberArk Central Credential Provider</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Must match the vault provider set in the frontend.</td>
</tr>
<tr>
<td>VaultDBPath</td>
<td>no</td>
<td></td>
<td></td>
<td>Specifies a location, from where database credentials should be retrieved by keys. Depending on the vault, can be vault path or query.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Keys used for HashiCorp are ‘password’ and ‘username’.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: secret/zabbix/database</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Keys used for CyberArk are ‘Content’ and ‘UserName’.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: AppID=zabbix_server&amp;Query=Safe=passwordSafe;Object=zabbix_proxy_database</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This option can only be used if DBUser and DBPassword are not specified.</td>
</tr>
<tr>
<td>VaultTLSCertFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Name of the SSL certificate file used for client authentication</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The certificate file must be in PEM1 format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If the certificate file contains also the private key, leave the SSL key file field empty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The directory containing this file is specified by the configuration parameter SSLCertLocation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This option can be omitted but is recommended for CyberArkCCP vault.</td>
</tr>
<tr>
<td>VaultTLSKeyFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Name of the SSL private key file used for client authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The private key file must be in PEM1 format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The directory containing this file is specified by the configuration parameter SSLKeyLocation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This option can be omitted but is recommended for CyberArkCCP vault.</td>
</tr>
<tr>
<td>VaultToken</td>
<td>yes, if Vault is set to HashiCorp, otherwise no</td>
<td></td>
<td></td>
<td>HashiCorp Vault authentication token that should have been generated exclusively for Zabbix proxy with read-only permission to the path specified in the optional VaultDBPath configuration parameter. It is an error if VaultToken and VAULT_TOKEN environment variable are defined at the same time.</td>
</tr>
<tr>
<td>VaultURL</td>
<td>no</td>
<td><a href="https://127.0.0.1/v2/vsphere">https://127.0.0.1/v2/vsphere</a> HTTP[S] URL. System-wide CA certificates directory will be used if SSLCAlocation is not specified.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VMwareCacheSize</td>
<td>no</td>
<td>256K-2G</td>
<td>8M</td>
<td>Shared memory size for storing VMware data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A VMware internal check zabbix{vmware,buffer,...} can be used to monitor the VMware cache usage (see Internal checks). Note that shared memory is not allocated if there are no vmware collector instances configured to start.</td>
</tr>
<tr>
<td>VMwareFrequency</td>
<td></td>
<td>10-86400</td>
<td>60</td>
<td>Delay in seconds between data gathering from a single VMware service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This delay should be set to the least update interval of any VMware monitoring item.</td>
</tr>
<tr>
<td>VMwarePerfFrequency</td>
<td>no</td>
<td>10-86400</td>
<td>60</td>
<td>Delay in seconds between performance counter statistics retrieval from a single VMware service.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This delay should be set to the least update interval of any VMware monitoring item that uses VMware performance counters.</td>
</tr>
<tr>
<td>VMwareTimeout</td>
<td></td>
<td>1-300</td>
<td>10</td>
<td>The maximum number of seconds vmware collector will wait for a response from VMware service (vCenter or ESX hypervisor).</td>
</tr>
</tbody>
</table>
3 Zabbix agent (UNIX)

Overview

This section lists parameters supported in a Zabbix agent configuration file (zabbix_agentd.conf).

Note that:

- The default values reflect daemon defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported in the beginning of the line.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>no</td>
<td></td>
<td></td>
<td>Sets an alias for an item key. It can be used to substitute long and complex item key with a smaller and simpler one. Multiple Alias parameters may be present. Multiple parameters with the same Alias key are allowed. Different Alias keys may reference the same item key. Aliases can be used in HostMetadataItem but not in HostnameItem parameters.</td>
</tr>
<tr>
<td>AllowKey</td>
<td>no</td>
<td></td>
<td></td>
<td>Allow execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports &quot;*&quot; character to match any number of any characters. Multiple key matching rules may be defined in combination with DenyKey. The parameters are processed one by one according to their appearance order. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.</td>
</tr>
<tr>
<td>AllowRoot</td>
<td>no</td>
<td></td>
<td>0</td>
<td>Allow the agent to run as 'root'. If disabled and the agent is started by 'root', the agent will try to switch to user 'zabbix' instead. Has no effect if started under a regular user.</td>
</tr>
<tr>
<td>BufferSend</td>
<td>no</td>
<td>1-3600</td>
<td>5</td>
<td>Do not keep data longer than N seconds in buffer.</td>
</tr>
<tr>
<td>BufferSize</td>
<td>no</td>
<td>2-65535</td>
<td>100</td>
<td>Maximum number of values in a memory buffer. The agent will send all collected data to Zabbix server or proxy if the buffer is full.</td>
</tr>
</tbody>
</table>

Examples:

1. Retrieving the ID of user ‘zabbix’.
   Alias=zabbix.userid:vfs.file.regexp[/etc/passwd,"^zabbix:::([0-9]+)""]
   Now shorthand key zabbix.userid may be used to retrieve data.

2. Getting CPU utilization with default and custom parameters.
   Alias=cpu.util:system.cpu.util
   Alias=cpu.util[*]:system.cpu.util[*]
   This allows use cpu.util key to get CPU utilization percentage with default parameters as well as use cpu.util[all, idle, avg15] to get specific data about CPU utilization.

3. Running multiple low-level discovery rules processing the same discovery items.
   Alias=vfs.fs.discovery[*]:vfs.fs.discovery
   Now it is possible to set up several discovery rules using vfs.fs.discovery with different parameters for each rule, e.g., vfs.fs.discovery[foo], vfs.fs.discovery[bar], etc.
   Allow execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports "*" character to match any number of any characters. Multiple key matching rules may be defined in combination with DenyKey. The parameters are processed one by one according to their appearance order. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| DebugLevel         | no        | 0-5      | 3       | Specifies debug level:  
|                    |           |          | 0 - basic information about starting and stopping of Zabbix processes  
|                    |           |          | 1 - critical information  
|                    |           |          | 2 - error information  
|                    |           |          | 3 - warnings  
|                    |           |          | 4 - for debugging (produces lots of information)  
|                    |           |          | 5 - extended debugging (produces even more information)  
| DenyKey            | no        |          |         | Deny execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports “*” character to match any number of any characters.  
|                    |           |          |         | Multiple key matching rules may be defined in combination with AllowKey. The parameters are processed one by one according to their appearance order.  
|                    |           |          |         | This parameter is supported since Zabbix 5.0.0.  
|                    |           |          |         | See also: Restricting agent checks.  
| EnableRemoteCommands |          |          | 0       | Whether remote commands from Zabbix server are allowed. This parameter is deprecated, use AllowKey=system.run[*] or DenyKey=system.run[*] instead  
|                    |           |          |         | It is internal alias for AllowKey/DenyKey parameters depending on value: 0 - DenyKey=system.run[*]  
|                    |           |          |         | 1 - AllowKey=system.run[*]  
| HeartbeatFrequency |           | 0-3600   | 60      | Frequency of heartbeat messages in seconds. Used for monitoring the availability of active checks.  
|                    |           |          |         | 0 - heartbeat messages disabled.  
| HostInterface      | no        | 0-255    |         | Optional parameter that defines host interface.  
|                    |           |          |         | Host interface is used at host autoregistration process.  
|                    |           |          |         | An agent will issue an error and not start if the value is over the limit of 255 characters.  
|                    |           |          |         | If not defined, value will be acquired from HostInterfaceItem.  
|                    |           |          |         | Supported since Zabbix 4.4.0.  
| HostInterfaceItem  |           |          |         | Optional parameter that defines an item used for getting host interface.  
|                    |           |          |         | Host interface is used at host autoregistration process.  
|                    |           |          |         | During an autoregistration request an agent will log a warning message if the value returned by specified item is over limit of 255 characters.  
|                    |           |          |         | This option is only used when HostInterface is not defined.  
|                    |           |          |         | Supported since Zabbix 4.4.0.  
| HostMetadata       | no        | 0-255    |         | Optional parameter that defines host metadata. Host metadata is used only at host autoregistration process (active agent).  
|                    |           |          |         | If not defined, the value will be acquired from HostMetadataItem.  
|                    |           |          |         | An agent will issue an error and not start if the specified value is over the limit or a non-UTF-8 string.  
| HostMetadataItem   |           |          |         | Optional parameter that defines a Zabbix agent item used for getting host metadata. This option is only used when HostMetadata is not defined.  
|                    |           |          |         | Supports UserParameters and aliases. Supports system.run[] regardless of AllowKey/DenyKey values.  
|                    |           |          |         | HostMetadataItem value is retrieved on each autoregistration attempt and is used only at host autoregistration process (active agent).  
|                    |           |          |         | During an autoregistration request an agent will log a warning message if the value returned by the specified item is over the limit of 255 characters.  
|                    |           |          |         | The value returned by the item must be a UTF-8 string otherwise it will be ignored.  


<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hostname</td>
<td>no</td>
<td>Set by HostnameItem</td>
<td></td>
<td>List of comma-delimited unique, case-sensitive hostnames. Required for active checks and must match hostnames as configured on the server. Value is acquired from HostnameItem if undefined. Allowed characters: alphanumeric, '.', ',', '_', and '-' . Maximum length: 128 characters per hostname, 2048 characters for the entire line.</td>
</tr>
<tr>
<td>HostnameItem</td>
<td>no</td>
<td>system.hostname</td>
<td></td>
<td>Optional parameter that defines a Zabbix agent item used for getting host name. This option is only used when Hostname is not defined. Does not support UserParameters or aliases, but does support system.run[] regardless of AllowKey/DenyKey values. The output length is limited to 512KB.</td>
</tr>
<tr>
<td>Include</td>
<td>no</td>
<td></td>
<td></td>
<td>You may include individual files or all files in a directory in the configuration file. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. For example: /absolute/path/to/config/files/*.conf. See special notes about limitations.</td>
</tr>
<tr>
<td>ListenBacklog</td>
<td>no</td>
<td>0 - INT_MAX SOMAXCONN</td>
<td></td>
<td>The maximum number of pending connections in the TCP queue. Default value is a hard-coded constant, which depends on the system. Maximum supported value depends on the system, too high values may be silently truncated to the 'implementation-specified maximum'.</td>
</tr>
<tr>
<td>ListenIP</td>
<td>no</td>
<td>0.0.0.0</td>
<td></td>
<td>List of comma delimited IP addresses that the agent should listen on. Multiple IP addresses are supported in version 1.8.3 and higher.</td>
</tr>
<tr>
<td>ListenPort</td>
<td>no</td>
<td>1024-32767 10050</td>
<td></td>
<td>Agent will listen on this port for connections from the server. Module to load at agent startup. Modules are used to extend functionality of the agent. Formats: LoadModule=&lt;module.so&gt; LoadModule=&lt;path/module.so&gt; LoadModule=/abs_path/module.so&gt; Either the module must be located in directory specified by LoadModulePath or the path must precede the module name. If the preceding path is absolute (starts with '/') then LoadModulePath is ignored. It is allowed to include multiple LoadModule parameters. Full path to location of agent modules. Default depends on compilation options. Name of log file.</td>
</tr>
<tr>
<td>LoadModulePath</td>
<td>no</td>
<td>Full path to location of agent modules. Default depends on compilation options. Name of log file.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogFile</td>
<td>yes, if LogType is set to file, otherwise no</td>
<td></td>
<td></td>
<td>Maximum size of log file in MB. 0 - disable automatic log rotation. Note: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.</td>
</tr>
<tr>
<td>LogFileSize</td>
<td>no</td>
<td>0-1024 1</td>
<td></td>
<td>Log output type: file - write log to file specified by LogFile parameter, system - write log to syslog, console - write log to standard output. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>LogType</td>
<td>no</td>
<td>file</td>
<td></td>
<td>Log output type: file - write log to file specified by LogFile parameter, system - write log to syslog, console - write log to standard output. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>--------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LogRemoteCommands</td>
<td>no</td>
<td>0</td>
<td>0</td>
<td>Enable logging of executed shell commands as warnings.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 - disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 - enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Commands will be logged only if executed remotely. Log entries will not be created if system.run[] is launched locally by HostMetadataItem, HostInterfaceItem or HostnameItem parameters.</td>
</tr>
<tr>
<td>MaxLinesPerSecond</td>
<td></td>
<td>1-1000</td>
<td>20</td>
<td>Maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log' and 'eventlog' active checks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The provided value will be overridden by the parameter 'maxlines', provided in 'log' or 'eventlog' item key.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Zabbix will process 10 times more new lines than set in MaxLinesPerSecond to seek the required string in log items.</td>
</tr>
<tr>
<td>PidFile</td>
<td>no</td>
<td>/tmp</td>
<td></td>
<td>Name of PID file.</td>
</tr>
<tr>
<td>RefreshActiveChecks</td>
<td></td>
<td>60-3600</td>
<td>120</td>
<td>How often list of active checks is refreshed, in seconds.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note that after failing to refresh active checks the next refresh will be attempted after 60 seconds.</td>
</tr>
<tr>
<td>Server</td>
<td>yes, if StartAgents is not explicitly set to 0</td>
<td></td>
<td></td>
<td>List of comma delimited IP addresses, optionally in CIDR notation, or hostnames of Zabbix servers and Zabbix proxies.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Incoming connections will be accepted only from the hosts listed here.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally and ':'/0 will allow any IPv4 or IPv6 address.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'0.0.0.0/0' can be used to allow any IPv4 address.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note, that &quot;IPv4-compatible IPv6 addresses&quot; (0000:/96 prefix) are supported but deprecated by RFC4291.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.domain</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Spaces are allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If port is not specified, default port is used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IPv6 addresses must be enclosed in square brackets if port for that host is specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If port is not specified, square brackets for IPv6 addresses are optional.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If this parameter is not specified, active checks are disabled.</td>
</tr>
<tr>
<td>ServerActive</td>
<td>no</td>
<td></td>
<td></td>
<td>Zabbix server/proxy address or cluster configuration to get active checks from.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Server/proxy address is IP address or DNS name and optional port separated by colon.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Cluster configuration is one or more server addresses separated by semicolon.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Multiple Zabbix servers/clusters and Zabbix proxies can be specified, separated by comma.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>More than one Zabbix proxy should not be specified from each Zabbix server/cluster.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If Zabbix proxy is specified then Zabbix server/cluster for that proxy should not be specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Multiple comma-delimited addresses can be provided to use several independent Zabbix servers in parallel. Spaces are allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If port is not specified, default port is used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IPv6 addresses must be enclosed in square brackets if port for that host is specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If port is not specified, square brackets for IPv6 addresses are optional.</td>
</tr>
</tbody>
</table>

Example for Zabbix proxy:
ServerActive=127.0.0.1:10051

Example for multiple servers:
ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| SourceIP           | no        |         |         | Source IP address for:  
- outgoing connections to Zabbix server or Zabbix proxy;  
- making connections while executing some items (web.page.get, net.tcp.port, etc.)  
Number of pre-forked instances of zabbix_agentd that process passive checks.  
If set to 0, disables passive checks and the agent will not listen on any TCP port.                                                                                                                                                      |
| StartAgents        | no        | 0-100   | 3       | Number of pre-forked instances of zabbix_agentd that process passive checks.  
If set to 0, disables passive checks and the agent will not listen on any TCP port.                                                                                                                                                                                      |
| Timeout            | no        | 1-30    | 3       | Spend no more than Timeout seconds on processing.  
What incoming connections to accept. Used for a passive checks. Multiple values can be specified, separated by comma:  
unencrypted - accept connections without encryption (default)  
psk - accept connections with TLS and a pre-shared key (PSK)  
cert - accept connections with TLS and a certificate  
This parameter is supported since Zabbix 3.0.0.                                                                                                                  |
| TLSAccept          | yes       |         |         | What incoming connections to accept. Used for a passive checks. Multiple values can be specified, separated by comma:  
unencrypted - accept connections without encryption (default)  
psk - accept connections with TLS and a pre-shared key (PSK)  
cert - accept connections with TLS and a certificate  
This parameter is supported since Zabbix 3.0.0.                                                                                                                  |
| TLSCAFile           | no        |         |         | Full path name of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.  
This parameter is supported since Zabbix 3.0.0.                                                                                                                  |
| TLSCertFile         | no        |         |         | Full path name of a file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components.  
This parameter is supported since Zabbix 3.0.0.                                                                                                                  |
| TLSCipherAll        | no        |         |         | GnuTLS priority string or OpenSSL (TLS 1.2) cipher string.  
Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.  
Example:  
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:  
This parameter is supported since Zabbix 4.4.7.                                                                                                                  |
| TLSCipherAll13      | no        |         |         | Cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.  
Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128:EECDHEPSK+AES128:kPSK+AES128:EECDHEPSK+AES128:kPSK  
This parameter is supported since Zabbix 4.4.7.                                                                                                                  |
| TLSCipherCert       | no        |         |         | GnuTLS priority string or OpenSSL (TLS 1.2) cipher string.  
Override the default ciphersuite selection criteria for certificate-based encryption.  
Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128:EECDHEPSK+AES128:kPSK+AES128:EECDHEPSK+AES128:kPSK  
This parameter is supported since Zabbix 4.4.7.                                                                                                                  |
| TLSCipherCert13      | no        |         |         | Cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate-based encryption.  
This parameter is supported since Zabbix 4.4.7.                                                                                                                  |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLSCipherPSK13</td>
<td>no</td>
<td></td>
<td></td>
<td>Cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for PSK-based encryption. Example: TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256 This parameter is supported since Zabbix 4.4.7.</td>
</tr>
<tr>
<td>TLSConnect</td>
<td>yes, if TLS certificate or PSK parameters are defined (even for unencrypted connection), otherwise no</td>
<td></td>
<td></td>
<td>How the agent should connect to Zabbix server or proxy. Used for active checks. Only one value can be specified: unencrypted - connect without encryption (default) psk - connect using TLS and a pre-shared key (PSK) cert - connect using TLS and a certificate This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSCRLFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSKeyFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the agent private key used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSPSKFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the agent pre-shared key used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSPSKIdentity</td>
<td>no</td>
<td></td>
<td></td>
<td>Pre-shared key identity string, used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSServerCertIssuer</td>
<td>no</td>
<td></td>
<td></td>
<td>Allowed server (proxy) certificate issuer. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSServerCertSubject</td>
<td>no</td>
<td></td>
<td></td>
<td>Allowed server (proxy) certificate subject. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>UnsafeUserParameters</td>
<td>0,1</td>
<td>0</td>
<td>0</td>
<td>Allow all characters to be passed in arguments to user-defined parameters. 0 - do not allow 1 - allow The following characters are not allowed: &quot;'&quot;*?[]{}~$!&amp;;()&gt;|#@ Additionally, newline characters are not allowed.</td>
</tr>
<tr>
<td>User</td>
<td>no</td>
<td></td>
<td>zabbix</td>
<td>Drop privileges to a specific, existing user on the system. Only has effect if run as ‘root’ and AllowRoot is disabled.</td>
</tr>
<tr>
<td>UserParameter</td>
<td>no</td>
<td></td>
<td></td>
<td>User-defined parameter to monitor. There can be several user-defined parameters. Format: UserParameter=&lt;key&gt;,&lt;shell command&gt; Note that shell command must not return empty string or EOL only. Shell commands may have relative paths, if UserParameterDir parameter is specified. Examples: UserParameter=system.test,who</td>
</tr>
</tbody>
</table>
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserParameterDir</td>
<td>no</td>
<td></td>
<td></td>
<td>Default search path for UserParameter commands. If used, the agent will change its working directory to the one specified here before executing a command. Thereby, UserParameter commands can have a relative ./ prefix instead of a full path. Only one entry is allowed. Example: UserParameterDir=/opt/myscripts</td>
</tr>
</tbody>
</table>

See also

1. Differences in the Zabbix agent configuration for active and passive checks starting from version 2.0.0

## 4 Zabbix agent 2 (UNIX)

### Overview

Zabbix agent 2 is a new generation of Zabbix agent and may be used in place of Zabbix agent.

This section lists parameters supported in a Zabbix agent 2 configuration file (zabbix_agent2.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported in the beginning of the line.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| Alias     | no        |       |         | Sets an alias for an item key. It can be used to substitute long and complex item key with a smaller and simpler one. Multiple Alias parameters may be present. Multiple parameters with the same Alias key are allowed. Different Alias keys may reference the same item key. Aliases can be used in HostMetadataItem but not in HostnameItem parameters. Examples:  
1. Retrieving the ID of user ‘zabbix’.  
Now shorthand key zabbix.userid may be used to retrieve data.  
2. Getting CPU utilization with default and custom parameters.  
Alias=cpu.util:system.cpu.util  
Alias=cpu.util[*]:system.cpu.util[*]  
This allows use cpu.util key to get CPU utilization percentage with default parameters as well as use cpu.util[all, idle, avg15] to get specific data about CPU utilization.  
3. Running multiple low-level discovery rules processing the same discovery items.  
Alias=vfs.fs.discovery[*]:vfs.fs.discovery  
Now it is possible to set up several discovery rules using vfs.fs.discovery with different parameters for each rule, e.g., vfs.fs.discovery[foo], vfs.fs.discovery[bar], etc. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowKey</td>
<td>no</td>
<td></td>
<td></td>
<td>Allow execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports &quot;*&quot; character to match any number of any characters. Multiple key matching rules may be defined in combination with DenyKey. The parameters are processed one by one according to their appearance order. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.</td>
</tr>
<tr>
<td>BufferSend</td>
<td>no</td>
<td>1-3600</td>
<td>5</td>
<td>The time interval in seconds which determines how often values are sent from the buffer to Zabbix server. Note, that if the buffer is full, the data will be sent sooner.</td>
</tr>
<tr>
<td>BufferSize</td>
<td>no</td>
<td>2-65535</td>
<td>100</td>
<td>Maximum number of values in a memory buffer. The agent will send all collected data to Zabbix server or proxy if the buffer is full. This parameter should only be used if persistent buffer is disabled (EnablePersistentBuffer=0).</td>
</tr>
<tr>
<td>ControlSocket</td>
<td>no</td>
<td>/tmp/agent.sock</td>
<td></td>
<td>The control socket, used to send runtime commands with <code>-R</code> option.</td>
</tr>
<tr>
<td>DebugLevel</td>
<td>no</td>
<td>0-5</td>
<td>3</td>
<td>Specifies debug level: 0 - basic information about starting and stopping of Zabbix processes 1 - critical information 2 - error information 3 - warnings 4 - for debugging (produces lots of information) 5 - extended debugging (produces even more information)</td>
</tr>
<tr>
<td>DenyKey</td>
<td>no</td>
<td></td>
<td></td>
<td>Deny execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports &quot;*&quot; character to match any number of any characters. Multiple key matching rules may be defined in combination with AllowKey. The parameters are processed one by one according to their appearance order. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.</td>
</tr>
<tr>
<td>EnablePersistentBuffer</td>
<td>0-1</td>
<td>0</td>
<td></td>
<td>Enable usage of local persistent storage for active items. 0 - disabled 1 - enabled If persistent storage is disabled, the memory buffer will be used.</td>
</tr>
<tr>
<td>ForceActiveChecksOnStart</td>
<td>0-1</td>
<td>0</td>
<td></td>
<td>Perform active checks immediately after restart for the first received configuration. 0 - disabled 1 - enabled Also available as per plugin configuration parameter, for example: Plugins.Uptime.System.ForceActiveChecksOnStart=1</td>
</tr>
<tr>
<td>HeartbeatFrequency</td>
<td>0-3600</td>
<td>60</td>
<td></td>
<td>Frequency of heartbeat messages in seconds. Used for monitoring the availability of active checks. 0 - heartbeat messages disabled.</td>
</tr>
<tr>
<td>HostInterface</td>
<td>no</td>
<td>0-255 characters</td>
<td></td>
<td>Optional parameter that defines host interface. Host interface is used at host autoregistration process. An agent will issue an error and not start if the value is over the limit of 255 characters. If not defined, value will be acquired from HostInterfaceItem. Supported since Zabbix 4.4.0.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------</td>
<td>-------------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HostInterfaceItem</td>
<td>no</td>
<td></td>
<td></td>
<td>Optional parameter that defines an item used for getting host interface. Host interface is used at host autoregistration process. During an autoregistration request an agent will log a warning message if the value returned by specified item is over limit of 255 characters. This option is only used when HostInterface is not defined. Supported since Zabbix 4.4.0.</td>
</tr>
<tr>
<td>HostMetadata</td>
<td>no</td>
<td>0-255</td>
<td>characters</td>
<td>Optional parameter that defines host metadata. Host metadata is used at host autoregistration process. An agent will issue an error and not start if the specified value is over the limit or a non-UTF-8 string. If not defined, the value will be acquired from HostMetadataItem.</td>
</tr>
<tr>
<td>HostMetadataItem</td>
<td>no</td>
<td></td>
<td></td>
<td>Optional parameter that defines an item used for getting host metadata. Host metadata item value is retrieved on each autoregistration attempt for host autoregistration process. During an autoregistration request an agent will log a warning message if the value returned by the specified item is over the limit of 255 characters. This option is only used when HostMetadata is not defined. Supports UserParameters and aliases. Supports system.run[] regardless of AllowKey/DenyKey values. The value returned by the item must be a UTF-8 string otherwise it will be ignored. Supports UserParameters and aliases. Supports system.run[] regardless of AllowKey/DenyKey values. The value returned by the item must be a UTF-8 string otherwise it will be ignored.</td>
</tr>
<tr>
<td>Hostname</td>
<td>no</td>
<td></td>
<td>Set by HostnameItem</td>
<td>List of comma-delimited unique, case-sensitive hostnames. Required for active checks and must match hostnames as configured on the server. Value is acquired from HostnameItem if undefined. Allowed characters: alphanumeric, '.', ',', '_' and '-' Maximum length: 128 characters per hostname, 2048 characters for the entire line.</td>
</tr>
<tr>
<td>Hostnametag</td>
<td>no</td>
<td></td>
<td>system.hostname</td>
<td>Used for generating Hostname if it is not defined. Ignored if Hostname is defined. Does not support UserParameters or aliases, but does support system.run[] regardless of AllowKey/DenyKey values. The output length is limited to 512KB.</td>
</tr>
<tr>
<td>Include</td>
<td>no</td>
<td></td>
<td></td>
<td>You may include individual files or all files in a directory in the configuration file. During the installation Zabbix will create the include directory in /usr/local/etc, unless modified during the compile time. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. For example: /absolute/path/to/config/files/*.conf. Since Zabbix 6.0.0 a path can be relative to zabbix_agent2.conf file location. See special notes about limitations.</td>
</tr>
<tr>
<td>ListenIP</td>
<td>no</td>
<td>0.0.0.0</td>
<td></td>
<td>List of comma-delimited IP addresses that the agent should listen on. The first IP address is sent to Zabbix server, if connecting to it, to retrieve the list of active checks.</td>
</tr>
<tr>
<td>ListenPort</td>
<td>no</td>
<td>1024-32767</td>
<td>10050</td>
<td>Agent will listen on this port for connections from the server.</td>
</tr>
<tr>
<td>LogFile</td>
<td>yes, if LogType is set to file, otherwise no</td>
<td>/tmp/zabbix_agent2.log</td>
<td></td>
<td>/tmp/zabbix_agent2.log if LogType is ‘file’.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
<td>-----------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LogFileSize</td>
<td>no</td>
<td>0-1024</td>
<td>1</td>
<td>Maximum size of log file in MB. 0 - disable automatic log rotation. Note: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.</td>
</tr>
<tr>
<td>LogType</td>
<td>no</td>
<td>file</td>
<td></td>
<td>Specifies where log messages are written to: system - syslog, file - file specified by LogFile parameter, console - standard output.</td>
</tr>
<tr>
<td>PersistentBufferFile</td>
<td>no</td>
<td></td>
<td></td>
<td>The file, where Zabbix Agent2 should keep SQLite database. Must be a full filename. This parameter is only used if persistent buffer is enabled (EnablePersistentBuffer=1).</td>
</tr>
<tr>
<td>PersistentBufferPeriod</td>
<td>1m-365d</td>
<td>1h</td>
<td></td>
<td>The time period for which data should be stored, when there is no connection to the server or proxy. Older data will be lost. Log data will be preserved. This parameter is only used if persistent buffer is enabled (EnablePersistentBuffer=1).</td>
</tr>
<tr>
<td>PidFile</td>
<td>no</td>
<td>/tmp/zabbix_agent2.pid</td>
<td></td>
<td>Name of PID file.</td>
</tr>
<tr>
<td>Plugin</td>
<td>no</td>
<td></td>
<td></td>
<td>Since Zabbix 6.0.0 most of the plugins have their own configuration files. The agent configuration file contains plugin parameters listed below.</td>
</tr>
<tr>
<td>Plugins.Log.MaxLinesPerSecond</td>
<td>0-1000</td>
<td>20</td>
<td></td>
<td>Maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log' and 'eventlog' active checks. The provided value will be overridden by the parameter 'maxlines', provided in 'log' or 'eventlog' item key. Note: Zabbix will process 10 times more new lines than set in MaxLinesPerSecond to seek the required string in log items. This parameter is supported since 4.4.2 and replaces MaxLinesPerSecond.</td>
</tr>
<tr>
<td>Plugins.System.Run.LogRemoteCommands</td>
<td>no</td>
<td>0</td>
<td></td>
<td>Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled Commands will be logged only if executed remotely. Log entries will not be created if system.run[] is launched locally by HostMetadataItem, HostInterfaceItem or HostnameItem parameters. This parameter is supported since 4.4.2 and replaces LogRemoteCommands.</td>
</tr>
<tr>
<td>PluginSocket</td>
<td>no</td>
<td>/tmp/agent.plugin.sock</td>
<td></td>
<td>Unix socket for loadable plugin communications.</td>
</tr>
<tr>
<td>PluginTimeout</td>
<td>no</td>
<td>1-30</td>
<td></td>
<td>Global timeout for connection with loadable plugins.</td>
</tr>
<tr>
<td>RefreshActiveChecks</td>
<td>60-3600</td>
<td>120</td>
<td></td>
<td>How often the list of active checks is refreshed, in seconds. Note that after failing to refresh active checks the next refresh will be attempted after 60 seconds.</td>
</tr>
<tr>
<td>Server</td>
<td>yes</td>
<td></td>
<td></td>
<td>List of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and Zabbix proxies. Incoming connections will be accepted only from the hosts listed here. If IPv6 support is enabled then '127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address. Example: Server=127.0.0.1,192.168.1.0/24,:1,2001:db8::/32,zabbix.example.com Spaces are allowed.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-----------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ServerActive</td>
<td>no</td>
<td></td>
<td></td>
<td>Zabbix server/proxy address or cluster configuration to get active checks from.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Server/proxy address is IP address or DNS name and optional port separated by colon.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Cluster configuration is one or more server addresses separated by semicolon.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Multiple Zabbix servers/clusters and Zabbix proxies can be specified, separated by comma.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>More than one Zabbix proxy should not be specified from each Zabbix server/cluster.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If Zabbix proxy is specified then Zabbix server/cluster for that proxy should not be specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Multiple comma-delimited addresses can be provided to use several independent Zabbix servers in parallel. Spaces are allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If port is not specified, default port is used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IPv6 addresses must be enclosed in square brackets if port for that host is specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If port is not specified, square brackets for IPv6 addresses are optional.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If this parameter is not specified, active checks are disabled.</td>
</tr>
<tr>
<td>Example for Zabbix proxy:</td>
<td></td>
<td></td>
<td></td>
<td>ServerActive=127.0.0.1:10051</td>
</tr>
<tr>
<td>Example for multiple servers:</td>
<td></td>
<td></td>
<td></td>
<td>ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051::1,[:12f0::1]</td>
</tr>
<tr>
<td>Example for high availability:</td>
<td></td>
<td></td>
<td></td>
<td>ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster3</td>
</tr>
<tr>
<td>Example for high availability with two clusters and one server:</td>
<td></td>
<td></td>
<td></td>
<td>ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.cluster2.node1;zabbix.cluster2.node2,zabbix.domain</td>
</tr>
<tr>
<td>SourceIP</td>
<td>no</td>
<td></td>
<td></td>
<td>Source IP address for:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- outgoing connections to Zabbix server or Zabbix proxy;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- making connections while executing some items (web.page.get, net.tcp.port, etc.)</td>
</tr>
<tr>
<td>StatusPort</td>
<td>no</td>
<td>1024-32767</td>
<td></td>
<td>If set, agent will listen on this port for HTTP status requests (<a href="http://localhost">http://localhost</a>:&lt;port&gt;/status).</td>
</tr>
<tr>
<td>Timeout</td>
<td>no</td>
<td>1-30</td>
<td>3</td>
<td>Spend no more than Timeout seconds on processing.</td>
</tr>
<tr>
<td>TLSAccept</td>
<td>yes, if TLS certificate or PSK parameters are defined (even for unencrypted connection), otherwise no</td>
<td></td>
<td></td>
<td>What incoming connections to accept. Used for a passive checks. Multiple values can be specified, separated by comma:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>unencrypted - accept connections without encryption (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>psk - accept connections with TLS and a pre-shared key (PSK)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>cert - accept connections with TLS and a certificate</td>
</tr>
<tr>
<td>TLSCAFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.</td>
</tr>
<tr>
<td>TLSCertFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------</td>
<td>----------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TLSConnect</td>
<td>yes, if TLS certificate or PSK parameters are defined (even for unencrypted connection), otherwise no</td>
<td></td>
<td></td>
<td>How the agent should connect to Zabbix server or proxy. Used for active checks. Only one value can be specified: unencrypted - connect without encryption (default) psk - connect using TLS and a pre-shared key (PSK) cert - connect using TLS and a certificate</td>
</tr>
<tr>
<td>TLSCRLFile</td>
<td>no</td>
<td></td>
<td>no</td>
<td>Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications with Zabbix components.</td>
</tr>
<tr>
<td>TLSKeyFile</td>
<td>no</td>
<td></td>
<td>no</td>
<td>Full pathname of a file containing the agent private key used for encrypted communications with Zabbix components.</td>
</tr>
<tr>
<td>TLSPSKFile</td>
<td>no</td>
<td></td>
<td>no</td>
<td>Full pathname of a file containing the agent pre-shared key used for encrypted communications with Zabbix components.</td>
</tr>
<tr>
<td>TLSPSKIdentity</td>
<td>no</td>
<td></td>
<td>no</td>
<td>Pre-shared key identity string, used for encrypted communications with Zabbix server.</td>
</tr>
<tr>
<td>TLSServerCertIssuer</td>
<td>no</td>
<td></td>
<td>no</td>
<td>Allowed server (proxy) certificate issuer.</td>
</tr>
<tr>
<td>TLSServerCertSubject</td>
<td>no</td>
<td></td>
<td>no</td>
<td>Allowed server (proxy) certificate subject.</td>
</tr>
<tr>
<td>UnsafeUserParameters</td>
<td>0,1</td>
<td>0</td>
<td>0</td>
<td>Allow all characters to be passed in arguments to user-defined parameters. The following characters are not allowed: &quot;'*?{}~$!&amp;</td>
</tr>
<tr>
<td>UserParameter</td>
<td>no</td>
<td></td>
<td>no</td>
<td>User-defined parameter to monitor. There can be several user-defined parameters. Format: UserParameter=&lt;key&gt;,&lt;shell command&gt; Note that shell command must not return empty string or EOL only. Shell commands may have relative paths, if UserParameterDir parameter is specified. Examples: UserParameter=system.test,who</td>
</tr>
<tr>
<td>UserParameterDir</td>
<td>no</td>
<td></td>
<td>no</td>
<td>Default search path for UserParameter commands. If used, the agent will change its working directory to the one specified here before executing a command. Thereby, UserParameter commands can have a relative ./ prefix instead of a full path. Only one entry is allowed. Example: UserParameterDir=/opt/myscripts</td>
</tr>
</tbody>
</table>

5 Zabbix agent (Windows)

Overview

This section lists parameters supported in a Zabbix agent (Windows) configuration file (zabbix_agent.conf).

Note that:

- The default values reflect daemon defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported in the beginning of the line.

Parameters
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>no</td>
<td></td>
<td></td>
<td>Sets an alias for an item key. It can be used to substitute long and complex item key with a smaller and simpler one. Multiple Alias parameters may be present. Multiple parameters with the same Alias key are allowed. Different Alias keys may reference the same item key. Aliases can be used in HostMetadataItem but not in HostnameItem or PerfCounter parameters.</td>
</tr>
<tr>
<td>AllowKey</td>
<td>no</td>
<td></td>
<td></td>
<td>Allow execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports &quot;*&quot; character to match any number of any characters. Multiple key matching rules may be defined in combination with DenyKey. The parameters are processed one by one according to their appearance order. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.</td>
</tr>
<tr>
<td>BufferSend</td>
<td>no</td>
<td>1-3600</td>
<td>5</td>
<td>Do not keep data longer than N seconds in buffer.</td>
</tr>
<tr>
<td>BufferSize</td>
<td>no</td>
<td>2-65535</td>
<td>100</td>
<td>Maximum number of values in a memory buffer. The agent will send all collected data to Zabbix server or proxy if the buffer is full.</td>
</tr>
<tr>
<td>DebugLevel</td>
<td>no</td>
<td>0-5</td>
<td>3</td>
<td>Specifies debug level: 0 - basic information about starting and stopping of Zabbix processes 1 - critical information 2 - error information 3 - warnings 4 - for debugging (produces lots of information) 5 - extended debugging (produces even more information)</td>
</tr>
<tr>
<td>DenyKey</td>
<td>no</td>
<td></td>
<td></td>
<td>Deny execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports &quot;*&quot; character to match any number of any characters. Multiple key matching rules may be defined in combination with AllowKey. The parameters are processed one by one according to their appearance order. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.</td>
</tr>
<tr>
<td>EnableRemoteCommands</td>
<td></td>
<td></td>
<td>0</td>
<td>Whether remote commands from Zabbix server are allowed. This parameter is deprecated, use AllowKey=system.run[<em>] or DenyKey=system.run[</em>] instead It is internal alias for AllowKey/DenyKey parameters depending on value: 0 - DenyKey=system.run[<em>] 1 - AllowKey=system.run[</em>].</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------</td>
<td>----------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HeartbeatFrequency</td>
<td>no</td>
<td>0-3600</td>
<td>60</td>
<td>Frequency of heartbeat messages in seconds. Used for monitoring the availability of active checks. 0 - heartbeat messages disabled.</td>
</tr>
<tr>
<td>HostInterface</td>
<td>no</td>
<td>0-255 characters</td>
<td></td>
<td>Optional parameter that defines host interface. Host interface is used at host autoregistration process. An agent will issue an error and not start if the value is over the limit of 255 characters. If not defined, value will be acquired from HostInterfaceItem. Supported since Zabbix 4.4.0.</td>
</tr>
<tr>
<td>HostInterfaceItem</td>
<td></td>
<td></td>
<td></td>
<td>Optional parameter that defines an item used for getting host interface. Host interface is used at host autoregistration process. During an autoregistration request an agent will log a warning message if the value returned by specified item is over limit of 255 characters. This option is only used when HostInterface is not defined. Supported since Zabbix 4.4.0.</td>
</tr>
<tr>
<td>HostMetadata</td>
<td>no</td>
<td>0-255 characters</td>
<td></td>
<td>Optional parameter that defines host metadata. Host metadata is used only at host autoregistration process (active agent). If not defined, the value will be acquired from HostMetadataItem. An agent will issue an error and not start if the specified value is over the limit or a non-UTF-8 string.</td>
</tr>
<tr>
<td>HostMetadataItem</td>
<td></td>
<td></td>
<td></td>
<td>Optional parameter that defines a Zabbix agent item used for getting host metadata. This option is only used when HostMetadata is not defined. Supports UserParameters, performance counters and aliases. Supports system.run[] regardless of EnableRemoteCommands value. HostMetadataItem value is retrieved on each autoregistration attempt and is used only at host autoregistration process (active agent). During an autoregistration request an agent will log a warning message if the value returned by the specified item is over the limit of 255 characters. The value returned by the item must be a UTF-8 string otherwise it will be ignored.</td>
</tr>
<tr>
<td>Hostname</td>
<td>no</td>
<td></td>
<td></td>
<td>List of comma-delimited unique, case-sensitive hostnames. Required for active checks and must match hostnames as configured on the server. Value is acquired from HostnameItem if undefined. Allowed characters: alphanumeric, '.', ',', '_' and '-'. Maximum length: 128 characters per hostname, 2048 characters for the entire line.</td>
</tr>
<tr>
<td>HostnameItem</td>
<td></td>
<td></td>
<td></td>
<td>Optional parameter that defines a Zabbix agent item used for getting host name. This option is only used when Hostname is not defined. Does not support UserParameters, performance counters or aliases, but does support system.run[] regardless of EnableRemoteCommands value. The output length is limited to 512KB. See also a more detailed description.</td>
</tr>
<tr>
<td>Include</td>
<td>no</td>
<td></td>
<td></td>
<td>You may include individual files or all files in a directory in the configuration file. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. For example: /absolute/path/to/config/files/*.conf. See special notes about limitations.</td>
</tr>
</tbody>
</table>

Include mandatory: no

Range: 0-3600

Default: 60

Supported since Zabbix 4.4.0.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListenBacklog</td>
<td>no</td>
<td>0 - INT_MAX</td>
<td>SOMAXCONN</td>
<td>The maximum number of pending connections in the TCP queue. Default value is a hard-coded constant, which depends on the system. Maximum supported value depends on the system, too high values may be silently truncated to the 'implementation-specified maximum'.</td>
</tr>
<tr>
<td>ListenIP</td>
<td>no</td>
<td>0.0.0.0</td>
<td></td>
<td>List of comma-delimited IP addresses that the agent should listen on.</td>
</tr>
<tr>
<td>ListenPort</td>
<td>no</td>
<td>1024-32767</td>
<td>10050</td>
<td>Agent will listen on this port for connections from the server.</td>
</tr>
<tr>
<td>LogFile</td>
<td>yes, if LogType is set to file, otherwise no</td>
<td>C:\zabbix_agentd.log</td>
<td></td>
<td>Name of the agent log file.</td>
</tr>
<tr>
<td>LogFileSize</td>
<td>no</td>
<td>0-1024</td>
<td>1</td>
<td>Maximum size of log file in MB. 0 - disable automatic log rotation. Note: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.</td>
</tr>
<tr>
<td>LogType</td>
<td>no</td>
<td>file</td>
<td></td>
<td>Log output type: file - write log to file specified by LogFile parameter, system - write log Windows Event Log, console - write log to standard output. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>LogRemoteCommands</td>
<td>no</td>
<td>0</td>
<td></td>
<td>Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled</td>
</tr>
<tr>
<td>MaxLinesPerSecond</td>
<td>1-1000</td>
<td>20</td>
<td></td>
<td>Maximum number of new lines the agent will send per second to Zabbix server or proxy processing 'log', 'logrt' and 'eventlog' active checks. The provided value will be overridden by the parameter 'maxlines', provided in 'log', 'logrt' or 'eventlog' item keys. Note: Zabbix will process 10 times more new lines than set in MaxLinesPerSecond to seek the required string in log items.</td>
</tr>
<tr>
<td>PerfCounter</td>
<td>no</td>
<td></td>
<td></td>
<td>Defines a new parameter &lt;parameter_name&gt; which is an average value for system performance counter &lt;perf_counter_path&gt; for the specified time period &lt;period&gt; (in seconds). Syntax: &lt;parameter_name&gt;,”&lt;perf_counter_path&gt;”,&lt;period&gt; For example, if you wish to receive average number of processor interrupts per second for last minute, you can define a new parameter &quot;interrupts&quot; as the following: PerfCounter = interrupts,&quot;\Processor(0)\Interrupts/sec&quot;,60 Please note double quotes around performance counter path. The parameter name (interrupts) is to be used as the item key when creating an item. Samples for calculating average value will be taken every second. You may run &quot;typeperf -qx&quot; to get list of all performance counters available in Windows.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>-----------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PerfCounterEn</td>
<td>no</td>
<td></td>
<td></td>
<td>Defines a new parameter &lt;parameter_name&gt; which is an average value for system performance counter &lt;perf_counter_path&gt; for the specified time period &lt;period&gt; (in seconds). Syntax: &lt;parameter_name&gt;,”&lt;perf_counter_path&gt;”,”&lt;period&gt; Compared to PerfCounter, perfcounter paths must be in English. Supported only on Windows Server 2008/Vista and above. For example, if you wish to receive average number of processor interrupts per second for last minute, you can define a new parameter “interrupts” as the following: PerfCounterEn = interrupts,”\Processor(0)\Interrupts/sec”,60 Please note double quotes around performance counter path. The parameter name (interrupts) is to be used as the item key when creating an item. Samples for calculating average value will be taken every second. You can find the list of English strings by viewing the following registry key: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\009. This parameter is supported since Zabbix 4.0.13 and 4.2.7.</td>
</tr>
<tr>
<td>RefreshActiveChecks</td>
<td>60-3600</td>
<td>120</td>
<td>60</td>
<td>How often list of active checks is refreshed, in seconds. Note that after failing to refresh active checks the next refresh will be attempted after 60 seconds.</td>
</tr>
<tr>
<td>Server</td>
<td>yes, if StartAgents is not explicitly set to 0</td>
<td></td>
<td></td>
<td>List of comma delimited IP addresses, optionally in CIDR notation, or hostnames of Zabbix servers. Incoming connections will be accepted only from the hosts listed here.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If IPv6 support is enabled then ‘127.0.0.1’, ‘::127.0.0.1’, ‘::ffff:127.0.0.1’ are treated equally and ‘::/0’ will allow any IPv4 or IPv6 address. ‘0.0.0.0/0’ can be used to allow any IPv4 address. Note, that “IPv4-compatible IPv6 addresses” (0000::/96 prefix) are supported but deprecated by RFC4291. Example: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.domain Spaces are allowed.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>--------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ServerActive</td>
<td>no</td>
<td>(*)</td>
<td></td>
<td>Zabbix server/proxy address or cluster configuration to get active checks from. Server/proxy address is IP address or DNS name and optional port separated by colon. Cluster configuration is one or more server addresses separated by semicolon. Multiple Zabbix servers/clusters and Zabbix proxies can be specified, separated by comma. More than one Zabbix proxy should not be specified from each Zabbix server/cluster. If Zabbix proxy is specified then Zabbix server/cluster for that proxy should not be specified. Multiple comma-delimited addresses can be provided to use several independent Zabbix servers in parallel. Spaces are allowed. If port is not specified, default port is used. IPv6 addresses must be enclosed in square brackets if port for that host is specified. If port is not specified, square brackets for IPv6 addresses are optional. If this parameter is not specified, active checks are disabled. Example for Zabbix proxy: ServerActive=127.0.0.1:10051 Example for multiple servers: ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1] Example for high availability: ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster.node3 Example for high availability with two clusters and one server: ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.cluster2.node1;zabbix.cluster2.node2,zabbix.domain</td>
</tr>
<tr>
<td>SourceIP</td>
<td>no</td>
<td></td>
<td></td>
<td>Source IP address for: - outgoing connections to Zabbix server or Zabbix proxy; - making connections while executing some items (web.page.get, net.tcp.port, etc.)</td>
</tr>
<tr>
<td>StartAgents</td>
<td>no</td>
<td>0-63 (*)</td>
<td>3</td>
<td>Number of pre-forked instances of zabbix_agentd that process passive checks. If set to 0, disables passive checks and the agent will not listen on any TCP port.</td>
</tr>
<tr>
<td>Timeout</td>
<td>no</td>
<td>1-30</td>
<td>3</td>
<td>Spend no more than Timeout seconds on processing What incoming connections to accept. Used for a passive checks. Multiple values can be specified, separated by comma: unencrypted - accept connections without encryption (default) psk - accept connections with TLS and a pre-shared key (PSK) cert - accept connections with TLS and a certificate This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSAccept</td>
<td>yes, if TLS certificate or PSK parameters are defined (even for unencrypted connection), otherwise no</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSCAFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSCertFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>--------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TLSConnect</td>
<td>yes, if TLS certificate or PSK parameters are defined (even for unencrypted connection), otherwise no</td>
<td></td>
<td></td>
<td>How the agent should connect to Zabbix server or proxy. Used for active checks. Only one value can be specified: unencrypted - connect without encryption (default) psk - connect using TLS and a pre-shared key (PSK) cert - connect using TLS and a certificate This parameter is supported since Zabbix 3.0.0.</td>
</tr>
<tr>
<td>TLSRCLFile</td>
<td>no</td>
<td>Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSKeyFile</td>
<td>no</td>
<td>Full pathname of a file containing the agent private key used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSPSKFile</td>
<td>no</td>
<td>Full pathname of a file containing the agent pre-shared key used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSPSKIdentity</td>
<td>no</td>
<td>Pre-shared key identity string, used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSServerCertIssuer</td>
<td></td>
<td>Allowed server (proxy) certificate issuer. This parameter is supported since Zabbix 3.0.0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSServerCertSubject</td>
<td></td>
<td>Allowed server (proxy) certificate subject. This parameter is supported since Zabbix 3.0.0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UnsafeUserParameters</td>
<td>0-1</td>
<td>0</td>
<td>Allow all characters to be passed in arguments to user-defined parameters. 0 - do not allow 1 - allow The following characters are not allowed: ' &quot; * [ ] { } ~ $ ! &amp; ; ( ) &gt;</td>
<td># @ Additionally, newline characters are not allowed.</td>
</tr>
<tr>
<td>UserParameterDir</td>
<td>no</td>
<td>Default search path for UserParameter commands. If used, the agent will change its working directory to the one specified here before executing a command. Thereby, UserParameter commands can have a relative . / prefix instead of a full path. Only one entry is allowed. Example: UserParameterDir=/opt/myscripts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UserParameter</td>
<td>no</td>
<td>User-defined parameter to monitor. There can be several user-defined parameters. Format: UserParameter=&lt;key&gt;,&lt;shell command&gt; Note that shell command must not return empty string or EOL only. Shell commands may have relative paths, if UserParameterDir parameter is specified. Examples: UserParameter=sys.test.who</td>
<td>wc -l UserParameter=check_cpu,./custom_script.sh</td>
<td></td>
</tr>
</tbody>
</table>

(*) The number of active servers listed in ServerActive plus the number of pre-forked instances for passive checks specified in StartAgents must be less than 64.

See also

1. Differences in the Zabbix agent configuration for active and passive checks starting from version 2.0.0.

6 Zabbix agent 2 (Windows)

Overview
Zabbix agent 2 is a new generation of Zabbix agent and may be used in place of Zabbix agent.

This section lists parameters supported in a Zabbix agent 2 configuration file (zabbix_agent2.win.conf).

Note that:

• The default values reflect process defaults, not the values in the shipped configuration files;
• Zabbix supports configuration files only in UTF-8 encoding without BOM;
• Comments starting with "#" are only supported in the beginning of the line. <br>

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alias</td>
<td>no</td>
<td></td>
<td></td>
<td>Sets an alias for an item key. It can be used to substitute long and complex item key with a smaller and simpler one. Multiple Alias parameters may be present. Multiple parameters with the same Alias key are allowed. Different Alias keys may reference the same item key. Aliases can be used in HostMetadataItem but not in HostnameItem parameters.</td>
</tr>
<tr>
<td>AllowKey</td>
<td>no</td>
<td></td>
<td></td>
<td>Allow execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports &quot;*&quot; character to match any number of any characters. Multiple key matching rules may be defined in combination with DenyKey. The parameters are processed one by one according to their appearance order. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.</td>
</tr>
<tr>
<td>BufferSend</td>
<td>no</td>
<td>1-3600</td>
<td>5</td>
<td>The time interval in seconds which determines how often values are sent from the buffer to Zabbix server. Note, that if the buffer is full, the data will be sent sooner.</td>
</tr>
<tr>
<td>BufferSize</td>
<td>no</td>
<td>2-65535</td>
<td>100</td>
<td>Maximum number of values in a memory buffer. The agent will send all collected data to Zabbix server or proxy if the buffer is full. This parameter should only be used if persistent buffer is disabled (EnablePersistentBuffer=0).</td>
</tr>
<tr>
<td>ControlSocket</td>
<td>no</td>
<td></td>
<td></td>
<td>The control socket, used to send runtime commands with <code>-R</code> option.</td>
</tr>
</tbody>
</table>

Examples:

1. Retrieving the ID of user ‘zabbix’.
   
   Alias=zabbix.userid:
   
   | vfs.file.regexp[/etc/passwd,"^zabbix::([0-9]+)""]|1|
   
   Now shorthand key `zabbix.userid` may be used to retrieve data.

2. Getting CPU utilization with default and custom parameters.
   
   Alias=cpu.util:system.cpu.util
   
   Alias=cpu.util[*]:system.cpu.util[*]
   
   This allows use `cpu.util` key to get CPU utilization percentage with default parameters as well as use `cpu.util[all, idle, avg15]` to get specific data about CPU utilization.

3. Running multiple low-level discovery rules processing the same discovery items.
   
   Alias=vfs.fs.discovery[*]:vfs.fs.discovery
   
   Now it is possible to set up several discovery rules using `vfs.fs.discovery` with different parameters for each rule, e.g., `vfs.fs.discovery[foo]`, `vfs.fs.discovery[bar]`, etc.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DebugLevel</td>
<td>no</td>
<td>0-5</td>
<td>3</td>
<td>Specifies debug level: 0 - basic information about starting and stopping of Zabbix processes 1 - critical information 2 - error information 3 - warnings 4 - for debugging (produces lots of information) 5 - extended debugging (produces even more information)</td>
</tr>
<tr>
<td>DenyKey</td>
<td>no</td>
<td></td>
<td></td>
<td>Deny execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports &quot;*&quot; character to match any number of any characters. Multiple key matching rules may be defined in combination with AllowKey. The parameters are processed one by one according to their appearance order. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.</td>
</tr>
<tr>
<td>EnablePersistentBuffer</td>
<td>no</td>
<td>0-1</td>
<td>0</td>
<td>Enable usage of local persistent storage for active items. 0 - disabled 1 - enabled If persistent storage is disabled, the memory buffer will be used.</td>
</tr>
<tr>
<td>ForceActiveChecksOnStart</td>
<td></td>
<td>0-1</td>
<td>0</td>
<td>Perform active checks immediately after restart for the first received configuration. 0 - disabled 1 - enabled Also available as per plugin configuration parameter, for example: Plugins.Uptime.System.ForceActiveChecksOnStart=1</td>
</tr>
<tr>
<td>HeartbeatFrequency</td>
<td>no</td>
<td>0-3600</td>
<td>60</td>
<td>Frequency of heartbeat messages in seconds. Used for monitoring the availability of active checks. 0 - heartbeat messages disabled.</td>
</tr>
<tr>
<td>HostInterface</td>
<td>no</td>
<td>0-255</td>
<td></td>
<td>Optional parameter that defines host interface. Host interface is used at host autoregistration process. An agent will issue an error and not start if the value is over the limit of 255 characters. If not defined, value will be acquired from HostInterfaceItem. Supported since Zabbix 4.4.0.</td>
</tr>
<tr>
<td>HostInterfaceItem</td>
<td>no</td>
<td></td>
<td></td>
<td>Optional parameter that defines an item used for getting host interface. Host interface is used at host autoregistration process. During an autoregistration request an agent will log a warning message if the value returned by specified item is over limit of 255 characters. This option is only used when HostInterface is not defined. Supported since Zabbix 4.4.0.</td>
</tr>
<tr>
<td>HostMetadata</td>
<td>no</td>
<td>0-255</td>
<td></td>
<td>Optional parameter that defines host metadata. Host metadata is used at host autoregistration process. An agent will issue an error and not start if the specified value is over the limit or a non-UTF-8 string. If not defined, the value will be acquired from HostMetadataItem.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>------------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HostMetadataItem</td>
<td>no</td>
<td>Optional</td>
<td>parameter that defines an item used for getting host metadata. Host metadata item value is retrieved on each autoregistration attempt for host autoregistration process. During an autoregistration request an agent will log a warning message if the value returned by the specified item is over the limit of 255 characters. This option is only used when HostMetadata is not defined. Supports UserParameters and aliases. Supports system.run[] regardless of EnableRemoteCommands value. The value returned by the item must be a UTF-8 string otherwise it will be ignored.</td>
<td></td>
</tr>
<tr>
<td>Hostname</td>
<td>no</td>
<td>List of comma-delimited unique, case-sensitive hostnames. Required for active checks and must match hostnames as configured on the server. Value is acquired from Hostnameltem if undefined. Allowed characters: alphanumeric, ‘.’, ‘,’ ‘,’ and ‘—’. Maximum length: 128 characters per hostname, 2048 characters for the entire line.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hostnameltem</td>
<td>no</td>
<td>system.hostname Item used for generating Hostname if it is not defined. Ignored if Hostname is defined. Does not support UserParameters or aliases, but does support system.run[] regardless of EnableRemoteCommands value. The output length is limited to 512KB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Include</td>
<td>no</td>
<td>You may include individual files or all files in a directory in the configuration file. During the installation Zabbix will create the include directory in /usr/local/etc, unless modified during the compile time. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. For example: /absolute/path/to/config/files/*.conf Since Zabbix 6.0.0 a path can be relative to zabbix_agent2.win.conf file location. See special notes about limitations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ListenIP</td>
<td>no</td>
<td>List of comma-delimited IP addresses that the agent should listen on. The first IP address is sent to Zabbix server, if connecting to it, to retrieve the list of active checks.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ListenPort</td>
<td>no</td>
<td>1024-32767</td>
<td>10050</td>
<td>Agent will listen on this port for connections from the server.</td>
</tr>
<tr>
<td>LogFile</td>
<td>yes, if LogType is set to file, otherwise no</td>
<td>c:\zabbix_agent2.log</td>
<td>c:\zabbix_agent2.log if LogType is ‘file’.</td>
<td></td>
</tr>
<tr>
<td>LogFileSize</td>
<td>no</td>
<td>0-1024</td>
<td>1</td>
<td>Maximum size of log file in MB. 0 - disable automatic log rotation. Note: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.</td>
</tr>
<tr>
<td>LogType</td>
<td>no</td>
<td>file</td>
<td>file</td>
<td>Specifies where log messages are written to: file - file specified by LogFile parameter, console - standard output.</td>
</tr>
<tr>
<td>PersistentBufferFile</td>
<td>no</td>
<td>The file, where Zabbix Agent2 should keep SQLite database. Must be a full filename. This parameter is only used if persistent buffer is enabled (EnablePersistentBuffer=1).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------</td>
<td>---------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PersistentBufferPeriod</td>
<td>no</td>
<td>1m-365d</td>
<td>1h</td>
<td>The time period for which data should be stored, when there is no connection to the server or proxy. Older data will be lost. Log data will be preserved. This parameter is only used if persistent buffer is enabled (EnablePersistentBuffer=1).</td>
</tr>
<tr>
<td>Plugins</td>
<td>no</td>
<td></td>
<td></td>
<td>Since Zabbix 6.0.0 most of the plugins have their own configuration files. The agent configuration file contains plugin parameters listed below.</td>
</tr>
<tr>
<td>Plugins.Log.MaxLinesPerSecond</td>
<td>no</td>
<td>1-1000</td>
<td>20</td>
<td>Maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log' and 'eventlog' active checks. The provided value will be overridden by the parameter 'maxlines', provided in 'log' or 'eventlog' item key. Note: Zabbix will process 10 times more new lines than set in MaxLinesPerSecond to seek the required string in log items. This parameter is supported since 4.4.2 and replaces MaxLinesPerSecond.</td>
</tr>
<tr>
<td>Plugins.SystemRun.LogRemoteCommands</td>
<td>0</td>
<td></td>
<td></td>
<td>Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled Commands will be logged only if executed remotely. Log entries will not be created if system.run[] is launched locally by HostMetadataItem, HostInterfaceItem or HostnameItem parameters. This parameter is supported since 4.4.2 and replaces LogRemoteCommands.</td>
</tr>
<tr>
<td>PluginSocket</td>
<td>no</td>
<td></td>
<td></td>
<td>Path to unix socket for loadable plugin communications.</td>
</tr>
<tr>
<td>PluginTimeout</td>
<td>no</td>
<td>1-30</td>
<td></td>
<td>Global timeout for connections with loadable plugins.</td>
</tr>
<tr>
<td>RefreshActiveChecks</td>
<td>no</td>
<td>60-3600</td>
<td>120</td>
<td>How often the list of active checks is refreshed, in seconds. Note that after failing to refresh active checks the next refresh will be attempted after 60 seconds.</td>
</tr>
<tr>
<td>Server</td>
<td>yes</td>
<td></td>
<td></td>
<td>List of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and Zabbix proxies. Incoming connections will be accepted only from the hosts listed here. If IPv6 support is enabled then '127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address. Example: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com Spaces are allowed.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
<td>---------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ServerActive</td>
<td>no</td>
<td></td>
<td></td>
<td>Zabbix server/proxy address or cluster configuration to get active checks from. Server/proxy address is IP address or DNS name and optional port separated by colon. Cluster configuration is one or more server addresses separated by semicolon. Multiple Zabbix servers/clusters and Zabbix proxies can be specified, separated by comma. More than one Zabbix proxy should not be specified from each Zabbix server/cluster. If Zabbix proxy is specified then Zabbix server/cluster for that proxy should not be specified. Multiple addresses can be provided to use several independent Zabbix servers in parallel. Spaces are allowed. If port is not specified, default port is used. IPv6 addresses must be enclosed in square brackets if port for that host is specified. If port is not specified, square brackets for IPv6 addresses are optional. If this parameter is not specified, active checks are disabled. Example for Zabbix proxy: ServerActive=127.0.0.1:10051 Example: ServerActive=127.0.0.1:20051,zabbix.example.com,[::1]:30051,:1,[12fc::1]</td>
</tr>
<tr>
<td>SourceIP</td>
<td>no</td>
<td></td>
<td></td>
<td>Source IP address for: - outgoing connections to Zabbix server or Zabbix proxy; - making connections while executing some items (web.page.get, net.tcp.port, etc.)</td>
</tr>
<tr>
<td>StatusPort</td>
<td>no</td>
<td>1024-32767</td>
<td>3</td>
<td>If set, agent will listen on this port for HTTP status requests (<a href="http://localhost">http://localhost</a>:&lt;port&gt;/status).</td>
</tr>
<tr>
<td>Timeout</td>
<td>no</td>
<td>1-30</td>
<td>3</td>
<td>Spend no more than Timeout seconds on processing. What incoming connections to accept. Used for a passive checks. Multiple values can be specified, separated by comma: unencrypted - accept connections without encryption (default) psk - accept connections with TLS and a pre-shared key (PSK) cert - accept connections with TLS and a certificate</td>
</tr>
<tr>
<td>TLSAccept</td>
<td>yes, if TLS certificate or PSK parameters are defined (even for unencrypted connection), otherwise no</td>
<td></td>
<td></td>
<td>How the agentshould connect to Zabbix server or proxy. Used for active checks. Only one value can be specified: unencrypted - connect without encryption (default) psk - connect using TLS and a pre-shared key (PSK) cert - connect using TLS and a certificate</td>
</tr>
<tr>
<td>TLSCAFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.</td>
</tr>
<tr>
<td>TLSCertFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components.</td>
</tr>
<tr>
<td>TLSConnect</td>
<td>yes, if TLS certificate or PSK parameters are defined (even for unencrypted connection), otherwise no</td>
<td></td>
<td></td>
<td>How the agentshould connect to Zabbix server or proxy. Used for active checks. Only one value can be specified: unencrypted - connect without encryption (default) psk - connect using TLS and a pre-shared key (PSK) cert - connect using TLS and a certificate</td>
</tr>
</tbody>
</table>
### 7 Zabbix agent 2 plugins

**Overview**

This section contains descriptions of configuration file parameters for Zabbix agent 2 plugins. Please use the sidebar to access information about the specific plugin.

#### 1 Ceph plugin

**Overview**

This section lists parameters supported in the Ceph Zabbix agent 2 plugin configuration file (ceph.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLSCRLFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications with Zabbix components.</td>
</tr>
<tr>
<td>TLSKeyFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the agent private key used for encrypted communications with Zabbix components.</td>
</tr>
<tr>
<td>TLSPSKFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Full pathname of a file containing the agent pre-shared key used for encrypted communications with Zabbix components.</td>
</tr>
<tr>
<td>TLSPSKIdentity</td>
<td>no</td>
<td></td>
<td></td>
<td>Pre-shared key identity string, used for encrypted communications with Zabbix server.</td>
</tr>
<tr>
<td>TLSServerCertIssuer</td>
<td>no</td>
<td></td>
<td></td>
<td>Allowed server (proxy) certificate issuer.</td>
</tr>
<tr>
<td>TLSServerCertSubject</td>
<td>no</td>
<td></td>
<td></td>
<td>Allowed server (proxy) certificate subject.</td>
</tr>
<tr>
<td>UnsafeUserParameters</td>
<td>0,1</td>
<td></td>
<td>0</td>
<td>Allow all characters to be passed in arguments to user-defined parameters.</td>
</tr>
<tr>
<td>UserParameterDir</td>
<td>no</td>
<td></td>
<td></td>
<td>Default search path for UserParameter commands. If used, the agent will change its working directory to the one specified here before executing a command. Thereby, UserParameter commands can have a relative ./ prefix instead of a full path. Only one entry is allowed.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------</td>
<td>-----------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Plugins.Ceph.InsecureSkipVerify</td>
<td></td>
<td>false / true</td>
<td>false</td>
<td>Determines whether an http client should verify the server’s certificate chain and host name. If true, TLS accepts any certificate presented by the server and any host name in that certificate. In this mode, TLS is susceptible to man-in-the-middle attacks (should be used only for testing).</td>
</tr>
<tr>
<td>Plugins.Ceph.KeepAlive</td>
<td></td>
<td>60-900</td>
<td>300</td>
<td>The maximum time of waiting (in seconds) before unused plugin connections are closed.</td>
</tr>
<tr>
<td>Plugins.Ceph.Sessions.&lt;SessionName&gt;.ApiKey</td>
<td></td>
<td></td>
<td></td>
<td>Named session API key. &lt;SessionName&gt; - name of a session for using in item keys.</td>
</tr>
<tr>
<td>Plugins.Ceph.Sessions.&lt;SessionName&gt;.Uri</td>
<td></td>
<td><a href="https://localhost:8003">https://localhost:8003</a></td>
<td></td>
<td>Connection string of a named session. &lt;SessionName&gt; - name of a session for using in item keys. Should not include embedded credentials (they will be ignored). Must match the URI format. Only https scheme is supported; a scheme can be omitted (since version 5.2.3). A port can be omitted (default=8003). Examples: <a href="https://127.0.0.1:8003">https://127.0.0.1:8003</a> localhost</td>
</tr>
<tr>
<td>Plugins.Ceph.Timeout</td>
<td></td>
<td>1-30</td>
<td></td>
<td>Request execution timeout (how long to wait for a request to complete before shutting it down).</td>
</tr>
</tbody>
</table>

See also:
- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

2 Docker plugin

Overview

This section lists parameters supported in the Docker Zabbix agent 2 plugin configuration file (docker.conf).

Note that:
- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins.Docker.Endpoint</td>
<td></td>
<td></td>
<td></td>
<td>Docker daemon unix-socket location. Must contain a scheme (only unix:/// is supported).</td>
</tr>
<tr>
<td>Plugins.Docker.Timeout</td>
<td></td>
<td>1-30</td>
<td></td>
<td>Request execution timeout (how long to wait for a request to complete before shutting it down).</td>
</tr>
</tbody>
</table>

See also:
- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

3 Memcached plugin

Overview

This section lists parameters supported in the Memcached Zabbix agent 2 plugin configuration file (memcached.conf).

Note that:
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins.Memcached.KeepAlive</td>
<td></td>
<td>60-900</td>
<td>300</td>
<td>The maximum time of waiting (in seconds) before unused plugin connections are closed.</td>
</tr>
<tr>
<td>Plugins.Memcached.Timeout</td>
<td></td>
<td>1-30</td>
<td>global</td>
<td>Request execution timeout (how long to wait for a request to complete before shutting it down).</td>
</tr>
</tbody>
</table>

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

4 Modbus plugin

Overview

This section lists parameters supported in the Modbus Zabbix agent 2 plugin configuration file (modbus.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins.Modbus.Sessions.&lt;SessionName&gt;.Endpoint</td>
<td></td>
<td></td>
<td></td>
<td>Endpoint is a connection string consisting of a protocol scheme, a host address and a port or serial port name and attributes.</td>
</tr>
</tbody>
</table>

See also:
• Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
• Instructions for configuring plugins

5 MongoDB plugin

Overview
This section lists parameters supported in the MongoDB Zabbix agent 2 plugin configuration file (mongodb.conf).

This is a loadable plugin, which is available in the official Zabbix agent 2 plugin repository.

Note that:
• The default values reflect process defaults, not the values in the shipped configuration files;
• Zabbix supports configuration files only in UTF-8 encoding without BOM;
• Comments starting with "#" are only supported at the beginning of the line.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins.MongoDB.KeepAlive</td>
<td>no</td>
<td>60-900</td>
<td>300</td>
<td>The maximum time of waiting (in seconds) before unused plugin connections are closed.</td>
</tr>
<tr>
<td>Plugins.MongoDB.Sessions.&lt;SessionName&gt;.Password</td>
<td>no</td>
<td></td>
<td></td>
<td>Name of a session for using in item keys.</td>
</tr>
<tr>
<td>Plugins.MongoDB.Sessions.&lt;SessionName&gt;.Uri</td>
<td>no</td>
<td></td>
<td></td>
<td>Connection string of a session. Should not include embedded credentials (they will be ignored). Must match the URI format. Only tcp scheme is supported; a scheme can be omitted. A port can be omitted (default=27017). Examples: tcp://127.0.0.1:27017, tcp:localhost, localhost</td>
</tr>
<tr>
<td>Plugins.MongoDB.System.Path</td>
<td>no</td>
<td></td>
<td></td>
<td>Path to plugin executable.</td>
</tr>
<tr>
<td>Plugins.MongoDB.Timeout</td>
<td>no</td>
<td>1-30</td>
<td>global</td>
<td>Request execution timeout (how long to wait for a request to complete before shutting it down).</td>
</tr>
</tbody>
</table>

See also:
• Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
• Instructions for configuring plugins

6 MQTT plugin

Overview
This section lists parameters supported in the MQTT Zabbix agent 2 plugin configuration file (mqtt.conf).

Note that:
• The default values reflect process defaults, not the values in the shipped configuration files;
• Zabbix supports configuration files only in UTF-8 encoding without BOM;
• Comments starting with "#" are only supported at the beginning of the line.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins.MQTT.Timeout</td>
<td>no</td>
<td>1-30</td>
<td>global</td>
<td>Request execution timeout (how long to wait for a request to complete before shutting it down).</td>
</tr>
</tbody>
</table>

See also:
• Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
• Instructions for configuring plugins
7 MySQL plugin

Overview

This section lists parameters supported in the MySQL Zabbix agent 2 plugin configuration file (mysql.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins.Mysql.CallTimeout</td>
<td></td>
<td>1-30</td>
<td>global</td>
<td>The maximum amount of time in seconds to wait for a request to be done.</td>
</tr>
<tr>
<td>Plugins.Mysql.KeepAlive</td>
<td></td>
<td>60-900</td>
<td>300</td>
<td>The maximum time of waiting (in seconds) before unused plugin connections are closed.</td>
</tr>
<tr>
<td>Plugins.Mysql.Sessions.&lt;SessionName&gt;.TLCACertFile</td>
<td></td>
<td></td>
<td></td>
<td>Full pathname of a file containing the top-level CA(s)</td>
</tr>
<tr>
<td>Plugins.Mysql.Sessions.&lt;SessionName&gt;.TLSKeyCertFile</td>
<td></td>
<td></td>
<td></td>
<td>Full path name of a file containing the agent certificate or certificate chain, used for encrypted communications between Zabbix agent 2 and monitored databases.</td>
</tr>
<tr>
<td>Plugins.Mysql.Sessions.&lt;SessionName&gt;.TLSConnect</td>
<td></td>
<td></td>
<td></td>
<td>Encryption type for communications between Zabbix agent 2 and monitored databases.</td>
</tr>
<tr>
<td>Plugins.Mysql.Sessions.&lt;SessionName&gt;.TLSKeyFile</td>
<td></td>
<td></td>
<td></td>
<td>Full path name of a file containing the database private key used for encrypted communications between Zabbix agent 2 and monitored databases.</td>
</tr>
<tr>
<td>Plugins.Mysql.Timeout</td>
<td></td>
<td>1-30</td>
<td>global</td>
<td>Request execution timeout (how long to wait for a request to complete before shutting it down).</td>
</tr>
</tbody>
</table>

See also:
8 Oracle plugin

Overview

This section lists parameters supported in the Oracle Zabbix agent 2 plugin configuration file (ceph.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins.Oracle.CallTimeout</td>
<td>1-30</td>
<td>global timeout</td>
<td></td>
<td>The maximum wait time in seconds for a request to be completed.</td>
</tr>
<tr>
<td>Plugins.Oracle.ConnectTimeout</td>
<td>1-30</td>
<td>global timeout</td>
<td></td>
<td>The maximum wait time in seconds for a connection to be established.</td>
</tr>
<tr>
<td>Plugins.Oracle.CustomQueriesPath</td>
<td></td>
<td>Full pathname of a directory containing .sql files with custom queries.</td>
<td></td>
<td>Disabled by default. Example: /etc/zabbix/oracle/sql</td>
</tr>
<tr>
<td>Plugins.Oracle.KeepAlive</td>
<td>60-900</td>
<td>300</td>
<td></td>
<td>The maximum time of waiting (in seconds) before unused plugin connections are closed.</td>
</tr>
<tr>
<td>Plugins.Oracle.Sessions.&lt;SessionName&gt;.Service</td>
<td></td>
<td></td>
<td></td>
<td>Named session service name to be used for connection (SID is not supported). Supported for: Oracle. &lt;PluginName&gt; - name of the plugin. &lt;SessionName&gt; - name of a session for using in item keys.</td>
</tr>
<tr>
<td>Plugins.Oracle.Sessions.&lt;SessionName&gt;.Uri</td>
<td>tcp://localhost:1521</td>
<td></td>
<td></td>
<td>Named session connection string for Oracle. &lt;SessionName&gt; - name of a session for using in item keys. Should not include embedded credentials (they will be ignored). Must match the URI format. Only tcp scheme is supported; a scheme can be omitted (since version 5.2.3). A port can be omitted (default=1521). Examples: tcp://127.0.0.1:1521 localhost</td>
</tr>
</tbody>
</table>

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins.Postgres.CallTimeout</td>
<td>no</td>
<td>1-30</td>
<td>global</td>
<td>The maximum wait time in seconds for a request to be completed.</td>
</tr>
<tr>
<td>Plugins.Postgres.Host</td>
<td>localhost</td>
<td></td>
<td></td>
<td>IP address or DNS name of the host used for PostgreSQL. Examples: localhost, 192.168.1.1</td>
</tr>
<tr>
<td>Plugins.Postgres.KeepAlive</td>
<td>60-900</td>
<td>300</td>
<td></td>
<td>The maximum time of waiting (in seconds) before unused plugin connections are closed.</td>
</tr>
<tr>
<td>Plugins.Postgres.Port</td>
<td>5432</td>
<td></td>
<td></td>
<td>A port to be used for PostgreSQL.</td>
</tr>
<tr>
<td>Plugins.Postgres.Sessions.&lt;SessionName&gt;.DBname</td>
<td>postgres</td>
<td></td>
<td></td>
<td>Database name of a named session. &lt;SessionName&gt; - name of a session for using in item keys.</td>
</tr>
<tr>
<td>Plugins.Postgres.Sessions.&lt;SessionName&gt;.TLSCAFile</td>
<td>no</td>
<td>Fullpathnameofafilecontainingthetop-levelCA(s)certificatesforencryptedcommunicationsbetweenZabbixagent2andmonitoreddatabases.</td>
<td>&lt;SessionName&gt; - name of a session for using in item keys.</td>
<td></td>
</tr>
<tr>
<td>Plugins.Postgres.Sessions.&lt;SessionName&gt;.TLSCertFile</td>
<td>no</td>
<td>Fullpathnameofafilecontainingtheagentcertificateorcertificatechain,usedforencryptedcommunicationsbetweenZabbixagent2andmonitoreddatabases.</td>
<td>&lt;SessionName&gt; - name of a session for using in item keys.</td>
<td></td>
</tr>
<tr>
<td>Plugins.Postgres.Sessions.&lt;SessionName&gt;.TLSConnect</td>
<td>no</td>
<td>EncryptiontypeforcommunicationsbetweenZabbixagent2andmonitoreddatabases.</td>
<td>&lt;SessionName&gt; - name of a session for using in item keys. Accepted values: required - require TLS connection; verify_ca - verify certificates; verify_full - verify certificates and IP address.</td>
<td></td>
</tr>
<tr>
<td>Plugins.Postgres.Sessions.&lt;SessionName&gt;.TLSKeyFile</td>
<td>no</td>
<td>FullpathnameofafilecontainingthedatabaseprivatekeyusedforencryptedcommunicationsbetweenZabbixagent2andmonitoreddatabases.</td>
<td>&lt;SessionName&gt; - name of a session for using in item keys.</td>
<td></td>
</tr>
<tr>
<td>Plugins.Postgres.Sessions.&lt;SessionName&gt;.Uri</td>
<td>postgres</td>
<td></td>
<td></td>
<td>Named session connection string for Oracle. &lt;SessionName&gt; - name of a session for using in item keys. Should not include embedded credentials (they will be ignored). Must match the URI format. Supported schemes are tcp and unix. Examples: tcp://127.0.0.1:5432 localhost</td>
</tr>
<tr>
<td>Plugins.Postgres.Timeout</td>
<td>1-30</td>
<td>global</td>
<td>timeout</td>
<td>Request execution timeout (how long to wait for a request to complete before shutting it down).</td>
</tr>
</tbody>
</table>

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

10 Redis plugin
Overview

This section lists parameters supported in the Redis Zabbix agent 2 plugin configuration file (redis.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins.Redis.KeepAlive</td>
<td>yes</td>
<td>60-900</td>
<td>300</td>
<td>The maximum time of waiting (in seconds) before unused plugin connections are closed.</td>
</tr>
<tr>
<td>Plugins.Redis.Timeout</td>
<td>no</td>
<td>1-30</td>
<td>global</td>
<td>Request execution timeout (how long to wait for a request to complete before shutting it down).</td>
</tr>
</tbody>
</table>

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

11 Smart plugin

Overview

This section lists parameters supported in the Smart Zabbix agent 2 plugin configuration file (smart.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plugins.Smart.Path</td>
<td>no</td>
<td></td>
<td>smartctl</td>
<td>Path to the smartctl executable.</td>
</tr>
<tr>
<td>Plugins.Smart.Timeout</td>
<td>no</td>
<td>1-30</td>
<td>global timeout</td>
<td>Request execution timeout (how long to wait for a request to complete before shutting it down).</td>
</tr>
</tbody>
</table>

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins
8 Zabbix Java gateway

If you use startup.sh and shutdown.sh scripts for starting Zabbix Java gateway, then you can specify the necessary configuration parameters in the settings.sh file. The startup and shutdown scripts source the settings file and take care of converting shell variables (listed in the first column) to Java properties (listed in the second column).

If you start Zabbix Java gateway manually by running java directly, then you specify the corresponding Java properties on the command line.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Property</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISTEN_IP</td>
<td>zabbix.listenIP</td>
<td>no</td>
<td>0.0.0.0</td>
<td></td>
<td>IP address to listen on.</td>
</tr>
<tr>
<td>LISTEN_PORT</td>
<td>zabbix.listenPort</td>
<td>no</td>
<td>1024-32767</td>
<td>10052</td>
<td>Port to listen on.</td>
</tr>
<tr>
<td>PID_FILE</td>
<td>zabbix.pidFile</td>
<td>no</td>
<td>/tmp/zabbix_java.pid</td>
<td></td>
<td>Name of PID file. If omitted, Zabbix Java Gateway is started as a console application.</td>
</tr>
<tr>
<td>PROPERTIES_FILE</td>
<td>zabbix.propertiesFile</td>
<td>no</td>
<td></td>
<td></td>
<td>Name of properties file. Can be used to set additional properties using a key-value format in such a way that they are not visible on a command line or to overwrite existing ones. For example: &quot;javax.net.ssl.trustStorePassword=&lt;password&gt;&quot;</td>
</tr>
<tr>
<td>START_POLLERS</td>
<td>zabbix.startPollers</td>
<td>no</td>
<td>1-1000</td>
<td>5</td>
<td>Number of worker threads to start.</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>zabbix.timeout</td>
<td>no</td>
<td>1-30</td>
<td>3</td>
<td>How long to wait for network operations.</td>
</tr>
</tbody>
</table>

Port 10052 is not IANA registered.

9 Zabbix web service

Overview

Zabbix web service is a process that is used for communication with external web services.

This section lists parameters supported in Zabbix web service configuration file (zabbix_web_service.conf).

Note that:
- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mandatory</th>
<th>Range</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowedIP</td>
<td>yes</td>
<td>List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and Zabbix proxies.Incoming connections will be accepted only from the hosts listed here. If IPv6 support is enabled then 127.0.0.1, ::127.0.0.1, ::ffff:127.0.0.1 are treated equally and ::/0 will allow any IPv4 or IPv6 address. 0.0.0.0/0 can be used to allow any IPv4 address.Example: 127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Mandatory</td>
<td>Range</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
<td>---------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DebugLevel</td>
<td>no</td>
<td>0-5</td>
<td>3</td>
<td>Specifies debug level: 0 - basic information about starting and stopping of Zabbix processes, 1 - critical information, 2 - error information, 3 - warnings, 4 - for debugging (produces lots of information), 5 - extended debugging (produces even more information).</td>
</tr>
<tr>
<td>ListenPort</td>
<td>no, yes</td>
<td>1024-32767</td>
<td>10053</td>
<td>The port service listens on for connections from the server. Log file name for LogType ‘file’ parameter. Example: /tmp/zabbix_web_service.log</td>
</tr>
<tr>
<td>LogFile</td>
<td>yes, if LogType is set to file, otherwise no</td>
<td>0-1024</td>
<td>1</td>
<td>Specifies where log messages are written to: system - syslog, file - file specified with LogFile parameter, console - standard output.</td>
</tr>
<tr>
<td>LogFileSize</td>
<td>no</td>
<td>0-1024</td>
<td>1</td>
<td>Maximum size of log file in MB. 0 - disable automatic log rotation.</td>
</tr>
<tr>
<td>LogType</td>
<td>no</td>
<td>system / file / console</td>
<td>file</td>
<td>Specifies where log messages are written to: system - syslog, file - file specified with LogFile parameter, console - standard output.</td>
</tr>
<tr>
<td>Timeout</td>
<td>no</td>
<td>1-30</td>
<td>3</td>
<td>Spend no more than Timeout seconds on processing.</td>
</tr>
<tr>
<td>TLSAccept</td>
<td>no</td>
<td>unencrypted / cert</td>
<td>unencrypted</td>
<td>Specifies what type of connection to use: unencrypted - accept connections without encryption (default), cert - accept connections with TLS and a certificate.</td>
</tr>
<tr>
<td>TLSCAFile</td>
<td>no</td>
<td>Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSCertFile</td>
<td>no</td>
<td>Full pathname of a file containing the service certificate or certificate chain, used for encrypted communications with Zabbix components.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLSKeyFile</td>
<td>no</td>
<td>Full pathname of a file containing the service private key used for encrypted communications with Zabbix components.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 10 Inclusion

**Overview**

Additional files or directories can be included into server/proxy/agent configuration using the `Include` parameter.

**Notes on inclusion**

- If the `Include` parameter is used for including a file, the file must be readable.
- If the `Include` parameter is used for including a directory:
  - All files in the directory must be readable.
  - No particular order of inclusion should be assumed (e.g. files are not included in alphabetical order).
  - All files in the directory are included into configuration.
  - Beware of file backup copies automatically created by some text editors. For example, if editing the `Include` directory.
- If the `Include` parameter is used for including files using a pattern:
  - All files matching the pattern must be readable.
  - No particular order of inclusion should be assumed (e.g. files are not included in alphabetical order).

### 4 Protocols

**1 Server-proxy data exchange protocol**

**Overview**

Server - proxy data exchange is based on JSON format.
Request and response messages must begin with header and data length.

Passive proxy

Proxy config request

The proxy config request is sent by server to provide proxy configuration data. This request is sent every ProxyConfigFrequency (server configuration parameter) seconds.

<table>
<thead>
<tr>
<th>name</th>
<th>value type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>server→proxy:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>request</td>
<td>string</td>
<td>'proxy config'</td>
</tr>
<tr>
<td>fields</td>
<td>object</td>
<td>one or more objects with &lt;table&gt; data</td>
</tr>
<tr>
<td>data</td>
<td>array</td>
<td>array of field names</td>
</tr>
<tr>
<td></td>
<td>string</td>
<td>field name</td>
</tr>
<tr>
<td></td>
<td>array</td>
<td>array of columns</td>
</tr>
<tr>
<td></td>
<td>string,number</td>
<td>column value with type depending on column type in database schema</td>
</tr>
<tr>
<td>proxy→server:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>response</td>
<td>string</td>
<td>the request success information ('success' or 'failed')</td>
</tr>
<tr>
<td>version</td>
<td>string</td>
<td>the proxy version (&lt;major&gt;.&lt;minor&gt;.&lt;build&gt;)</td>
</tr>
</tbody>
</table>

Example:

```json
server→proxy:
{
  "request": "proxy config",
  "globalmacro":{
    "fields":[
      "globalmacroid",
      "macro",
      "value"
    ],
    "data":[
      [2,
        "${SNMP_COMMUNITY}",
        "public"
      ]
    ],
    "hosts":{
      "fields":[
        "hostid",
        "host",
        "status",
        "ipmi_authtype",
        "ipmi_privilege",
        "ipmi_username",
        "ipmi_password",
        "name",
        "tls_connect",
        "tls_accept",
        "tlsIssuer",
        "tlsSubject",
        "tls_psk_identity",
        "tls_psk"
      ],
      "data":[
        [10001,
         "Linux",
         
```
3,
-1,
2,
"Linux",
1,
1,
"Zabbix Agent",
1,
1,
"Zabbix Agent",
1,
1,
"Zabbix Agent",
1,
1,
"Logger",
0,
-1,
2,
"Logger",
1,
1,
"Logger",
1,
1,
"interface":{
  "fields":[
    "interfaceid",
    "hostid",
    "main",
    "type",
    "useip",
    "ip",
    "dns",
    "port",
    "bulk"
  ],
  "data":[
    2,
  ]
}
The proxy data request is used to obtain host interface availability, historical, discovery and autoregistration data from proxy. This request is sent every ProxyDataFrequency (server configuration parameter) seconds.

### name | value type | description
--- | --- | ---
**server→proxy:**
request | string | 'proxy data'
**proxy→server:**
session | string | data session token
interface | array | (optional) array of interface availability data objects
available | number | Interface availability
  - **0.** INTERFACEAVAILABLEUNKNOWN - unknown
  - **1.** INTERFACEAVAILABLETRUE - available
  - **2.** INTERFACEAVAILABLEFALSE - unavailable
error | string | Interface error message or empty string
history | array | (optional) array of history data objects
  - itemid | number | item identifier
  - clock | number | item value timestamp (seconds)
  - ns | number | item value timestamp (nanoseconds)
  - value | string | (optional) item value
  - id | number | value identifier (ascending counter, unique within one data session)
  - timestamp | number | (optional) timestamp of log type items
  - source | string | (optional) eventlog item source value
  - severity | number | (optional) eventlog item severity value
  - eventid | number | (optional) eventlog item eventid value
  - state | string | (optional) item state
  - **0.** ITEMSTATENORMAL
  - **1.** ITEMSTATENOTSUPPORTED
lastlogsize | number | (optional) last log size of log type items
mtime | number | (optional) modify time of log type items
discovery | array | (optional) array of discovery data objects
clock | number | the discovery data timestamp
druleid | number | the discovery rule identifier
dcheckid | number | the discovery check identifier or null for discovery rule data
<table>
<thead>
<tr>
<th>name</th>
<th>value type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>type</strong></td>
<td>number</td>
<td>the discovery check type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1 discovery rule data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0. SVC_SSH - SSH service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. SVC_LDAP - LDAP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. SVC_SMTP - SMTP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. SVC_FTP - FTP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. SVC_HTTP - HTTP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. SVC_POP - POP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. SVC_NNTP - NNTP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7. SVC_IMAP - IMAP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8. SVC_TCP - TCP port availability check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9. SVC_AGENT - Zabbix agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10. SVC_SNMPv1 - SNMPv1 agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11. SVC_SNMPv2 - SNMPv2 agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12. SVC_ICMPPING - ICMP ping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13. SVC_SNMPv3 - SNMPv3 agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14. SVC_HTTPS - HTTPS service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15. SVC_TELNET - Telnet availability check</td>
</tr>
<tr>
<td><strong>ip</strong></td>
<td>string</td>
<td>the host IP address</td>
</tr>
<tr>
<td><strong>dns</strong></td>
<td>string</td>
<td>the host DNS name</td>
</tr>
<tr>
<td><strong>port</strong></td>
<td>number</td>
<td>(optional) service port number</td>
</tr>
<tr>
<td><strong>key_</strong></td>
<td>string</td>
<td>(optional) the item key for discovery check of type 9 SVC_AGENT</td>
</tr>
<tr>
<td><strong>value</strong></td>
<td>string</td>
<td>(optional) value received from the service, can be empty for most of services</td>
</tr>
<tr>
<td><strong>status</strong></td>
<td>number</td>
<td>(optional) service status:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0. DOBJECT_STATUS_UP - Service UP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. DOBJECT_STATUS_DOWN - Service DOWN</td>
</tr>
<tr>
<td><strong>auto</strong></td>
<td>array</td>
<td>(optional) array of autoregistration data objects</td>
</tr>
<tr>
<td><strong>regis-</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>tra-</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>tion</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>clock</strong></td>
<td>number</td>
<td>the autoregistration data timestamp</td>
</tr>
<tr>
<td><strong>host</strong></td>
<td>string</td>
<td>the host name</td>
</tr>
<tr>
<td><strong>ip</strong></td>
<td>string</td>
<td>(optional) the host IP address</td>
</tr>
<tr>
<td><strong>dns</strong></td>
<td>string</td>
<td>(optional) the resolved DNS name from IP address</td>
</tr>
<tr>
<td><strong>port</strong></td>
<td>string</td>
<td>(optional) the host port</td>
</tr>
<tr>
<td><strong>host_metadata</strong></td>
<td>string</td>
<td>(optional) the host metadata sent by agent (based on HostMetadata or HostMetadataItem agent configuration parameter)</td>
</tr>
<tr>
<td><strong>tasks</strong></td>
<td>array</td>
<td>(optional) array of tasks</td>
</tr>
<tr>
<td><strong>type</strong></td>
<td>number</td>
<td>the task type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0. ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT - remote command result</td>
</tr>
<tr>
<td><strong>status</strong></td>
<td>number</td>
<td>the remote command execution status:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0. ZBX_TM_REMOTE_COMMAND_COMPLETED - the remote command completed successfully</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. ZBX_TM_REMOTE_COMMAND_FAILED - the remote command failed</td>
</tr>
<tr>
<td><strong>error</strong></td>
<td>string</td>
<td>(optional) the error message</td>
</tr>
<tr>
<td><strong>parent_taskid</strong></td>
<td>number</td>
<td>the parent task id</td>
</tr>
<tr>
<td><strong>more</strong></td>
<td>number</td>
<td>(optional) 1 - there are more history data to send</td>
</tr>
<tr>
<td><strong>clock</strong></td>
<td>number</td>
<td>(optional) data transfer timestamp (seconds)</td>
</tr>
<tr>
<td><strong>ns</strong></td>
<td>number</td>
<td>(optional) data transfer timestamp (nanoseconds)</td>
</tr>
<tr>
<td><strong>version</strong></td>
<td>string</td>
<td>the proxy version (&lt;major&gt;.&lt;minor&gt;.&lt;build&gt;)</td>
</tr>
<tr>
<td>server-proxy:</td>
<td>string</td>
<td>the request success information ('success' or 'failed')</td>
</tr>
<tr>
<td><strong>response</strong></td>
<td>string</td>
<td>(optional) array of tasks</td>
</tr>
<tr>
<td><strong>tasks</strong></td>
<td>array</td>
<td>(optional) array of tasks</td>
</tr>
<tr>
<td>name</td>
<td>value type</td>
<td>description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>type</td>
<td>number</td>
<td>the task type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. ZBX_TM_TASK_PROCESS_REMOTE_COMMAND - remote command</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. ZBX_TM_TASK_PROCESS_REMOTE_COMMAND - remotecommand</td>
</tr>
<tr>
<td>clock</td>
<td>number</td>
<td>the task creation time</td>
</tr>
<tr>
<td>ttl</td>
<td>number</td>
<td>the time in seconds after which task expires</td>
</tr>
<tr>
<td>commandtype</td>
<td>number</td>
<td>the remote command type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0. ZBX_SCRIPT_TYPE_CUSTOM_SCRIPT - use custom script</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. ZBX_SCRIPT_TYPE_IPMI - use IPMI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. ZBX_SCRIPT_TYPE_SSH - use SSH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. ZBX_SCRIPT_TYPE_TERMINAL - use Telnet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. ZBX_SCRIPT_TYPE_GLOBAL_SCRIPT - use global script (currently functionally equivalent to custom script)</td>
</tr>
<tr>
<td>command</td>
<td>string</td>
<td>the remote command to execute</td>
</tr>
<tr>
<td>execute_on</td>
<td>number</td>
<td>the execution target for custom scripts:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0. ZBX_SCRIPT_EXECUTE_ON_AGENT - execute script on agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. ZBX_SCRIPT_EXECUTE_ON_SERVER - execute script on server</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. ZBX_SCRIPT_EXECUTE_ON_PROXY - execute script on proxy</td>
</tr>
<tr>
<td>port</td>
<td>number</td>
<td>(optional) the port for telnet and ssh commands</td>
</tr>
<tr>
<td>authtype</td>
<td>number</td>
<td>(optional) the authentication type for ssh commands</td>
</tr>
<tr>
<td>username</td>
<td>string</td>
<td>(optional) the user name for telnet and ssh commands</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>(optional) the password for telnet and ssh commands</td>
</tr>
<tr>
<td>publickey</td>
<td>string</td>
<td>(optional) the public key for ssh commands</td>
</tr>
<tr>
<td>privatekey</td>
<td>string</td>
<td>(optional) the private key for ssh commands</td>
</tr>
<tr>
<td>parent_taskid</td>
<td>number</td>
<td>the parent task id</td>
</tr>
<tr>
<td>hostid</td>
<td>number</td>
<td>target hostid</td>
</tr>
</tbody>
</table>

Example:

server→proxy:
```json
{
  "request": "proxy data"
}
```

proxy→server:
```json
{
  "session": "12345678901234567890123456789012",
  "interface availability": [
    {
      "interfaceid": 1,
      "available": 1,
      "error": ""
    },
    {
      "interfaceid": 2,
      "available": 2,
      "error": "Get value from agent failed: cannot connect to [[127.0.0.1]:10049]: [111] Connection refused"
    },
    {
      "interfaceid": 3,
      "available": 1,
      "error": ""
    },
    {
      "interfaceid": 4,
      "available": 1,
      "error": ""
    }
  ]
}
```
"history data": [ 
  { 
    "itemid": "12345",
    "clock": 1478609647,
    "ns": 32510044,
    "value": "52956612",
    "id": 1 
  }, 
  { 
    "itemid": "12346",
    "clock": 1478609647,
    "ns": 330690279,
    "state": 1,
    "value": "Cannot find information for this network interface in /proc/net/dev.",
    "id": 2 
  } 
],
"discovery data": [ 
  { 
    "clock": 1478608764,
    "drule": 2,
    "dcheck": 3,
    "type": 12,
    "ip": "10.3.0.10",
    "dns": "vdebian",
    "status": 1 
  }, 
  { 
    "clock": 1478608764,
    "drule": 2,
    "dcheck": null,
    "type": -1,
    "ip": "10.3.0.10",
    "dns": "vdebian",
    "status": 1 
  } 
],
"auto registration": [ 
  { 
    "clock": 1478608371,
    "host": "Logger1",
    "ip": "10.3.0.1",
    "dns": "localhost",
    "port": "10050" 
  }, 
  { 
    "clock": 1478608381,
    "host": "Logger2",
    "ip": "10.3.0.2",
    "dns": "localhost",
    "port": "10050" 
  } 
],
"tasks": [ 
  { 
    "type": 0,
    "status": 0,
    "parent_taskid": 10 
  }, 
  { 
    "type": 0,
    "status": 1, 
  } 
}
"error": "No permissions to execute task.",
  "parent_taskid": 20
}
]
"version":"5.4.0"
}

server→proxy:
{
  "response": "success",
  "tasks": [
    {
      "type": 1,
      "clock": 1478608371,
      "ttl": 600,
      "commandtype": 2,
      "command": "restart_service1.sh",
      "execute_on": 2,
      "port": 80,
      "authtype": 0,
      "hostname": "userA",
      "password": "password1",
      "publickey": "MI...",
      "privatekey": "lsuusFncCzWBQ7RKNUSEsmQRMGkVb1/3j+skZ6UtW+5u091HNsJ6tQ5QcGk0n7zh",
      "parent_taskid": 10,
      "hostid": 10070
    },
    {
      "type": 1,
      "clock": 1478608381,
      "ttl": 600,
      "commandtype": 1,
      "command": "restart_service2.sh",
      "execute_on": 0,
      "authtype": 0,
      "username": "",
      "password": "",
      "publickey": "",
      "privatekey": "",
      "parent_taskid": 20,
      "hostid": 10084
    }
  ]
}

Active proxy

Proxy heartbeat request

The proxy heartbeat request is sent by proxy to report that proxy is running. This request is sent every HeartbeatFrequency (proxy configuration parameter) seconds.

<table>
<thead>
<tr>
<th>name</th>
<th>value type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proxy→server:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>request</td>
<td>string</td>
<td>'proxy heartbeat’</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>the proxy name</td>
</tr>
<tr>
<td>version</td>
<td>string</td>
<td>the proxy version (&lt;major&gt;.&lt;minor&gt;.&lt;build&gt;)</td>
</tr>
<tr>
<td>server→proxy:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>response</td>
<td>string</td>
<td>the request success information ('success' or 'failed')</td>
</tr>
</tbody>
</table>

proxy→server:
Proxy config request

The proxy config request is sent by proxy to obtain proxy configuration data. This request is sent every ConfigFrequency (proxy configuration parameter) seconds.

<table>
<thead>
<tr>
<th>name</th>
<th>value type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proxy→server:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>request</td>
<td>string</td>
<td>'proxy config'</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>proxy name</td>
</tr>
<tr>
<td>version</td>
<td>string</td>
<td>the proxy version (&lt;major&gt;.&lt;minor&gt;.&lt;build&gt;)</td>
</tr>
<tr>
<td>server→proxy:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>request</td>
<td>string</td>
<td>'proxy config'</td>
</tr>
<tr>
<td>&lt;table&gt;</td>
<td>object</td>
<td>one or more objects with &lt;table&gt; data</td>
</tr>
<tr>
<td>fields</td>
<td>array</td>
<td>array of field names</td>
</tr>
<tr>
<td></td>
<td>string</td>
<td>field name</td>
</tr>
<tr>
<td>data</td>
<td>array</td>
<td>array of rows</td>
</tr>
<tr>
<td></td>
<td>array</td>
<td>array of columns</td>
</tr>
<tr>
<td></td>
<td>string,number</td>
<td>column value with type depending on column type in database schema</td>
</tr>
<tr>
<td>proxy→server:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>response</td>
<td>string</td>
<td>the request success information ('success' or 'failed')</td>
</tr>
</tbody>
</table>

Example:

proxy→server:

```
{
  "request": "proxy config",
  "host": "Proxy #12",
  "version": "5.4.0"
}
```

server→proxy:

```
{
  "globalmacro":{
    "fields":[
      "globalmacroid",
      "macro",
      "value"
    ],
    "data":[
      [
        2,
        "${SNMP_COMMUNITY}"
      ]
    ]
  },
  "hosts":{
    "fields":[
      "hostid",
      "host",
      "version"
    ],
    "data":[
      [
        "192.168.1.123",
        "Proxy #12",
        "5.4.0"
      ]
    ]
  }
}
```

1452
"status",
"ipmi_authtype",
"ipmi_privilege",
"ipmi_username",
"ipmi_password",
"name",
"tls_connect",
"tls_accept",
"tls_issuer",
"tls_subject",
"tls_psk_identity",
"tls_psk"
],
"data": [
[
10001,
"Linux",
3,
-1,
2,
"",
"",
"Linux",
1,
1,
"",
"",
"",
""
],
[
10050,
"Zabbix Agent",
3,
-1,
2,
"",
"",
"Zabbix Agent",
1,
1,
"",
"",
"",
"
],
[
10105,
"Logger",
0,
-1,
2,
"",
"",
"Logger",
1,
1,
"",
"",
"",
"
]
Proxy data request

The proxy data request is sent by proxy to provide host interface availability, history, discovery and autoregistration data. This request is sent every DataSenderFrequency (proxy configuration parameter) seconds.

<table>
<thead>
<tr>
<th>name</th>
<th>value type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proxy→server:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>request</td>
<td>string</td>
<td>'proxy data'</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>the proxy name</td>
</tr>
<tr>
<td>session</td>
<td>string</td>
<td>data session token</td>
</tr>
<tr>
<td>interface avail-</td>
<td>array</td>
<td>(optional) array of interface availability data objects</td>
</tr>
<tr>
<td>ability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>interfaceid</td>
<td>number</td>
<td>interface identifier</td>
</tr>
<tr>
<td>available</td>
<td>number</td>
<td>Interface availability</td>
</tr>
<tr>
<td>0, INTERFACE_AVAILABLE_UNKNOWN - unknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1, INTERFACE_AVAILABLE_TRUE - available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2, INTERFACE_AVAILABLE_FALSE - unavailable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>error</td>
<td>string</td>
<td>Interface error message or empty string</td>
</tr>
<tr>
<td>history</td>
<td>array</td>
<td>(optional) array of history data objects</td>
</tr>
<tr>
<td>itemid</td>
<td>number</td>
<td>item identifier</td>
</tr>
<tr>
<td>clock</td>
<td>number</td>
<td>item value timestamp (seconds)</td>
</tr>
<tr>
<td>ns</td>
<td>number</td>
<td>item value timestamp (nanoseconds)</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>(optional) item value</td>
</tr>
<tr>
<td>name</td>
<td>value type</td>
<td>description</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>id</td>
<td>number</td>
<td>value identifier (ascending counter, unique within one data session)</td>
</tr>
<tr>
<td>timestamp</td>
<td>number</td>
<td>(optional) timestamp of log type items</td>
</tr>
<tr>
<td>source</td>
<td>string</td>
<td>(optional) eventlog item source value</td>
</tr>
<tr>
<td>severity</td>
<td>number</td>
<td>(optional) eventlog item severity value</td>
</tr>
<tr>
<td>eventid</td>
<td>number</td>
<td>(optional) eventlog item eventid value</td>
</tr>
<tr>
<td>state</td>
<td>string</td>
<td>(optional) item state</td>
</tr>
<tr>
<td>lastlogsize</td>
<td>number</td>
<td>(optional) last log size of log type items</td>
</tr>
<tr>
<td>mtime</td>
<td>number</td>
<td>(optional) modify time of log type items</td>
</tr>
<tr>
<td>discovery</td>
<td>array</td>
<td>(optional) array of discovery data objects</td>
</tr>
<tr>
<td>clock</td>
<td>number</td>
<td>the discovery data timestamp</td>
</tr>
<tr>
<td>druleid</td>
<td>number</td>
<td>the discovery rule identifier</td>
</tr>
<tr>
<td>dcheckid</td>
<td>number</td>
<td>the discovery check identifier or null for discovery rule data</td>
</tr>
<tr>
<td>type</td>
<td>number</td>
<td>the discovery check type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1 discovery rule data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0. SVC_SSH - SSH service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. SVC_LDAP - LDAP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. SVC_SMTP - SMTP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. SVC_FTP - FTP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. SVC_HTTP - HTTP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. SVC_POP - POP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. SVC_NNTP - NNTP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7. SVC_IMAP - IMAP service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8. SVC_TCP - TCP port availability check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9. SVC_AGENT - Zabbix agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10. SVC_SNMPv1 - SNMPv1 agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11. SVC_SNMPv2 - SNMPv2 agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12. SVC_ICMPPING - ICMP ping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13. SVC_SNMPv3 - SNMPv3 agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14. SVC_HTTPS - HTTPS service check</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15. SVC_TELNET - Telnet availability check</td>
</tr>
<tr>
<td>ip</td>
<td>string</td>
<td>the host IP address</td>
</tr>
<tr>
<td>dns</td>
<td>string</td>
<td>the host DNS name</td>
</tr>
<tr>
<td>port</td>
<td>number</td>
<td>(optional) service port number</td>
</tr>
<tr>
<td>key_</td>
<td>string</td>
<td>(optional) the item key for discovery check of type 9 SVC_AGENT</td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>(optional) value received from the service, can be empty for most of services</td>
</tr>
<tr>
<td>status</td>
<td>number</td>
<td>(optional) service status:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0. DOBJECT_STATUS_UP - Service UP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. DOBJECT_STATUS_DOWN - Service DOWN</td>
</tr>
<tr>
<td>autoregistration</td>
<td>array</td>
<td>(optional) array of autoregistration data objects</td>
</tr>
<tr>
<td>clock</td>
<td>number</td>
<td>the autoregistration data timestamp</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>(optional) the host IP address</td>
</tr>
<tr>
<td>ip</td>
<td>string</td>
<td>(optional) the resolved DNS name from IP address</td>
</tr>
<tr>
<td>dns</td>
<td>string</td>
<td>(optional) the host port</td>
</tr>
<tr>
<td>port</td>
<td>string</td>
<td>(optional) the host metadata sent by agent (based on HostMetadata or HostMetadataItem agent configuration parameter)</td>
</tr>
<tr>
<td>host_metadata</td>
<td>string</td>
<td>(optional) the host metadata sent by agent (based on HostMetadata or HostMetadataItem agent configuration parameter)</td>
</tr>
<tr>
<td>tasks</td>
<td>array</td>
<td>(optional) array of tasks</td>
</tr>
<tr>
<td>type</td>
<td>number</td>
<td>the task type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0. ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT - remote command result</td>
</tr>
<tr>
<td>status</td>
<td>number</td>
<td>the remote command execution status:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0. ZBX_TM_REMOTE_COMMAND_COMPLETED - the remote command completed successfully</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. ZBX_TM_REMOTE_COMMAND_FAILED - the remote command failed</td>
</tr>
<tr>
<td>name</td>
<td>value type</td>
<td>description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>error</td>
<td>string</td>
<td>(optional) the error message</td>
</tr>
<tr>
<td>parent_taskid</td>
<td>number</td>
<td>the parent task id</td>
</tr>
<tr>
<td>more</td>
<td>number</td>
<td>(optional) 1 - there are more history data to send</td>
</tr>
<tr>
<td>clock</td>
<td>number</td>
<td>(optional) data transfer timestamp (seconds)</td>
</tr>
<tr>
<td>ns</td>
<td>number</td>
<td>(optional) data transfer timestamp (nanoseconds)</td>
</tr>
<tr>
<td>version</td>
<td>string</td>
<td>the proxy version (&lt;major&gt;.&lt;minor&gt;.&lt;build&gt;)</td>
</tr>
<tr>
<td>server→proxy:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>response</td>
<td>string</td>
<td>the request success information ('success' or 'failed')</td>
</tr>
<tr>
<td>upload</td>
<td>string</td>
<td>historical data (history, auto-registration, host availability, network</td>
</tr>
<tr>
<td></td>
<td></td>
<td>discovery) upload control. Possible values - 'enabled' or 'disabled'</td>
</tr>
<tr>
<td>tasks</td>
<td>array</td>
<td>(optional) array of tasks</td>
</tr>
<tr>
<td>type</td>
<td>number</td>
<td>the task type:</td>
</tr>
<tr>
<td>clock</td>
<td>number</td>
<td>the task creation time</td>
</tr>
<tr>
<td>ttl</td>
<td>number</td>
<td>the time in seconds after which task expires</td>
</tr>
<tr>
<td>commandtype</td>
<td>number</td>
<td>the remote command type:</td>
</tr>
<tr>
<td>command</td>
<td>string</td>
<td>the remote command to execute</td>
</tr>
<tr>
<td>execute_on</td>
<td>number</td>
<td>the execution target for custom scripts:</td>
</tr>
<tr>
<td>port</td>
<td>number</td>
<td>(optional) the port for telnet and ssh commands</td>
</tr>
<tr>
<td>authtype</td>
<td>number</td>
<td>(optional) the authentication type for ssh commands</td>
</tr>
<tr>
<td>username</td>
<td>string</td>
<td>(optional) the user name for telnet and ssh commands</td>
</tr>
<tr>
<td>password</td>
<td>string</td>
<td>(optional) the password for telnet and ssh commands</td>
</tr>
<tr>
<td>publickey</td>
<td>string</td>
<td>(optional) the public key for ssh commands</td>
</tr>
<tr>
<td>privatekey</td>
<td>string</td>
<td>(optional) the private key for ssh commands</td>
</tr>
<tr>
<td>parent_taskid</td>
<td>number</td>
<td>the parent task id</td>
</tr>
<tr>
<td>hostid</td>
<td>number</td>
<td>target hostid</td>
</tr>
</tbody>
</table>

Example:

```
proxy→server:
{
    "request": "proxy data",
    "host": "Proxy #12",
    "session": "12345678901234567890123456789012",
    "interface availability": [
        {
            "interfaceid": 1,
            "available": 1,
            "error": ""
        },
        {
            "interfaceid": 2,
            "available": 2,
            "error": "Get value from agent failed: cannot connect to [[127.0.0.1]:10049]: [111] Connection refused"
        },
        {
            "interfaceid": 3,
            "available": 1,
            "error": ""
        }
    ]
}
```
"type": 2,
"clock": 1478608371,
"ttl": 600,
"commandtype": 2,
"command": "restart_service1.sh",
"execute_on": 2,
"port": 80,
"authtype": 0,
"username": "userA",
"password": "password1",
"publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zh2j6+H0qtjTkVzwTCpvKe",
"privatekey": "lsuusFncCzWBQ7RKNUSEsmQRMSGkVb1/3j+skZ6UtW+5u9lHNaJ6tQ5QCqGKukO1De7zh",
"parent_taskid": 10,
"hostid": 10070
},
{
"type": 2,
"clock": 1478608381,
"ttl": 600,
"commandtype": 1,
"command": "restart_service2.sh",
"execute_on": 0,
"authtype": 0,
"username": "",
"password": "",
"publickey": "",
"privatekey": "",
"parent_taskid": 20,
"hostid": 10084
}
],
"tasks": [
{
"type": 0,
"status": 0,
"parent_taskid": 10
},
{
"type": 0,
"status": 1,
"error": "No permissions to execute task."
,
"parent_taskid": 20
}
],
"version": "5.4.0"
}

server→proxy:

{
"response": "success",
"upload": "enabled",
"tasks": [
{
"type": 1,
"clock": 1478608371,
"ttl": 600,
"commandtype": 2,
"command": "restart_service1.sh",
"execute_on": 2,
"port": 80,
"authtype": 0,
"username": "userA",
"password": "password1",
"publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zh2j6+H0qtjTkVzwTCpvKe",
"privatekey": "lsuusFncCzWBQ7RKNUSEsmQRMSGkVb1/3j+skZ6UtW+5u9lHNaJ6tQ5QCqGKukO1De7zh",
"parent_taskid": 10,
"hostid": 10070
}
]
2 Zabbix agent protocol

Please refer to Passive and active agent checks page for more information.

3 Zabbix agent 2 protocol

Overview

This section provides information on:

- Agent2 -> Server: active checks request
- Server -> Agent2: active checks response
- Agent2 -> Server: agent data request
- Server -> Agent2: agent data response
- Agent2 -> Server: heartbeat message

Active checks request

The active checks request is used to obtain the active checks to be processed by agent. This request is sent by the agent upon start and then with RefreshActiveChecks intervals.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Mandatory</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>string</td>
<td>yes</td>
<td>active checks</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>yes</td>
<td>Host name.</td>
</tr>
<tr>
<td>version</td>
<td>string</td>
<td>yes</td>
<td>The agent version: &lt;major&gt;.&lt;minor&gt;.</td>
</tr>
<tr>
<td>host_metadata</td>
<td>string</td>
<td>no</td>
<td>The configuration parameter HostMetadata or HostMetadataItem metric value.</td>
</tr>
<tr>
<td>interface</td>
<td>string</td>
<td>no</td>
<td>The configuration parameter HostInterface or HostInterfaceItem metric value.</td>
</tr>
<tr>
<td>ip</td>
<td>string</td>
<td>no</td>
<td>The configuration parameter ListenIP first IP if set.</td>
</tr>
<tr>
<td>port</td>
<td>number</td>
<td>no</td>
<td>The configuration parameter ListenPort value if set and not default agent listening port.</td>
</tr>
</tbody>
</table>

Example:

```
{
  "request": "active checks",
  "host": "Zabbix server",
  "clock": 1478608381,
  "ttl": 600,
  "command": "restart_service2.sh",
  "execute_on": 0,
  "authtype": 0,
  "username": "",
  "password": "",
  "publickey": "",
  "privatekey": "",
  "parent_taskid": 20,
  "hostid": 10084
}
```
"version": "6.0",
"host_metadata": "mysql,nginx",
"hostinterface": "zabbix.server.lan",
"ip": "159.168.1.1",
"port": 12050
}

Active checks response

The active checks response is sent by the server back to agent after processing active checks request.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Mandatory</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>string</td>
<td>yes</td>
<td>success</td>
</tr>
<tr>
<td>info</td>
<td>string</td>
<td>no</td>
<td>Error information in the case of failure.</td>
</tr>
<tr>
<td>data</td>
<td>array of objects</td>
<td>no</td>
<td>Active check items.</td>
</tr>
<tr>
<td>key</td>
<td>string</td>
<td>no</td>
<td>Item key with expanded macros.</td>
</tr>
<tr>
<td>itemid</td>
<td>number</td>
<td>no</td>
<td>Item identifier.</td>
</tr>
<tr>
<td>delay</td>
<td>string</td>
<td>no</td>
<td>Item update interval.</td>
</tr>
<tr>
<td>lastlogsize</td>
<td>number</td>
<td>no</td>
<td>Item lastlogsize.</td>
</tr>
<tr>
<td>mtime</td>
<td>number</td>
<td>no</td>
<td>Item mtime.</td>
</tr>
<tr>
<td>regexp</td>
<td>array of objects</td>
<td>no</td>
<td>Global regular expressions.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>no</td>
<td>Global regular expression name.</td>
</tr>
<tr>
<td>expression</td>
<td>string</td>
<td>no</td>
<td>Global regular expression.</td>
</tr>
<tr>
<td>expression_type</td>
<td>number</td>
<td>no</td>
<td>Global regular expression type.</td>
</tr>
<tr>
<td>exp_delimiter</td>
<td>string</td>
<td>no</td>
<td>Global regular expression delimiter.</td>
</tr>
<tr>
<td>case_sensitive</td>
<td>number</td>
<td>no</td>
<td>Global regular expression case sensitiviness setting.</td>
</tr>
</tbody>
</table>

Example:

```json
{
  "response": "success",
  "data": [
    {
      "key": "log[/home/zabbix/logs/zabbix_agentd.log]",
      "itemid": 1234,
      "delay": "30s",
      "lastlogsize": 0,
      "mtime": 0
    },
    {
      "key": "agent.version",
      "itemid": 5678,
      "delay": "10m",
      "lastlogsize": 0,
      "mtime": 0
    }
  ]
}
```

Agent data request

The agent data request contains the gathered item values.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Mandatory</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>string</td>
<td>yes</td>
<td>agent data</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>yes</td>
<td>Host name.</td>
</tr>
<tr>
<td>version</td>
<td>string</td>
<td>yes</td>
<td>The agent version: &lt;major&gt;.&lt;minor&gt;.</td>
</tr>
<tr>
<td>session</td>
<td>string</td>
<td>yes</td>
<td>Unique session identifier generated each time when agent is started.</td>
</tr>
</tbody>
</table>
### Field | Type | Mandatory | Value
--- | --- | --- | ---
**data** | array of objects | yes | Item values.
  **id** | number | yes | The value identifier (incremental counter used for checking duplicated values in the case of network problems).
  **itemid** | number | yes | Item identifier.
  **value** | string | no | The item value.
  **lastlogsize** | number | no | The item lastlogsize.
  **mtime** | number | no | The item mtime.
  **state** | number | no | The item state.
  **source** | string | no | The item event log source.
  **eventid** | number | no | The item event log eventid.
  **severity** | number | no | The item event log severity.
  **timestamp** | number | no | The item event log timestamp.
  **clock** | number | yes | The value timestamp (seconds since Epoch).
  **ns** | number | yes | The value timestamp nanoseconds.

**Example:**

```json
{
  "request": "agent data",
  "data": [
    {
      "id": 1,
      "itemid": 5678,
      "value": "2.4.0",
      "clock": 1400675595,
      "ns": 7680644
    },
    {
      "id": 2,
      "itemid": 1234,
      "lastlogsize": 112,
      "value": "19845:20140621:141708.521 Starting Zabbix Agent [<hostname>]. Zabbix 2.4.0 (revision 50000).",
      "clock": 1400675595,
      "ns": 77053975
    }
  ],
  "host": "Zabbix server",
  "version": "6.0",
  "session": "1234456akdsjhfoui"
}
```

**Agent data response**

The agent data response is sent by the server back to agent after processing the agent data request.

### Field | Type | Mandatory | Value
--- | --- | --- | ---
**response** | string | yes | success | failed
**info** | string | yes | Item processing results.

**Example:**

```json
{
  "response": "success",
  "info": "processed: 2; failed: 0; total: 2; seconds spent: 0.003534"
}
```

**Heartbeat message**

The heartbeat message is sent by an active agent to Zabbix server/proxy every HeartbeatFrequency seconds (configured in the Zabbix agent 2 configuration file).
It is used to monitor the availability of active checks.

```json
{
    "request": "active check heartbeat",
    "host": "Zabbix server",
    "heartbeat_freq": 60
}
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Mandatory</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>string</td>
<td>yes</td>
<td>active check heartbeat</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>yes</td>
<td>The host name.</td>
</tr>
<tr>
<td>heartbeat_freq</td>
<td>number</td>
<td>yes</td>
<td>The agent heartbeat frequency (HeartbeatFrequency configuration parameter).</td>
</tr>
</tbody>
</table>

4 Zabbix sender protocol

Please refer to the trapper item page for more information.

5 Header

Overview

The header is present in response and request messages between Zabbix components. It is required to determine the length of message, if it is compressed or not and the format of message length fields. The header consists of:

<PROTOCOL> - "ZBXD" (4 bytes).
<FLAGS> - the protocol flags, (1 byte). 0x01 - Zabbix communications protocol, 0x02 - compression, 0x04 - large packet.
<DATALEN> - data length (4 bytes or 8 bytes for large packet). I will be formatted as 01/00/00/00 (four bytes, 32 bit number in little-endian format) or 01/00/00/00/00/00/00/00 (eight bytes, 64 bit number in little-endian format) for large packet.
<RESERVED> - uncompressed data length (4 bytes or 8 bytes for large packet). I will be formatted as 01/00/00/00 (four bytes, 32 bit number in little-endian format) or 01/00/00/00/00/00/00/00 (eight bytes, 64 bit number in little-endian format) for large packet.

When compression is enabled (0x02 flag) the <RESERVED> bytes contains uncompressed data size. When compression is not enabled then <RESERVED> should be zeroes.

Zabbix protocol has 1GB packet size limit per connection. The limit of 1GB is applied for received packet data length and for uncompressed data length, however, when large packet is enabled (0x04 flag) it is possible for Zabbix proxy to receive configuration with size up to 16GB; note that large packet can only be used for Zabbix proxy configuration, and Zabbix server will automatically set (0x04 flag) and send length fields as 8 bytes each when data length before compression exceeds 4GB.

Implementation

Here are the code snippets showing how to add Zabbix protocol header to the data you want to send in order to obtain the packet you should send to Zabbix so that it is interpreted correctly.

Bash

```bash
printf -v LENGTH '%016x' "${#DATA}"
PACK=""
for (( i=14; i>=0; i-=2 ))
doi
    PACK="$PACK\x${LENGTH:$i:2}"
done
printf "ZBXD\1$PACK\"" "$DATA"
```

Java

```java
byte[] header = new byte[] {'Z', 'B', 'X', 'D', '\1',(byte)(data.length & 0xFF),(byte)((data.length >> 8)};
byte[] packet = new byte[header.length + data.length];
System.arraycopy(header, 0, packet, 0, header.length);
System.arraycopy(data, 0, packet, header.length, data.length);
```

or

PHP

```php
$packet = "ZBXD\1" . pack('P', strlen($data)) . $data;
```
Perl
my $packet = "ZBXD\1" . pack('V', strlen($data)) . "\0\0\0\0" . $data;

Python
packet = "ZBXD\1" + struct.pack('<Q', len(data)).decode() + data

6 Real-time export protocol

This section presents details of the real-time export protocol in a newline-delimited JSON format for:

- trigger events
- item values
- trends

All files have a .ndjson extension. Each line of the export file is a JSON object.

Trigger events

The following information is exported for a problem event:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock</td>
<td>number</td>
<td>Number of seconds since Epoch to the moment when problem was detected (integer part).</td>
</tr>
<tr>
<td>ns</td>
<td>number</td>
<td>Number of nanoseconds to be added to clock to get a precise problem detection time.</td>
</tr>
<tr>
<td>value</td>
<td>number</td>
<td>1 (always).</td>
</tr>
<tr>
<td>eventid</td>
<td>number</td>
<td>Problem event ID.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Problem event name.</td>
</tr>
<tr>
<td>severity</td>
<td>number</td>
<td>Problem event severity (0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster).</td>
</tr>
<tr>
<td>hosts</td>
<td>array</td>
<td>List of hosts involved in the trigger expression; there should be at least one element in array.</td>
</tr>
<tr>
<td>- host</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>Host name.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Visible host name.</td>
</tr>
<tr>
<td>groups</td>
<td>array</td>
<td>List of host groups of all hosts involved in the trigger expression; there should be at least one element in array.</td>
</tr>
<tr>
<td>-</td>
<td>array</td>
<td></td>
</tr>
<tr>
<td></td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>tag</td>
<td>array</td>
<td>List of problem tags (can be empty).</td>
</tr>
<tr>
<td>-</td>
<td>array</td>
<td></td>
</tr>
<tr>
<td>tag</td>
<td>object</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td>string</td>
<td>Tag name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following information is exported for a recovery event:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock</td>
<td>number</td>
<td>Number of seconds since Epoch to the moment when problem was resolved (integer part).</td>
</tr>
<tr>
<td>ns</td>
<td>number</td>
<td>Number of nanoseconds to be added to clock to get a precise problem resolution time.</td>
</tr>
<tr>
<td>value</td>
<td>number</td>
<td>0 (always).</td>
</tr>
<tr>
<td>eventid</td>
<td>number</td>
<td>Recovery event ID.</td>
</tr>
<tr>
<td>p_eventid</td>
<td>number</td>
<td>Problem event ID.</td>
</tr>
</tbody>
</table>

Examples
Problem:
{"clock":1519304285,"ns":123456789,"value":1,"name":"Either Zabbix agent is unreachable on Host B or pollers are too busy ... Y","Group Z","Zabbix servers"},
"tags": ["tag":"availability","value":""],
"tags": ["tag":"data center","value":"Riga"]

Recovery:
{"clock":1519304345,"ns":987654321,"value":0,"eventid":43,"p_eventid":42}

Problem (multiple problem event generation):
{"clock":1519304286,"ns":123456789,"value":1,"eventid":43,"name":"Either Zabbix agent is unreachable on Host B or pollers are too busy ... Y","Group Z","Zabbix servers"},
"tags": ["tag":"availability","value":""],
"tags": ["tag":"data center","value":"Riga"]

Recovery:
{"clock":1519304346,"ns":987654321,"value":0,"eventid":44,"p_eventid":43}

Item values

The following information is exported for a collected item value:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>object</td>
<td>Host name of the item host.</td>
</tr>
<tr>
<td>host name</td>
<td>string</td>
<td>Host name.</td>
</tr>
<tr>
<td>groups</td>
<td>array</td>
<td>List of host groups of the item host; there should be at least one element in array.</td>
</tr>
<tr>
<td>itemid</td>
<td>number</td>
<td>Item ID.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Visible item name.</td>
</tr>
<tr>
<td>clock</td>
<td>number</td>
<td>Number of seconds since Epoch to the moment when value was collected (integer part).</td>
</tr>
<tr>
<td>ns</td>
<td>number</td>
<td>Number of nanoseconds to be added to clock to get a precise value collection time.</td>
</tr>
<tr>
<td>timestamp</td>
<td>number</td>
<td>0 if not available.</td>
</tr>
<tr>
<td>source</td>
<td>string</td>
<td>Empty string if not available.</td>
</tr>
<tr>
<td>severity</td>
<td>number</td>
<td>0 if not available.</td>
</tr>
<tr>
<td>eventid</td>
<td>number</td>
<td>0 if not available.</td>
</tr>
<tr>
<td>value</td>
<td>number (for numeric items) or string (for text items)</td>
<td>Collected item value.</td>
</tr>
<tr>
<td>type</td>
<td>number</td>
<td>Collected value type: 0 - numeric float, 1 - character, 2 - log, 3 - numeric unsigned, 4 - text</td>
</tr>
</tbody>
</table>

Examples

Numeric (unsigned) value:
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"itemid":3,"name":"Agent availability","clock":1519304285,"ns":123456789,"value":1,"type":3}

Numeric (float) value:
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"itemid":4,"name":"CPU Load","clock":1519304285,"ns":123456789,"value":0.1,"type":0}

Character, text value:
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"itemid":2,"name":"Agent version","clock":1519304285,"ns":123456789,"value":"3.4.4","type":4}
Log value:
{"host":{"host":"Host A","name":"Host A visible"},"groups":["Group X","Group Y","Group Z"],"itemid":1,"name":"Messages
file message"}

Trends

The following information is exported for a calculated trend value:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>object</td>
<td>Host name of the item host.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Host name.</td>
</tr>
<tr>
<td>groups</td>
<td>array</td>
<td>List of host groups of the item host; there should be at least one element in array.</td>
</tr>
<tr>
<td>itemid</td>
<td>number</td>
<td>Item ID.</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Visible item name.</td>
</tr>
<tr>
<td>clock</td>
<td>number</td>
<td>Number of seconds since Epoch to the moment when value was collected (integer part).</td>
</tr>
<tr>
<td>count</td>
<td>number</td>
<td>Number of values collected for a given hour.</td>
</tr>
<tr>
<td>min</td>
<td>number</td>
<td>Minimum item value for a given hour.</td>
</tr>
<tr>
<td>avg</td>
<td>number</td>
<td>Average item value for a given hour.</td>
</tr>
<tr>
<td>max</td>
<td>number</td>
<td>Maximum item value for a given hour.</td>
</tr>
</tbody>
</table>
| type   | number| Value type:
|        |       | 0 - numeric float, 3 - numeric unsigned                                      |

Examples

Numeric (unsigned) value:
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"itemid":3,"name":"Agent availability"}

Numeric (float) value:
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"itemid":4,"name":"CPU Load"}

5 Items

1 vm.memory.size parameters

Overview

This section provides some parameter details for the `vm.memory.size[<mode>]` agent item.

Parameters

The following parameters are available for this item:

- active - memory currently in use or very recently used, and so it is in RAM
- anon - memory not associated with a file (cannot be re-read from it)
- available - available memory, calculated differently depending on the platform (see the table below)
- buffers - cache for things like file system metadata
- cached - cache for various things
- exec - executable code, typically from a (program) file
- file - cache for contents of recently accessed files
- free - memory that is readily available to any entity requesting memory
- inactive - memory that is marked as not used
- pavailable - 'available' memory as percentage of 'total' (calculated as available/total*100)
- pinned - same as 'wired'
- pused - 'used' memory as percentage of 'total' (calculated as used/total*100)
- shared - memory that may be simultaneously accessed by multiple processes
- slab - total amount of memory used by the kernel to cache data structures for its own use
- total - total physical memory available
- used - used memory, calculated differently depending on the platform (see the table below)
• **wired** - memory that is marked to always stay in RAM. It is never moved to disk.

Some of these parameters are platform-specific and might not be available on your platform. See Zabbix agent items for details.

Platform-specific calculation of available and used:

<table>
<thead>
<tr>
<th>Platform</th>
<th>&quot;available&quot;</th>
<th>&quot;used&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>free + cached</td>
<td>real memory in use</td>
</tr>
<tr>
<td>FreeBSD</td>
<td>inactive + cached + free</td>
<td>active + wired + cached</td>
</tr>
<tr>
<td>HP UX</td>
<td>free</td>
<td>total - free</td>
</tr>
<tr>
<td>Linux&lt;3.14</td>
<td>free + buffers + cached</td>
<td>total - free</td>
</tr>
<tr>
<td>Linux 3.14+</td>
<td>/proc/meminfo, see &quot;MemAvailable&quot; in Linux kernel documentation for details.</td>
<td>total - free</td>
</tr>
<tr>
<td>(also backported to 3.10 on RHEL 7)</td>
<td>Note that free + buffers + cached is no longer equal to 'available' due to not all the page cache can be freed and low watermark being used in calculation.</td>
<td>total - free</td>
</tr>
<tr>
<td>NetBSD</td>
<td>inactive + execpages + file + free</td>
<td>total - free</td>
</tr>
<tr>
<td>OpenBSD</td>
<td>inactive + free + cached</td>
<td>active + wired</td>
</tr>
<tr>
<td>OSX</td>
<td>inactive + free</td>
<td>active + wired</td>
</tr>
<tr>
<td>Solaris</td>
<td>free</td>
<td>total - free</td>
</tr>
<tr>
<td>Win32</td>
<td>free</td>
<td>total - free</td>
</tr>
</tbody>
</table>

The sum of vm.memory.size[used] and vm.memory.size[available] does not necessarily equal total. For instance, on FreeBSD:

* Active, inactive, wired, cached memories are considered used, because they store some useful information.

* At the same time inactive, cached, free memories are considered available, because these kinds of memories can be given instantaneously to processes that request more memory.

So inactive memory is both used and available simultaneously. Because of this, the vm.memory.size[used] item is designed for informational purposes only, while vm.memory.size[available] is designed to be used in triggers.

See also

1. Additional details about memory calculation in different OS

2 Passive and active agent checks

Overview

This section provides details on passive and active checks performed by Zabbix agent.

Zabbix uses a JSON based communication protocol for communicating with Zabbix agent.

See also: Zabbix agent 2 protocol details.

Passive checks

A passive check is a simple data request. Zabbix server or proxy asks for some data (for example, CPU load) and Zabbix agent sends back the result to the server.

**Server request**

For definition of header and data length please refer to protocol details.

```
<item key>
```

**Agent response**

```
<Data>[
\0<ERROR>
]
```

Above, the part in square brackets is optional and is only sent for not supported items.

For example, for supported items:

1. Server opens a TCP connection
2. Server sends `<HEADER><DATALEN>agent.ping`
3. Agent reads the request and responds with `<HEADER><DATALEN>1`
4. Server processes data to get the value, ‘1’ in our case
5. TCP connection is closed

For not supported items:
1. Server opens a TCP connection
2. Server sends `<HEADER><DATALEN>vfs.fs.size[/nono]`
3. Agent reads the request and responds with `<HEADER><DATALEN>ZBX_NOTSUPPORTED\0Cannot obtain filesystem information: [2] Nosuchfileordirectory`
4. Server processes data, changes item state to not supported with the specified error message
5. TCP connection is closed

Active checks

Active checks require more complex processing. The agent must first retrieve from the server(s) a list of items for independent processing.

The servers to get the active checks from are listed in the 'ServerActive' parameter of the agent configuration file. The frequency of asking for these checks is set by the 'RefreshActiveChecks' parameter in the same configuration file. However, if refreshing active checks fails, it is retried after hardcoded 60 seconds.

The agent then periodically sends the new values to the server(s).

If an agent is behind the firewall you might consider using only Active checks because in this case you wouldn’t need to modify the firewall to allow initial incoming connections.

Getting the list of items

**Agent request**

The active checks request is used to obtain the active checks to be processed by agent. This request is sent by the agent upon start and then with RefreshActiveChecks intervals.

```json
{
    "request": "active checks",
    "host": "Zabbix server",
    "host_metadata": "mysql,nginx",
    "hostinterface": "zabbix.server.lan",
    "ip": "159.168.1.1",
    "port": 12050
}
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Mandatory</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>string</td>
<td>yes</td>
<td>active checks</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>yes</td>
<td>Host name.</td>
</tr>
<tr>
<td>host_metadata</td>
<td>string</td>
<td>no</td>
<td>The configuration parameter HostMetadata or HostMetadataltem metric value.</td>
</tr>
<tr>
<td>hostinterface</td>
<td>string</td>
<td>yes</td>
<td>HostInterface or HostInterfaceItem metric value.</td>
</tr>
<tr>
<td>ip</td>
<td>string</td>
<td>no</td>
<td>The configuration parameter ListenIP first IP if set.</td>
</tr>
<tr>
<td>port</td>
<td>number</td>
<td>no</td>
<td>The configuration parameter ListenPort value if set and not default agent listening port.</td>
</tr>
</tbody>
</table>

**Server response**

The active checks response is sent by the server back to agent after processing the active checks request.

```json
{
    "response": "success",
    "data": [
        {
            "key": "log[/home/zabbix/logs/zabbix_agentd.log]",
            "key_orig": "log[/home/zabbix/logs/zabbix_agentd.log]",
            "itemid": 1234,
            "delay": "30s",
            "lastlogsize": 0,
            "mtime": 0
        },
        {
            "key": "agent.version",
            "key_orig": "agent.version",
            "itemid": 5678,
            "delay": "10m",
            "lastlogsize": 0,
        }
    ]
}
```
Field | Type | Mandatory | Value |
--- | --- | --- | --- |
response | string | yes | success | failed |
info | string | no | Error information in the case of failure. |
data | array | no | Active check items. |
key | string | no | Item key with expanded macros. |
key_orig | string | no | Item key without expanded macros. |
itemid | number | no | Item identifier. |
delay | string | no | Item update interval. |
lastlogsize | number | no | Item lastlogsize. |
tmtime | number | no | Item mttime. |
refresh_unsupported | number | no | Unsupported item refresh interval. |
regexp | array | no | Global regular expressions. |
name | string | no | Global regular expression name. |
expression | string | no | Global regular expression. |
expression_type | number | no | Global regular expression type. |
exp_delimiter | string | no | Global regular expression delimiter. |
case_sensitive | number | no | Global regular expression case sensitiveness setting. |

The server must respond with success.

For example:

1. Agent opens a TCP connection
2. Agent asks for the list of checks
3. Server responds with a list of items (item key, delay)
4. Agent parses the response
5. TCP connection is closed
6. Agent starts periodical collection of data

Note that (sensitive) configuration data may become available to parties having access to the Zabbix server trapper port when using an active check. This is possible because anyone may pretend to be an active agent and request item configuration data; authentication does not take place unless you use encryption options.

Sending in collected data

**Agent sends**

The agent data request contains the gathered item values.

```json
{
    "request": "agent data",
    "data": [
        {
            "host": "Zabbix server",
            "key": "agent.version",
            "value": "2.4.0",
            "clock": 1400675595,
            "ns": 76808644
        },
        {
            "host": "Zabbix server",
            "key": "log[/home/zabbix/logs/zabbix_agentd.log]",
            "lastlogsize": 112,
            "value": "19845:20140621:141708.521 Starting Zabbix Agent [<hostname>]. Zabbix 2.4.0 (revision 50000).",
            "clock": 1400675595,
            "ns": 77053975
        }
    ]
}
```
A virtual ID is assigned to each value. Value ID is a simple ascending counter, unique within one data session (identified by the session token). This ID is used to discard duplicate values that might be sent in poor connectivity environments.

**Server response**

The agent data response is sent by the server back to the agent after processing the agent data request.

```json
{
    "response": "success",
    "info": "processed: 2; failed: 0; total: 2; seconds spent: 0.003534"
}
```

::: note
important

If sending of some values fails on the server (for example, because host or item has been disabled or deleted), agent will not retry sending of those values. :::

For example:

1. Agent opens a TCP connection
2. Agent sends a list of values
3. Server processes the data and sends the status back
4. TCP connection is closed

Note how in the example above the not supported status for vfs.fs.size[/nono] is indicated by the “state” value of 1 and the error message in “value” property.

Error message will be trimmed to 2048 symbols on server side.

**Heartbeat message**

The heartbeat message is sent by an active agent to Zabbix server/proxy every HeartbeatFrequency seconds (configured in the Zabbix agent configuration file).

It is used to monitor the availability of active checks.
{  
"request": "active check heartbeat",
"host": "Zabbix server",
"heartbeat_freq": 60
}

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Mandatory</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>string</td>
<td>yes</td>
<td>active check heartbeat</td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td>yes</td>
<td>The hostname</td>
</tr>
<tr>
<td>heartbeat_freq</td>
<td>number</td>
<td>yes</td>
<td>The agent heartbeat frequency (HeartbeatFrequency configuration parameter).</td>
</tr>
</tbody>
</table>

Older XML protocol

Zabbix will take up to 16 MB of XML Base64-encoded data, but a single decoded value should be no longer than 64 KB otherwise it will be truncated to 64 KB while decoding.

3 Trapper items

Overview

Zabbix server uses a JSON-based communication protocol for receiving data from Zabbix sender with the help of trapper item.

Request and response messages must begin with header and data length.

Zabbix sender request

```json
{
    "request": "sender data",
    "data": [
    
    {
    "host": "<hostname>",
    "key": "trap",
    "value": "test value"
    }
    ]
}
```

Zabbix server response

```json
{
    "response": "success",
    "info": "processed: 1; failed: 0; total: 1; seconds spent: 0.060753"
}
```

Zabbix sender request with a timestamp

Alternatively Zabbix sender can send a request with a timestamp and nanoseconds.

```json
{
    "request": "sender data",
    "data": [
    
    {
    "host": "<hostname>",
    "key": "trap",
    "value": "test value",
    "clock": 1516710794,
    "ns": 592397170
    },
    {
    "host": "<hostname>",
    "key": "trap",
    "value": "test value",
    "clock": 1516710795,
    "ns": 192399456
    }
    ]
}
```
Zabbix server response

```
{
  "response":"success",
  "info":"processed: 2; failed: 0; total: 2; seconds spent: 0.060904"
}
```

4 Minimum permission level for Windows agent items

Overview

When monitoring systems using an agent, a good practice is to obtain metrics from the host on which the agent is installed. To use the principle of least privilege, it is necessary to determine what metrics are obtained from the agent.

The table in this document allows you to select the minimum rights for guaranteed correct operation of Zabbix agent.

If a different user is selected for the agent to work, rather than 'LocalSystem', then for the operation of agent as a Windows service, the new user must have the rights "Log on as a service" from "Local Policy→User Rights Assignment" and the right to create, write and delete the Zabbix agent log file. An Active Directory user must be added to the Performance Monitor Users group.

When working with the rights of an agent based on the "minimum technically acceptable" group, prior provision of rights to objects for monitoring is required.

Common agent items supported on Windows

<table>
<thead>
<tr>
<th>Item key</th>
<th>User group</th>
<th>Recommended</th>
<th>Minimum technically acceptable (functionality is limited)</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent.hostname</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>agent.ping</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>agent.variant</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>agent.version</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>log</td>
<td>Administrators</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>log.count</td>
<td>Administrators</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>logrt</td>
<td>Administrators</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>logrt.count</td>
<td>Administrators</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>net.dns</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>net.dns.record</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>net.if.discovery</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>net.if.in</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>net.if.out</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>net.if.total</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>net.tcp.listen</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>net.tcp.port</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>net.tcp.service</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>net.tcp.service.perf</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>net.udp.service</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>net.udp.service.perf</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>proc.num</td>
<td>Administrators</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>system.cpu.discovery</td>
<td>Performance Monitor Users</td>
<td>Performance Monitor Users</td>
<td></td>
</tr>
<tr>
<td>system.cpu.load</td>
<td>Performance Monitor Users</td>
<td>Performance Monitor Users</td>
<td></td>
</tr>
<tr>
<td>system.cpu.num</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>system.cpu.util</td>
<td>Performance Monitor Users</td>
<td>Performance Monitor Users</td>
<td></td>
</tr>
<tr>
<td>system.hostname</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>system.localtime</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>system.run</td>
<td>Administrators</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>system.sw.arch</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>system.swap.size</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>system.uname</td>
<td>Guests</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>system.uptime</td>
<td>Performance Monitor Users</td>
<td>Performance Monitor Users</td>
<td></td>
</tr>
</tbody>
</table>

1471
<table>
<thead>
<tr>
<th>Item key</th>
<th>User group</th>
</tr>
</thead>
<tbody>
<tr>
<td>vfs.dir.count</td>
<td>Administrators</td>
</tr>
<tr>
<td>vfs.dir.get</td>
<td>Administrators</td>
</tr>
<tr>
<td>vfs.dir.size</td>
<td>Administrators</td>
</tr>
<tr>
<td>vfs.file.cksum</td>
<td>Administrators</td>
</tr>
<tr>
<td>vfs.file.contents</td>
<td>Administrators</td>
</tr>
<tr>
<td>vfs.file.exists</td>
<td>Administrators</td>
</tr>
<tr>
<td>vfs.file.md5sum</td>
<td>Administrators</td>
</tr>
<tr>
<td>vfs.file.regex</td>
<td>Administrators</td>
</tr>
<tr>
<td>vfs.file.regmatch</td>
<td>Administrators</td>
</tr>
<tr>
<td>vfs.file.size</td>
<td>Administrators</td>
</tr>
<tr>
<td>vfs.file.time</td>
<td>Administrators</td>
</tr>
<tr>
<td>vfs.fs.discovery</td>
<td>Administrators</td>
</tr>
<tr>
<td>vfs.fs.size</td>
<td>Administrators</td>
</tr>
<tr>
<td>vm.memory.size</td>
<td>Guests</td>
</tr>
<tr>
<td>web.page.get</td>
<td>Guests</td>
</tr>
<tr>
<td>web.page.perf</td>
<td>Guests</td>
</tr>
<tr>
<td>web.page.regexp</td>
<td>Guests</td>
</tr>
<tr>
<td>zabbix.stats</td>
<td>Guests</td>
</tr>
</tbody>
</table>

Windows-specific item keys

<table>
<thead>
<tr>
<th>Item key</th>
<th>User group</th>
<th>Minimum technically acceptable (functionality is limited)</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventlog</td>
<td>Event Log Readers</td>
<td>Guests</td>
</tr>
<tr>
<td>net.if.list</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>perf_counter</td>
<td>Performance Monitor Users</td>
<td>Performance Monitor Users</td>
</tr>
<tr>
<td>proc_info</td>
<td>Administrators</td>
<td>Guests</td>
</tr>
<tr>
<td>service.discovery</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>service.info</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>services</td>
<td>Guests</td>
<td>Guests</td>
</tr>
<tr>
<td>wmi.get</td>
<td>Administrators</td>
<td>Guests</td>
</tr>
<tr>
<td>vm.vmemory.size</td>
<td>Guests</td>
<td>Guests</td>
</tr>
</tbody>
</table>

5 Encoding of returned values

Zabbix server expects every returned text value in the UTF8 encoding. This is related to any type of checks: Zabbix agent, SSH, Telnet, etc.

Different monitored systems/devices and checks can return non-ASCII characters in the value. For such cases, almost all possible zabbix keys contain an additional item key parameter - `<encoding>`. This key parameter is optional but it should be specified if the returned value is not in the UTF8 encoding and it contains non-ASCII characters. Otherwise the result can be unexpected and unpredictable.

A description of behavior with different database backends in such cases follows.

MySQL

If a value contains a non-ASCII character in non UTF8 encoding - this character and the following will be discarded when the database stores this value. No warning messages will be written to the zabbix_server.log.

Relevant for at least MySQL version 5.1.61

PostgreSQL

If a value contains a non-ASCII character in non UTF8 encoding - this will lead to a failed SQL query (PGRES_FATAL_ERROR:ERROR invalid byte sequence for encoding) and data will not be stored. An appropriate warning message will be written to the zabbix_server.log.

Relevant for at least PostgreSQL version 9.1.3

6 Large file support
Large file support, often abbreviated to LFS, is the term applied to the ability to work with files larger than 2 GB on 32-bit operating systems. Since Zabbix 2.0 support for large files has been added. This change affects at least log file monitoring and all vfs.file.* items. Large file support depends on the capabilities of a system at Zabbix compilation time, but is completely disabled on a 32-bit Solaris due to its incompatibility with procfs and swapctl.

## 7 Sensor

Each sensor chip gets its own directory in the sysfs /sys/devices tree. To find all sensor chips, it is easier to follow the device symlinks from /sys/class/hwmon/hwmon*, where * is a real number (0,1,2,...).

The sensor readings are located either in /sys/class/hwmon/hwmon*/ directory for virtual devices, or in /sys/class/hwmon/hwmon*/device directory for non-virtual devices. A file, called name, located inside hwmon* or hwmon*/device directories contains the name of the chip, which corresponds to the name of the kernel driver used by the sensor chip.

There is only one sensor reading value per file. The common scheme for naming the files that contain sensor readings inside any of the directories mentioned above is: `<type><number>_<item>`, where

- **type** - for sensor chips is "in" (voltage), "temp" (temperature), "fan" (fan), etc.,
- **item** - “input” (measured value), “max” (high threshold), “min” (low threshold), etc.,
- **number** - always used for elements that can be present more than once (usually starts from 1, except for voltages which start from 0). If files do not refer to a specific element they have a simple name with no number.

The information regarding sensors available on the host can be acquired using `sensor-detect` and `sensors` tools (lm-sensors package: [http://lm-sensors.org/](http://lm-sensors.org/)). `Sensor-detect` helps to determine which modules are necessary for available sensors. When modules are loaded the `sensors` program can be used to show the readings of all sensor chips. The labeling of sensor readings, used by this program, can be different from the common naming scheme (`<type><number>_<item>`):

- If there is a file called `<type><number>_label`, then the label inside this file will be used instead of `<type><number>_<item>` name;
- If there is no `<type><number>_label` file, then the program searches inside the `/etc/sensors.conf` (could be also `/etc/sensors3.conf`, or different) for the name substitution.

This labeling allows user to determine what kind of hardware is used. If there is neither `<type><number>_label` file nor label inside the configuration file the type of hardware can be determined by the name attribute (hwmon*/device/name). The actual names of sensors, which zabbix_agent accepts, can be obtained by running `sensors` program with `-u` parameter (`sensors -u`).

In `sensors` program the available sensors are separated by the bus type (ISA adapter, PCI adapter, SPI adapter, Virtual device, ACPI interface, HID adapter).

On Linux 2.4:

(Sensor readings are obtained from `/proc/sys/dev/sensors directory)

- **device** - device name (if `<mode>` is used, it is a regular expression);
- **sensor** - sensor name (if `<mode>` is used, it is a regular expression);
- **mode** - possible values: avg, max, min (if this parameter is omitted, device and sensor are treated verbatim).

Example key: `sensor[w83781d-i2c-0-2d,temp1]`

Prior to Zabbix 1.8.4, the `sensor[temp1]` format was used.

On Linux 2.6+:

(Sensor readings are obtained from `/sys/class/hwmon` directory)

- **device** - device name (non regular expression). The device name could be the actual name of the device (e.g 0000:00:18.3) or the name acquired using sensors program (e.g. `k8temp-pci-00c3`). It is up to the user to choose which name to use;
- **sensor** - sensor name (non regular expression);
- **mode** - possible values: avg, max, min (if this parameter is omitted, device and sensor are treated verbatim).

Example key:

`sensor[k8temp-pci-00c3,temp,max]` or `sensor[0000:00:18.3,temp1]`

`sensor[smsc47b397-isa-0880,in,avg]` or `sensor[smsc47b397.2176,in1]`

Obtaining sensor names

Sensor labels, as printed by the sensors command, cannot always be used directly because the naming of labels may be different for each sensor chip vendor. For example, sensors output might contain the following lines:
$ sensors

in0: +2.24 V (min = +0.00 V, max = +3.32 V)
Vcore: +1.15 V (min = +0.00 V, max = +2.99 V)
+3.3V: +3.30 V (min = +2.97 V, max = +3.63 V)
+12V: +13.00 V (min = +0.00 V, max = +15.94 V)

M/B Temp: +30.0°C (low = -127.0°C, high = +127.0°C)

Out of these, only one label may be used directly:

$ zabbix_get -s 127.0.0.1 -k sensor[lm85-i2c-0-2e,in0]
2.240000

Attempting to use other labels (like Vcore or +12V) will not work.

$ zabbix_get -s 127.0.0.1 -k sensor[lm85-i2c-0-2e,Vcore]
ZBX_NOTSUPPORTED

To find out the actual sensor name, which can be used by Zabbix to retrieve the sensor readings, run sensors -u. In the output, the following may be observed:

$ sensors -u

... Vcore:

in1_input: 1.15
in1_min: 0.00
in1_max: 2.99
in1_alarm: 0.00
...

+12V:

in4_input: 13.00
in4_min: 0.00
in4_max: 15.94
in4_alarm: 0.00
...

So Vcore should be queried as in1, and +12V should be queried as in4.\(^6\)

$ zabbix_get -s 127.0.0.1 -k sensor[lm85-i2c-0-2e,in1]
1.301000

Not only voltage (in), but also current (curr), temperature (temp) and fan speed (fan) readings can be retrieved by Zabbix.

### 8 Notes on memtype parameter in proc.mem items

**Overview**

The *memtype* parameter is supported on Linux, AIX, FreeBSD, and Solaris platforms.

Three common values of 'memtype' are supported on all of these platforms: *pmem*, *rss* and *vsize*. Additionally, platform-specific 'memtype' values are supported on some platforms.

**AIX**

See values supported for 'memtype' parameter on AIX in the table.

<table>
<thead>
<tr>
<th>Supported value</th>
<th>Description</th>
<th>Source in procentry64 structure</th>
<th>Tries to be compatible with</th>
</tr>
</thead>
<tbody>
<tr>
<td>vsize(^1)</td>
<td>Virtual memory size</td>
<td>pi_size</td>
<td>ps -o pmem</td>
</tr>
<tr>
<td>pmem</td>
<td>Percentage of real memory</td>
<td>pi_prm</td>
<td>ps -o rssize</td>
</tr>
<tr>
<td>rss</td>
<td>Resident set size</td>
<td>pi_trss + pi_drss</td>
<td>&quot;ps gvw&quot; SIZE column</td>
</tr>
<tr>
<td>size</td>
<td>Size of process (code + data)</td>
<td>pi_dvm</td>
<td>&quot;ps gvw&quot; TSIZ column</td>
</tr>
<tr>
<td>dsize</td>
<td>Data size</td>
<td>pi_dsize</td>
<td></td>
</tr>
<tr>
<td>tsize</td>
<td>Text (code) size</td>
<td>pi_tsize</td>
<td></td>
</tr>
<tr>
<td>sdsise</td>
<td>Data size from shared library</td>
<td>pi_sdsise</td>
<td></td>
</tr>
</tbody>
</table>

\(^6\) According to specification these are voltages on chip pins and generally speaking may need scaling.
### FreeBSD

See values supported for ‘memtype’ parameter on FreeBSD in the table.

<table>
<thead>
<tr>
<th>Supported value</th>
<th>Description</th>
<th>Source in procentry64 structure</th>
<th>Tries to be compatible with</th>
</tr>
</thead>
<tbody>
<tr>
<td>drss</td>
<td>Data resident set size</td>
<td>pi_drss</td>
<td></td>
</tr>
<tr>
<td>trss</td>
<td>Text resident set size</td>
<td>pi_trss</td>
<td></td>
</tr>
</tbody>
</table>

### Linux

See values supported for ‘memtype’ parameter on Linux in the table.

<table>
<thead>
<tr>
<th>Supported value</th>
<th>Description</th>
<th>Source in /proc/&lt;pid&gt;/status file</th>
</tr>
</thead>
<tbody>
<tr>
<td>vsize ¹</td>
<td>Virtual memory size</td>
<td>VmSize</td>
</tr>
<tr>
<td>pmem</td>
<td>Percentage of real memory</td>
<td>(VmRSS/total_memory) * 100</td>
</tr>
<tr>
<td>rss</td>
<td>Resident set size</td>
<td>VmRSS</td>
</tr>
<tr>
<td>data</td>
<td>Size of data segment</td>
<td>VmData</td>
</tr>
<tr>
<td>exe</td>
<td>Size of code segment</td>
<td>VmExe</td>
</tr>
<tr>
<td>hwm</td>
<td>Peak resident set size</td>
<td>VmHWM</td>
</tr>
<tr>
<td>lck</td>
<td>Size of locked memory</td>
<td>VmLck</td>
</tr>
<tr>
<td>lib</td>
<td>Size of shared libraries</td>
<td>VmLib</td>
</tr>
<tr>
<td>peak</td>
<td>Peak virtual memory size</td>
<td>VmPeak</td>
</tr>
<tr>
<td>pin</td>
<td>Size of pinned pages</td>
<td>VmPin</td>
</tr>
<tr>
<td>pte</td>
<td>Size of page table entries</td>
<td>VmPTE</td>
</tr>
<tr>
<td>size</td>
<td>Size of process code + data + stack segments</td>
<td>VmExe + VmData + VmStk</td>
</tr>
<tr>
<td>stk</td>
<td>Size of stack segment</td>
<td>VmStk</td>
</tr>
<tr>
<td>swap</td>
<td>Size of swap space used</td>
<td>VmSwap</td>
</tr>
</tbody>
</table>

Notes for Linux:

1. Not all ‘memtype’ values are supported by older Linux kernels. For example, Linux 2.4 kernels do not support hwm, pin, peak, pte and swap values.

2. We have noticed that self-monitoring of the Zabbix agent active check process with `proc.mem[... , ..., , , , , , data]` shows a value that is 4 kB larger than reported by `VmData` line in the agent’s `/proc/<pid>/status` file. At the time of self-measurement the agent’s data segment increases by 4 kB and then returns to the previous size.

### Solaris

See values supported for ‘memtype’ parameter on Solaris in the table.
<table>
<thead>
<tr>
<th>Supported value</th>
<th>Description</th>
<th>Source in psinfo structure</th>
<th>Tries to be compatible with</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>vsize</strong></td>
<td>Size of process image</td>
<td>pr_size</td>
<td>ps -O vsz</td>
</tr>
<tr>
<td>pmem</td>
<td>Percentage of real memory</td>
<td>pr_pctmem</td>
<td>ps -O pmem</td>
</tr>
<tr>
<td>rss</td>
<td>Resident set size</td>
<td>pr_rssize</td>
<td>ps -O rss</td>
</tr>
</tbody>
</table>

It may be underestimated - see rss description in "man ps".

Footnotes

1 Default value.

### 9 Notes on selecting processes in proc.mem and proc.num items

Processes modifying their commandline

Some programs use modifying their commandline as a method for displaying their current activity. A user can see the activity by running `ps` and `top` commands. Examples of such programs include PostgreSQL, Sendmail, Zabbix.

Let's see an example from Linux. Let's assume we want to monitor a number of Zabbix agent processes.

```
$ ps -fu zabbix
```

Selecting processes by name and user does the job:

```
$ zabbix_get -s localhost -k 'proc.num[zabbix_agentd,zabbix]'
```

Now let's rename `zabbix_agentd` executable to `zabbix_agentd_30` and restart it.

```
$ ps -fu zabbix
```

Now selecting processes by name and user produces an incorrect result:

```
$ zabbix_get -s localhost -k 'proc.num[zabbix_agentd_30,zabbix]'
```

Why a simple renaming of executable to a longer name lead to quite different result?

Zabbix agent starts with checking the process name. `/proc/<pid>/status` file is opened and the line `Name` is checked. In our case the `Name` lines are:

```
$ grep Name /proc/{6715,6716,6717,6718,6719,6720}/status
```

```bash
/proc/6715/status:Name: zabbix_agentd_3
/proc/6716/status:Name: zabbix_agentd_3
/proc/6717/status:Name: zabbix_agentd_3
/proc/6718/status:Name: zabbix_agentd_3
/proc/6719/status:Name: zabbix_agentd_3
/proc/6720/status:Name: zabbix_agentd_3
```

Now let's rename `zabbix_agentd` executable to `zabbix_agentd_30` and restart it.

```
$ ps -fu zabbix
```

Now selecting processes by name and user produces an incorrect result:

```
$ zabbix_get -s localhost -k 'proc.num[zabbix_agentd_30,zabbix]'
```

Why a simple renaming of executable to a longer name lead to quite different result?

Zabbix agent starts with checking the process name. `/proc/<pid>/status` file is opened and the line `Name` is checked. In our case the `Name` lines are:

```
$ grep Name /proc/{6715,6716,6717,6718,6719,6720}/status
```

```bash
/proc/6715/status:Name: zabbix_agentd_3
/proc/6716/status:Name: zabbix_agentd_3
/proc/6717/status:Name: zabbix_agentd_3
/proc/6718/status:Name: zabbix_agentd_3
/proc/6719/status:Name: zabbix_agentd_3
/proc/6720/status:Name: zabbix_agentd_3
```

1476
The process name in status file is truncated to 15 characters. A similar result can be seen with ps command:

```bash
$ ps -u zabbix
    PID  TTY         TIME CMD
...
   6715 ?  00:00:00 zabbix_agentd_3
   6716 ?  00:00:01 zabbix_agentd_3
   6717 ?  00:00:00 zabbix_agentd_3
   6718 ?  00:00:00 zabbix_agentd_3
   6719 ?  00:00:00 zabbix_agentd_3
   6720 ?  00:00:00 zabbix_agentd_3
...
```

Obviously, that is not equal to our proc.num[] name parameter value zabbix_agentd_30. Having failed to match the process name from status file the Zabbix agent turns to /proc/<pid>/cmdline file.

How the agent sees the "cmdline" file can be illustrated with running a command:

```bash
$ for i in 6715 6716 6717 6718 6719 6720; do cat /proc/$i/cmdline | awk '{gsub(/\x0/,"<NUL>"); print}'; done
/sbin/zabbix_agentd_30: collector [idle 1 sec]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL>... 
/sbin/zabbix_agentd_30: listener #1 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL>... 
/sbin/zabbix_agentd_30: listener #2 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL>... 
/sbin/zabbix_agentd_30: listener #3 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL>... 
/sbin/zabbix_agentd_30: active checks #1 [idle 1 sec]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL>... 
```

/proc/<pid>/cmdline files in our case contain invisible, non-printable null bytes, used to terminate strings in C language. The null bytes are shown as "<NUL>" in this example.

Zabbix agent checks "cmdline" for the main process and takes a zabbix_agentd_30, which matches our name parameter value zabbix_agentd_30. So, the main process is counted by item proc.num[zabbix_agentd_30,zabbix].

When checking the next process, the agent takes zabbix_agentd_30: collector [idle 1 sec] from the cmdline file and it does not meet our name parameter zabbix_agentd_30. So, only the main process which does not modify its commandline, gets counted. Other agent processes modify their command line and are ignored.

This example shows that the name parameter cannot be used in proc.mem[] and proc.num[] for selecting processes in this case.

For proc.get[] item, when Zabbix agent checks "cmdline" for the process name, it will only use part of the name starting from the last slash and until the first space or colon sign. Process name received from cmdline file will only be used if its beginning completely matches the shortened process name in the status file. The algorithm is the same for both process name in the filter and in the JSON output.

Using cmdline parameter with a proper regular expression produces a correct result:

```bash
$ zabbix_get -s localhost -k 'proc.num[,zabbix,,zabbix_agentd_30[ :]]'
6
```

Be careful when using proc.get[], proc.mem[] and proc.num[] items for monitoring programs which modify their command lines.

Before putting name and cmdline parameters into proc.get[], proc.mem[] and proc.num[] items, you may want to test the parameters using proc.num[] item and ps command.

Linux kernel threads

Threads cannot be selected with cmdline parameter in proc.get[], proc.mem[] and proc.num[] items.

Let's take as an example one of kernel threads:

```bash
$ ps -ef | grep kthreadd
root  2  0  0 09:33 ? 00:00:00 [kthreadd]
```

It can be selected with process name parameter:

```bash
$ zabbix_get -s localhost -k 'proc.num[kthreadd,root]'
1
```
But selection by process `cmdline` parameter does not work:

```
$ zabbix_get -s localhost -k 'proc.num[,root,,kthreadd]' 
0
```

The reason is that Zabbix agent takes the regular expression specified in `cmdline` parameter and applies it to contents of process `/proc/<pid>/cmdline`. For kernel threads their `/proc/<pid>/cmdline` files are empty. So, `cmdline` parameter never matches.

Counting of threads in `proc.mem[]` and `proc.num[]` items

Linux kernel threads are counted by `proc.num[]` item but do not report memory in `proc.mem[]` item. For example:

```
$ ps -ef | grep kthread
root  2  0 0 09:51 ? 00:00:00 [kthreadd]

$ zabbix_get -s localhost -k 'proc.num[kthreadd]'
1

$ zabbix_get -s localhost -k 'proc.mem[kthreadd]'
ZBX_NOTSUPPORTED: Cannot get amount of "VmSize" memory.
```

But what happens if there is a user process with the same name as a kernel thread? Then it could look like this:

```
$ ps -ef | grep kthread
root  2  0 0 09:51 ? 00:00:00 [kthreadd]
  zabbix  9611  6133  0 17:58 pts/1 00:00:00 ./kthreadd

$ zabbix_get -s localhost -k 'proc.num[kthreadd]'
2

$ zabbix_get -s localhost -k 'proc.mem[kthreadd]'
4157440
```

`proc.num[]` counted both the kernel thread and the user process. `proc.mem[]` reports memory for the user process only and counts the kernel thread memory as if it was 0. This is different from the case above when ZBX_NOTSUPPORTED was reported.

Be careful when using `proc.mem[]` and `proc.num[]` items if the program name happens to match one of the thread.

Before putting parameters into `proc.mem[]` and `proc.num[]` items, you may want to test the parameters using `proc.num[]` item and `ps` command.

## 10 Implementation details of net.tcp.service and net.udp.service checks

Implementation of net.tcp.service and net.udp.service checks is detailed on this page for various services specified in the service parameter.

The `net.tcp.service` parameters

<table>
<thead>
<tr>
<th>Service</th>
<th>Description and Default Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftp</td>
<td>Creates a TCP connection and expects the first 4 characters of the response to be “220 “, then sends “QUIT\n”. Default port 21 is used if not specified.</td>
</tr>
<tr>
<td>http</td>
<td>Creates a TCP connection without expecting and sending anything. Default port 80 is used if not specified.</td>
</tr>
<tr>
<td>https</td>
<td>Uses (and only works with) libcurl, does not verify the authenticity of the certificate, does not verify the host name in the SSL certificate, only fetches the response header (HEAD request). Default port 443 is used if not specified.</td>
</tr>
<tr>
<td>imap</td>
<td>Creates a TCP connection and expects the first 4 characters of the response to be &quot;* OK&quot;, then sends &quot;a1 LOGOUT\n&quot;. Default port 143 is used if not specified.</td>
</tr>
<tr>
<td>ldap</td>
<td>Opens a connection to an LDAP server and performs an LDAP search operation with filter set to (objectClass=__). Expects successful retrieval of the first attribute of the first entry. Default port 389 is used if not specified.</td>
</tr>
</tbody>
</table>
**nttp**

Creates a TCP connection and expects the first 3 characters of the response to be “200” or “201”, then sends “QUIT\r\n”. Default port 119 is used if not specified.

**pop**

Creates a TCP connection and expects the first 3 characters of the response to be “+OK”, then sends “QUIT\r\n”. Default port 110 is used if not specified.

**smtp**

Creates a TCP connection and expects the first 3 characters of the response to be “220”, followed by a space, the line ending or a dash. The lines containing a dash belong to a multi-line response and the response will be re-read until a line without the dash is received. Then sends “QUIT\r\n”. Default port 25 is used if not specified.

**ssh**

Creates a TCP connection. If the connection has been established, both sides exchange an identification string (SSH-major.minor-XXXX), where major and minor are protocol versions and XXXX is a string. Zabbix checks if the string matching the specification is found and then sends back the string “SSH-major.minor-zabbix_agent\r\n” or “0\n” on mismatch. Default port 22 is used if not specified.

**tcp**

Creates a TCP connection without expecting and sending anything. Unlike the other checks requires the port parameter to be specified.

**telnet**

Creates a TCP connection and expects a login prompt (‘:’ at the end). Default port 23 is used if not specified.

**n tp**

Sends an SNTP packet over UDP and validates the response according to [RFC 4330, section 5](https://tools.ietf.org/html/rfc4330). Default port 123 is used if not specified.

### 11 proc.get parameters

#### Overview

The item `proc.get[<name>,<user>,<cmdline>,<mode>]` is supported on Linux, Windows, FreeBSD, OpenBSD, and NetBSD.

List of process parameters returned by the item varies depending on the operating system and ‘mode’ argument value.

#### Linux

The following process parameters are returned on Linux for each mode:

<table>
<thead>
<tr>
<th>mode=process</th>
<th>mode=thread</th>
<th>mode=summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid: PID</td>
<td>pid: PID</td>
<td>name: process name</td>
</tr>
<tr>
<td>ppid: parent PID</td>
<td>ppid: parent PID</td>
<td>processes: number of processes</td>
</tr>
<tr>
<td>name: process name</td>
<td>name: process name</td>
<td>vsize: virtual memory size</td>
</tr>
<tr>
<td>cmdline: command with arguments</td>
<td>user: user (real) the process runs under</td>
<td>pmem: percentage of real memory</td>
</tr>
<tr>
<td>user: user (real) the process runs under</td>
<td>group: group (real) the process runs under</td>
<td>rss: resident set size</td>
</tr>
<tr>
<td>group: group (real) the process runs under</td>
<td>uid: user ID</td>
<td>data: size of data segment</td>
</tr>
<tr>
<td>uid: user ID</td>
<td>gid: ID of the group the process runs under</td>
<td>exe: size of code segment</td>
</tr>
<tr>
<td>gid: ID of the group the process runs under</td>
<td>tid: thread ID</td>
<td>lib: size of shared libraries</td>
</tr>
<tr>
<td>vszsize: virtual memory size</td>
<td>tname: thread name</td>
<td>lck: size of locked memory</td>
</tr>
<tr>
<td>pmem: percentage of real memory</td>
<td>cputime_user: total CPU seconds (user)</td>
<td>pin: size of pinned pages</td>
</tr>
<tr>
<td>rss: resident set size</td>
<td>cputime_system: total CPU seconds (system)</td>
<td>pte: size of page table entries</td>
</tr>
</tbody>
</table>
BSD-based OS

The following process parameters are returned on FreeBSD, OpenBSD, and NetBSD for each mode:
### Windows

The following process parameters are returned on Windows for each mode:

<table>
<thead>
<tr>
<th>mode=process</th>
<th>mode=thread</th>
<th>mode=summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid: PID</td>
<td>pid: PID</td>
<td>name: process name</td>
</tr>
<tr>
<td>ppid: parent PID</td>
<td>ppid: parent PID</td>
<td>processes: number of processes</td>
</tr>
<tr>
<td>name: process name</td>
<td>name: process name</td>
<td>vmsize: virtual memory size</td>
</tr>
<tr>
<td>user: user the process runs under</td>
<td>user: user the process runs under</td>
<td>wkset: size of process working set</td>
</tr>
<tr>
<td>sid: user SID</td>
<td>sid: user SID</td>
<td>cputime_user: total CPU seconds (user)</td>
</tr>
<tr>
<td>vmsize: virtual memory size</td>
<td>tid: thread ID</td>
<td>cputime_system: total CPU seconds (system)</td>
</tr>
<tr>
<td>wkset: size of process working set</td>
<td></td>
<td>threads: number of threads</td>
</tr>
<tr>
<td>cputime_user: total CPU seconds (user)</td>
<td></td>
<td>page_faults: number of page faults</td>
</tr>
<tr>
<td>cputime_system: total CPU seconds (system)</td>
<td></td>
<td>handles: number of handles</td>
</tr>
<tr>
<td>threads: number of threads</td>
<td></td>
<td>io_read_b: IO bytes read</td>
</tr>
<tr>
<td>page_faults: number of page faults</td>
<td></td>
<td>io_write_b: IO bytes written</td>
</tr>
<tr>
<td>handles: number of handles</td>
<td></td>
<td>io_read_op: IO read operations</td>
</tr>
<tr>
<td>io_read_b: IO bytes read</td>
<td></td>
<td>io_write_op: IO write operations</td>
</tr>
<tr>
<td>io_write_b: IO bytes written</td>
<td></td>
<td>io_other_b: IO bytes transferred, other than reads and writes</td>
</tr>
<tr>
<td>io_read_op: IO read operations</td>
<td></td>
<td>io_other_op: IO operations, other than reads and writes</td>
</tr>
<tr>
<td>io_write_op: IO write operations</td>
<td></td>
<td>io_other_op: IO operations, other than reads and writes</td>
</tr>
<tr>
<td>io_other_b: IO bytes transferred, other than reads and writes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>io_other_op: IO operations, other than reads and writes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 12 Unreachable/unavailable host interface settings

Overview
Several configuration parameters define how Zabbix server should behave when an agent check (Zabbix, SNMP, IPMI, JMX) fails and a host interface becomes unreachable.

Unreachable interface

A host interface is treated as unreachable after a failed check (network error, timeout) by Zabbix, SNMP, IPMI or JMX agents. Note that Zabbix agent active checks do not influence interface availability in any way.

From that moment **UnreachableDelay** defines how often an interface is rechecked using one of the items (including LLD rules) in this unreachability situation and such rechecks will be performed already by unreachable pollers (or IPMI pollers for IPMI checks).

By default it is 15 seconds before the next check.

In the Zabbix server log unreachability is indicated by messages like these:

Zabbix agent item "system.cpu.load[percpu,avg1]" on host "New host" failed: first network error, wait for 15 seconds

Zabbix agent item "system.cpu.load[percpu,avg15]" on host "New host" failed: another network error, wait for 15 seconds

Note that the exact item that failed is indicated and the item type (Zabbix agent).

The **Timeout** parameter will also affect how early an interface is rechecked during unreachability. If the Timeout is 20 seconds and UnreachableDelay 30 seconds, the next check will be in 50 seconds after the first attempt.

The **UnreachablePeriod** parameter defines how long the unreachability period is in total. By default UnreachablePeriod is 45 seconds. UnreachablePeriod should be several times bigger than UnreachableDelay, so that an interface is rechecked more than once before an interface becomes unavailable.

Switching interface back to available

When the unreachability period is over, the interface is polled again, decreasing priority for item that turned the interface into unreachable state. If the unreachable interface reappears, the monitoring returns to normal automatically:

resuming Zabbix agent checks on host "New host": connection restored

Once interface becomes available, the host does not poll all its items immediately for two reasons:

- It might overload the host.
- The interface restore time is not always matching planned item polling schedule time.

So, after the interface becomes available, items are not polled immediately, but they are getting rescheduled to their next polling round.

Unavailable interface

After the UnreachablePeriod ends and the interface has not reappeared, the interface is treated as unavailable.

In the server log it is indicated by messages like these:

temporarily disabling Zabbix agent checks on host "New host": interface unavailable

and in the frontend the host availability icon goes from green/gray to yellow/red (the unreachable interface details can be seen in the hint box that is displayed when a mouse is positioned on the host availability icon):

![Zabbix Interface Status](#)

The **UnavailableDelay** parameter defines how often an interface is checked during interface unavailability.

By default it is 60 seconds (so in this case "temporarily disabling", from the log message above, will mean disabling checks for one minute).

When the connection to the interface is restored, the monitoring returns to normal automatically, too:

enabling Zabbix agent checks on host "New host": interface became available
13 Remote monitoring of Zabbix stats

Overview
It is possible to make some internal metrics of Zabbix server and proxy accessible remotely by another Zabbix instance or a third party tool. This can be useful so that supporters/service providers can monitor their client Zabbix servers/proxies remotely or, in organizations where Zabbix is not the main monitoring tool, that Zabbix internal metrics can be monitored by a third party system in an umbrella-monitoring setup.

Zabbix internal stats are exposed to a configurable set of addresses listed in the new `StatsAllowedIP` server/proxy parameter. Requests will be accepted only from these addresses.

**Items**
To configure querying of internal stats on another Zabbix instance, you may use two items:

- `zabbix[stats,<ip>,<port>]` internal item - for direct remote queries of Zabbix server/proxy. `<ip>` and `<port>` are used to identify the target instance.
- `zabbix.stats[<ip>,<port>]` agent item - for agent-based remote queries of Zabbix server/proxy. `<ip>` and `<port>` are used to identify the target instance.

See also: Internal items, Zabbix agent items

The following diagram illustrates the use of either item depending on the context.

- Server → external Zabbix instance (`zabbix[stats,<ip>,<port>]`)
Exposed metrics

The stats items gather the statistics in bulk and return a JSON, which is the basis for dependent items to get their data from. The following **internal metrics** are returned by either of the two items:

- `zabbix[boottime]`
- `zabbix[hosts]`
- `zabbix[items]`
- `zabbix[items_unsupported]`
- `zabbix[preprocessing_queue]` (server only)
- `zabbix[process,<type>,<mode>,<state>]` (only process type based statistics)
- `zabbix[rcache,<cache>,<mode>]`
- `zabbix[requiredperformance]`
- `zabbix[triggers]` (server only)
- `zabbix[uptime]`
- `zabbix[vcache,buffer,<mode>]` (server only)
- `zabbix[vcache,cache,<parameter>]`
- `zabbix[version]`
- `zabbix[vmware,buffer,<mode>]`
- `zabbix[wcache,<cache>,<mode>]` ('trends' cache type server only)

Templates

Templates are available for remote monitoring of Zabbix server or proxy internal metrics from an external instance:

- Remote Zabbix server
- Remote Zabbix proxy

Note that in order to use a template for remote monitoring of multiple external instances, a separate host is required for each external instance monitoring.

Trapper process

Receiving internal metric requests from an external Zabbix instance is handled by the trapper process that validates the request, gathers the metrics, creates the JSON data buffer and sends the prepared JSON back, for example, from server:

```json
{
  "response": "success",
  "data": {
    "boottime": N,
    "uptime": N,
    "hosts": N,
    "items": N,
    "items_unsupported": N,
    "preprocessing_queue": N,
    "process": {
      "alert manager": {
        "busy": {
          "avg": N,
          "max": N,
          "min": N
        },
        "idle": {
          "avg": N,
          "max": N,
          "min": N
        },
        "count": N
      },
      "queue": N
    }
  }
}
```
"rcache": {  
"total": N,
"free": N,
"pfree": N,
"used": N,
"pused": N
},
"requiredperformance": N,
"triggers": N,
"uptime": N,
"vcache": {  
"buffer": {  
"total": N,
"free": N,
"pfree": N,
"used": N,
"pused": N
  },
"cache": {  
"requests": N,
"hits": N,
"misses": N,
"mode": N
  }
},
"vmware": {  
"total": N,
"free": N,
"pfree": N,
"used": N,
"pused": N
},
"version": "N",
"wcache": {  
"values": {  
"all": N,
"float": N,
"uint": N,
"str": N,
"log": N,
"text": N,
"not supported": N
  },
"history": {  
"pfree": N,
"free": N,
"total": N,
"used": N,
"pused": N
  },
"index": {  
"pfree": N,
"free": N,
"total": N,
"used": N,
"pused": N
  },
"trend": {  
"pfree": N,
"free": N,
"total": N,
"used": N,
Internal queue items

There are also another two items specifically allowing to remote query internal queue stats on another Zabbix instance:

- `zabbix[stats,<ip>,<port>,queue,<from>,<to>]` internal item - for direct internal queue queries to remote Zabbix server/proxy
- `zabbix.stats[<ip>,<port>,queue,<from>,<to>]` agent item - for agent-based internal queue queries to remote Zabbix server/proxy

See also: [Internal items](#), [Zabbix agent items](#)

### 14 Configuring Kerberos with Zabbix

**Overview**

Kerberos authentication can be used in web monitoring and HTTP items in Zabbix since version 4.4.0.

This section describes an example of configuring Kerberos with Zabbix server to perform web monitoring of `www.example.com` with user ‘zabbix’.

**Steps**

**Step 1**

Install Kerberos package.

For Debian/Ubuntu:

```bash
apt install krb5-user
```

For RHEL/CentOS:

```bash
yum install krb5-workstation
```

**Step 2**

Configure Kerberos configuration file (see MIT documentation for details)

```bash
cat /etc/krb5.conf
```

*[libdefaults]*

```bash
    default_realm = EXAMPLE.COM
```

### The following krb5.conf variables are only for MIT Kerberos.

```bash
    kdc_timesync = 1
    ccache_type = 4
    forwardable = true
    proxiable = true
```

*[realms]*

```bash
    EXAMPLE.COM = {
    }
```

*[domain_realm]*

```bash
    .example.com=EXAMPLE.COM
    example.com=EXAMPLE.COM
```

**Step 3**

Create a Kerberos ticket for user zabbix. Run the following command as user zabbix:

```bash
kinit zabbix
```

It is important to run the above command as user zabbix. If you run it as root the authentication will not work.

**Step 4**


Create a web scenario or HTTP agent item with Kerberos authentication type.

Optionally can be tested with the following curl command:

curl -v --negotiate -u : http://example.com

Note that for lengthy web monitoring it is necessary to take care of renewing the Kerberos ticket. Default time of ticket expiration is 10h.

### 15 modbus.get parameters

**Overview**

The table below presents details of the `modbus.get()` item parameters.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Defaults</th>
<th>Example</th>
</tr>
</thead>
</table>
| endpoint  | Protocol and address of the endpoint, defined as `protocol://connection_string` | protocol: none | tcp://192.168.6.1:511  
tcp://192.168.6.2  
tcp://[:1]:511 |
|           | Possible protocol values: rtu, ascii (Agent 2 only), tcp | rtu/ascii protocol: | tcp://:1  
tcp://localhost:511 |
|           | Connection string format: | port_name: none | tcp://localhost  
rtu://COM1:9600:8n  
ascii://COM2:1200:7o2 |
|           | with tcp - address:port | speed: 115200 | tcp://localhost  
tcp://localhost  
tcp://localhost  
tcp://localhost  
tcp://localhost |
|           | with serial line: rtu, ascii -port_name:speed:params | params: 8n1 | tcp://localhost  
tcp://localhost  
tcp://localhost  
tcp://localhost  
tcp://localhost |
|           | where 'speed' - 1200, 9600 etc | | tcp://localhost  
tcp://localhost  
tcp://localhost  
tcp://localhost  
tcp://localhost |
|           | 'params' - data bits (5,6,7 or 8), parity (n,e or o for none/even/odd), stop bits (1 or 2) | | tcp://localhost  
tcp://localhost  
tcp://localhost  
tcp://localhost  
tcp://localhost |
| slave id  | Modbus address of the device it is intended for (1 to 247), see MODBUS Messaging Implementation Guide (page 23) | serial: 1 | 2  
tcp: 255 (0xFF) |
| function  | Empty or value of a supported function: | empty | 3  
1 - Read Coil,  
2 - Read Discrete Input,  
3 - Read Holding Registers,  
4 - Read Input Registers |
| address   | Address of the first registry, coil or input. | empty function: | 9999  
00001 |
|           | If ‘function’ is empty, then ‘address’ should be in range for: | non-empty function: | 0  
Coil - 00001 - 09999  
Discrete input - 10001 - 19999  
Input register - 30001 - 39999  
Holding register - 40001 - 49999 |
|           | If ‘function’ is not empty, the ‘address’ field will be from 0 till 65535 and used without modification (PDU) | | |
| count     | Count of sequenced ‘type’ which will be read from device, where: | 1 | 2  
for Coil or Discrete input the ‘type’ = 1 bit  
for other cases: (count*type)/2 = real count of registers for reading  
If ‘offset’ is not 0, the value will be added to ‘real count’  
Acceptable range for ‘real count’ is 1:65535 |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Defaults</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Data type:</td>
<td>bit</td>
<td>uint16</td>
</tr>
<tr>
<td></td>
<td>for Read Coil and Read Discrete Input - bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>for Read Holding Registers and Read Input Registers:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>int8 - 8bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>uint8 - 8bit (unsigned)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>int16 - 16bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>uint16 - 16bit (unsigned)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>int32 - 32bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>uint32 - 32bit (unsigned)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>float - 32bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>uint64 - 64bit (unsigned)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>double - 64bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>endianness</td>
<td>Endianness type:</td>
<td>be</td>
<td>le</td>
</tr>
<tr>
<td></td>
<td>be - Big Endian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>le - Little Endian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mbe - Mid-Big Endian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mle - Mid-Little Endian</td>
<td></td>
<td></td>
</tr>
<tr>
<td>offset</td>
<td>Number of registers, starting from 'address', the result of which will be discarded.</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>The size of each register is 16bit (needed to support equipment that does not support random read access).</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6 Supported functions

Click on the respective function group to see more details.

<table>
<thead>
<tr>
<th>Function group</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregate functions</td>
<td>avg, bucket_percentile, count, histogram_quantile, item_count, kurtosis, mad, max, min, skewness, stddevpop, stdevsamp, sum, sumofquares, varpop, varsamp</td>
</tr>
<tr>
<td>Bitwise functions</td>
<td>bitand, bitshift, bitnot, bitor, bitwise, bitxor</td>
</tr>
<tr>
<td>Date and time functions</td>
<td>date, dayofmonth, dayofweek, now, time</td>
</tr>
<tr>
<td>History functions</td>
<td>change, changecount, count, countunique, find, first, fuzzytime, last, logeventid, logseverity, logsource, monodec, monoinc, nodata, percentile, rate</td>
</tr>
<tr>
<td>Trend functions</td>
<td>baselinedev, baselinewma, trendavg, trendcount, trendmax, trendmin, trendstl, trendsum</td>
</tr>
<tr>
<td>Mathematical functions</td>
<td>abs, acos, asin, atan, atan2, avg, cbt, ceil, cos, cosh, cot, degrees, e, exp, expm1, floor, log, log10, max, min, mod, pi, power, radians, rand, round, signum, sin, sinh, sqrt, sum, tan, truncate</td>
</tr>
<tr>
<td>Operator functions</td>
<td>between, in</td>
</tr>
<tr>
<td>Prediction functions</td>
<td>forecast, timeleft</td>
</tr>
<tr>
<td>String functions</td>
<td>ascii, bitlength, bytelength, char, concat, insert, left, length, ltrim, mid, repeat, replace, right, rtrim, trim</td>
</tr>
</tbody>
</table>

These functions are supported in trigger expressions and calculated items.

1 Aggregate functions

Except where stated otherwise, all functions listed here are supported in:

- Trigger expressions
• Calculated items

Aggregate functions can work with either:

• history of items, for example, `min(/host/key,1h)`
• `foreach functions` as the only parameter, for example, `min(last_foreach(//*/key))`

Some general notes on function parameters:

• Function parameters are separated by a comma
• Optional function parameters (or parameter parts) are indicated by `< >`
• Function-specific parameters are described with each function
• `/host/key` and `(sec|#num)<:time shift>` parameters must never be quoted

Common parameters

• `/host/key` is a common mandatory first parameter for the functions referencing the host item history
• `(sec|#num)<:time shift>` is a common second parameter for the functions referencing the host item history, where:
  - `sec` - maximum evaluation period in seconds (time suffixes can be used), or
  - `#num` - maximum evaluation range in latest collected values (if preceded by a hash mark)
  - `time shift` (optional) allows to move the evaluation point back in time. See more details on specifying time shift.

Aggregate functions

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>Description</th>
<th>Function-specific parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>avg</strong></td>
<td>/host/key,(sec</td>
<td>#num)&lt;:time shift&gt;</td>
<td>See common parameters.</td>
</tr>
<tr>
<td></td>
<td>Average value of an item within the defined evaluation period.</td>
<td></td>
<td>Examples:</td>
</tr>
<tr>
<td></td>
<td>=&gt; avg(/host/key,1h) → average value for the last hour until now</td>
<td></td>
<td>=&gt; avg(/host/key,1h:now-1d) → average value for an hour from 25 hours ago to 24 hours ago from now</td>
</tr>
<tr>
<td></td>
<td>=&gt; avg(/host/key,#5) → average value of the five latest values</td>
<td></td>
<td>=&gt; avg(/host/key,#5:now-1d) → average value of the five latest values excluding the values received in the last 24 hours</td>
</tr>
<tr>
<td></td>
<td>=&gt; avg(/host/key,#5:now-1d) → average value of the five latest values excluding the values received in the last 24 hours</td>
<td></td>
<td>Time shift is useful when there is a need to compare the current average value with the average value some time ago.</td>
</tr>
<tr>
<td><strong>bucket_percentile</strong></td>
<td>item filter,time period,percentage</td>
<td>item filter - see item filter, time period - see time period, percentage - percentage (0-100)</td>
<td>Supported only in calculated items.</td>
</tr>
<tr>
<td></td>
<td>Calculates the percentile from the buckets of a histogram.</td>
<td></td>
<td>This function is an alias for histogram_quantile(percentage/100, bucket_rate_foreach(item filter, time period, 1))</td>
</tr>
<tr>
<td><strong>count</strong></td>
<td>func_foreach(item filter,&lt;time period&gt;)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### FUNCTION

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
<th>Supported Value Types</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count of values in an array returned by a foreach function.</strong></td>
<td><strong>func_foreach</strong> - foreach function for which the number of returned values should be counted (with supported arguments). See foreach functions for details.</td>
<td>int</td>
<td>=&gt; count(max_foreach(/net.if.in[*],1h)) → number of net.if.in items that received data in the last hour until now</td>
<td>Note that using <strong>count()</strong> with a history-related foreach function (max_foreach, avg_foreach, etc.) may lead to performance implications, whereas using <strong>exists_foreach()</strong>, which works only with configuration data, will not have such effect.</td>
</tr>
<tr>
<td><strong>histogram_quantile</strong> (quantile,bucket1,value1,bucket2,value2,...)</td>
<td>Calculates the $\phi$-quantile from the buckets of a histogram.</td>
<td>- $0 \leq \phi \leq 1$&lt;br&gt;- <em>bucketN, valueN</em> - manually entered pairs ($\geq 2$) of parameters or response of bucket_rate_foreach</td>
<td><strong>Supported only in calculated items.</strong></td>
<td>Functionally corresponds to ‘histogram_quantile’ of PromQL. Returns -1 if values of the last ‘Infinity’ bucket (“+inf”) are equal to 0. Example: =&gt; histogram_quantile(0.75,1.0,last(/host/rate_bucket[1.0]),&quot;+Inf&quot;,last(/host/rate_bucket[Inf])</td>
</tr>
<tr>
<td><strong>item_count</strong> (item filter)</td>
<td>Count of existing items in configuration that match filter criteria.</td>
<td><strong>item filter</strong> - criteria for item selection, allows referencing by host group, host, item key, and tags. Wildcards are supported. See item filter for more details.</td>
<td>Supported value type: int</td>
<td>Works as an alias for the count(exists_foreach(item_filter)) function. Example: =&gt; item_count(/agent.ping?[group=&quot;Host group 1&quot;])) → number of hosts with the agent.ping item in the “Host group 1”</td>
</tr>
<tr>
<td><strong>kurtosis</strong> (/host/key,(sec</td>
<td>#num)&lt;:time shift&gt;)</td>
<td>“Tailedness” of the probability distribution in collected values within the defined evaluation period. See also: Kurtosis</td>
<td>Supported value types: float, int</td>
<td>Example: =&gt; kurtosis(/host/key,1h) → kurtosis for the last hour until now</td>
</tr>
<tr>
<td><strong>mad</strong> (/host/key,(sec</td>
<td>#num)&lt;:time shift&gt;)</td>
<td>Median absolute deviation in collected values within the defined evaluation period. See also: Median absolute deviation</td>
<td>Supported value types: float, int</td>
<td>Example: =&gt; mad(/host/key,1h) → median absolute deviation for the last hour until now</td>
</tr>
</tbody>
</table>
**FUNCTION**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Supported value types: float, int</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highest value of an item within the defined evaluation period.</td>
<td>See common parameters.</td>
<td>Example: =&gt; $\text{max}(/\text{host}/\text{key}, 1\text{h}) - \text{min}(/\text{host}/\text{key}, 1\text{h})$ → calculate the difference between the maximum and minimum values within the last hour until now (delta of values)</td>
<td></td>
</tr>
</tbody>
</table>

| min (/host/key,(sec|#num)<:time shift>) | Lowest value of an item within the defined evaluation period. | Supported value types: float, int | Example: => $\text{max}(/\text{host}/\text{key}, 1\text{h}) - \text{min}(/\text{host}/\text{key}, 1\text{h})$ → calculate the difference between the maximum and minimum values within the last hour until now (delta of values) |

| skewness (/host/key,(sec|#num)<:time shift>) | Asymmetry of the probability distribution in collected values within the defined evaluation period. | Supported value types: float, int | Example: => $\text{skewness}(/\text{host}/\text{key}, 1\text{h})$ → skewness for the last hour until now |

| stddevpop (/host/key,(sec|#num)<:time shift>) | Population standard deviation in collected values within the defined evaluation period. | Supported value types: float, int | Example: => $\text{stddevpop}(/\text{host}/\text{key}, 1\text{h})$ → population standard deviation for the last hour until now |

| stddevsamp (/host/key,(sec|#num)<:time shift>) | Sample standard deviation in collected values within the defined evaluation period. | Supported value types: float, int | Example: => $\text{stddevsamp}(/\text{host}/\text{key}, 1\text{h})$ → sample standard deviation for the last hour until now |

| sum (/host/key,(sec|#num)<:time shift>) | Sum of collected values within the defined evaluation period. | Supported value types: float, int | Example: => $\text{sum}(/\text{host}/\text{key}, 1\text{h})$ → sum of values for the last hour until now |

| sumofsquares (/host/key,(sec|#num)<:time shift>) | The sum of squares in collected values within the defined evaluation period. | Supported value types: float, int | Example: => $\text{sumofsquares}(/\text{host}/\text{key}, 1\text{h})$ → sum of squares for the last hour until now |

| varpop (/host/key,(sec|#num)<:time shift>) | | | |

1491


## 1 Foreach functions

### Overview

Foreach functions return aggregate values from the history of multiple items. Foreach functions are used in aggregate calculations.

The following foreach functions are supported:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg_foreach</td>
<td>Returns average values</td>
</tr>
<tr>
<td>bucket_rate_foreach</td>
<td>Returns pairs (bucket upper bound, rate value) suitable for use in the histogram_quantile() function, where &quot;bucket upper bound&quot; is the value of item key parameter defined by the &lt;parameter number&gt; parameter.</td>
</tr>
<tr>
<td>count_foreach</td>
<td>Returns the number of values</td>
</tr>
<tr>
<td>exists_foreach</td>
<td>Returns the number of currently enabled items</td>
</tr>
<tr>
<td>last_foreach</td>
<td>Returns last values</td>
</tr>
<tr>
<td>max_foreach</td>
<td>Returns maximum values</td>
</tr>
<tr>
<td>min_foreach</td>
<td>Returns minimum values</td>
</tr>
<tr>
<td>sum_foreach</td>
<td>Returns the sum of values</td>
</tr>
</tbody>
</table>

Foreach functions return an array of values - one for each item. Items are selected by using filters in the first parameter. See aggregate calculations for more details on using foreach functions.

### Parameters

Foreach functions support two common parameters - item filter and time period:

```
foreach_function(item filter, time period)
```

for example:

```
avg_foreach(*mysql.qps? [group="MySQL Servers"], 5m)
```

Some functions support additional parameters.

**Item filter**

The **first** parameter allows to filter the items we are interested in. A complex filter may be used, referencing the item key, host group and tags, as illustrated by the examples:

<table>
<thead>
<tr>
<th>Syntax example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/host/key[abc.*]</td>
<td>Matches similar items on this host.</td>
</tr>
<tr>
<td>*/key</td>
<td>Matches the same item of any host.</td>
</tr>
<tr>
<td>*/key?[group=&quot;ABC&quot; and tag=&quot;tagname:value&quot;]</td>
<td>Matches the same item of any host from the ABC group having 'tagname:value' tags.</td>
</tr>
</tbody>
</table>
Syntax example
/*/key[a..,c]?(group="ABC" and tag="Tag1") or (group="DEF" and (tag="Tag2" or tag="Tag3:value")))

<table>
<thead>
<tr>
<th>Description</th>
<th>Function-specific parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>*/key[a..,c]?(group=&quot;ABC&quot; and tag=&quot;Tag1&quot;) or (group=&quot;DEF&quot; and (tag=&quot;Tag2&quot; or tag=&quot;Tag3:value&quot;)))</td>
<td>Matches similar items of any host from the ABC or DEF group with the respective tags.</td>
<td></td>
</tr>
</tbody>
</table>

All referenced items must exist and collect data. Only enabled items on enabled hosts are included in the calculations.

If the item key of a referenced item is changed, the filter must be updated manually.

Specifying a parent host group includes the parent group and all nested host groups with their items.

**Time period**

The **second** parameter allows to specify the time period for aggregation. The time period can only be expressed as time, the amount of values (prefixed with #) is not supported.

**Supported unit symbols** can be used in this parameter for convenience, for example ‘5m’ (five minutes) instead of ‘300s’ (300 seconds) or ‘1d’ (one day) instead of ‘86400’ (86400 seconds).

Time period is ignored by the server if passed with the last_foreach function and can thus be omitted:

last_foreach(/*/key?[group="host group"])

Time period is not supported with the exists_foreach function.

**Additional parameters**

A third optional parameter is supported by the bucket_rate_foreach function:

bucket_rate_foreach(item filter, time period,<parameter number>)

where <parameter number> is the position of the “bucket” value in the item key. For example, if the “bucket” value in myItem[aaa,0.2] is ‘0.2’, then its position is 2.

The default value of <parameter number> is ‘1’.

**2 Bitwise functions**

All functions listed here are supported in:

- Trigger expressions
- Calculated items

Some general notes on function parameters:

- Function parameters are separated by a comma
- Expressions are accepted as parameters
- Optional function parameters (or parameter parts) are indicated by < >

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>Description</th>
<th>Function-specific parameters</th>
<th>Comments</th>
</tr>
</thead>
</table>
| bitand   | Value of “bitwise AND” of an item value and mask. | value - value to check, mask (mandatory) - 64-bit unsigned integer (0 - 18446744073709551615) | Supported value types: int

Although the comparison is done in a bitwise manner, all the values must be supplied and are returned in decimal. For example, checking for the 3rd bit is done by comparing to 4, not 100.

Examples:

=> `bitand(last(/host/key),12)`=8 or `bitand(last(/host/key),12)`=4 → 3rd or 4th bit set, but not both at the same time

=> `bitand(last(/host/key),20)`=16 → 3rd bit not set and 5th bit set.

bitlshift (value, bits to shift)
## Function

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Function-specific parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bitwiseshiftleft</code></td>
<td>Bitwise shift left of an item value.</td>
<td>value - value to check</td>
<td>Supported value types: int</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bits to shift (mandatory) -</td>
<td>Although the comparison is done in a bitwise manner, all the values must</td>
</tr>
<tr>
<td></td>
<td></td>
<td>number of bits to shift</td>
<td>be supplied and are returned in decimal. For example, checking for the 3rd</td>
</tr>
<tr>
<td><code>bitnot</code></td>
<td>Value of “bitwise NOT” of an item value.</td>
<td>value - value to check</td>
<td>bit is done by comparing to 4, not 100.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>bitor</code></td>
<td>Value of “bitwise OR” of an item value</td>
<td>value - value to check</td>
<td>Supported value types: int</td>
</tr>
<tr>
<td></td>
<td>and mask</td>
<td>mask (mandatory) - 64-bit</td>
<td>Although the comparison is done in a bitwise manner, all the values must</td>
</tr>
<tr>
<td></td>
<td></td>
<td>unsigned integer (0 - 18446744073709551615)</td>
<td>be supplied and are returned in decimal. For example, checking for the 3rd bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>is done by comparing to 4, not 100.</td>
</tr>
<tr>
<td><code>bitshift</code></td>
<td>Bitwise shift right of an item value.</td>
<td>value - value to check</td>
<td>Supported value types: int</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bits to shift (mandatory) -</td>
<td>Although the comparison is done in a bitwise manner, all the values must</td>
</tr>
<tr>
<td></td>
<td></td>
<td>number of bits to shift</td>
<td>be supplied and are returned in decimal. For example, checking for the 3rd bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>is done by comparing to 4, not 100.</td>
</tr>
<tr>
<td><code>bitxor</code></td>
<td>Value of “bitwise exclusive OR” of an</td>
<td>value - value to check</td>
<td>Supported value types: int</td>
</tr>
<tr>
<td></td>
<td>item value and mask.</td>
<td>mask (mandatory) - 64-bit</td>
<td>Although the comparison is done in a bitwise manner, all the values must</td>
</tr>
<tr>
<td></td>
<td></td>
<td>unsigned integer (0 - 18446744073709551615)</td>
<td>be supplied and are returned in decimal. For example, checking for the 3rd bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>is done by comparing to 4, not 100.</td>
</tr>
</tbody>
</table>

### 3 Date and time functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

Date and time functions cannot be used in the expression alone; at least one non-time-based function referencing the host item must be present in the expression.

---

**FUNCTION**

**Description**

- **date**
  - Current date in YYYYMMDD format.

- **dayofmonth**
  - Day of month in range of 1 to 31.

**Function-specific parameters**

- Example:
  - `=> date()<20220101`

- Example:
  - `=> dayofmonth()=1`
FUNCTION

**dayofweek**
Day of week in range of 1 to 7 (Mon - 1, Sun - 7).
Example: => dayofweek() < 6

**now**
Number of seconds since the Epoch (00:00:00 UTC, January 1, 1970).
Example: => now() < 1640998800

**time**
Current time in HHMMSS format.
Example: => time() > 000000 and time() < 060000

4 History functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

Some general notes on function parameters:

- Function parameters are separated by a comma
- Optional function parameters (or parameter parts) are indicated by < >
- Function-specific parameters are described with each function
- /host/key and (sec|#num)<:time shift> parameters must never be quoted

Common parameters:

- /host/key is a common mandatory first parameter for the functions referencing the host item history
- (sec|#num)<:time shift> is a common second parameter for the functions referencing the host item history, where:
  - sec - maximum evaluation period in seconds (time suffixes can be used), or
  - #num - maximum evaluation range in latest collected values (if preceded by a hash mark)
  - time shift (optional) allows to move the evaluation point back in time. See more details on specifying time shift.

History functions

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>Description</th>
<th>Function-specific parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>change</strong> (/host/key)</td>
<td>The amount of difference between the previous and latest value.</td>
<td>Supported value types: float, int, str, text, log</td>
<td>For strings returns: 0 - values are equal 1 - values differ</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: =&gt; change(/host/key) &gt; 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Numeric difference will be calculated, as seen with these incoming example values ('previous' and 'latest' value = difference): '1' and '5' = +4 '3' and '1' = -2 '0' and '-2.5' = -2.5</td>
</tr>
</tbody>
</table>

**changecount**

 divulged text

See also: abs for comparison
**FUNCTION**

Number of changes between adjacent values within the defined evaluation period.

See common parameters.

**mode** (optional; must be double-quoted)

Supported modes:
- all - count all changes (default)
- dec - count decreases
- inc - count increases

Supported value types: float, int, str, text, log

For non-numeric value types, mode parameter is ignored.

Examples:

```
=> changecount(/host/key, 1w)  → number of value changes for the last week until now
```

```
=> changecount(/host/key,#10,“inc”)  → number of value increases (relative to the adjacent value) among the last 10 values
```

```
=> changecount(/host/key,24h,“dec”)  → number of value decreases (relative to the adjacent value) for the last 24 hours until now
```
FUNCTION

Number of values within the defined evaluation period.

See common parameters.

**operator** (optional; must be double-quoted)

Supported operators:
- `eq` - equal (default)
- `ne` - not equal
- `gt` - greater
- `ge` - greater or equal
- `lt` - less
- `le` - less or equal
- `like` - matches if contains pattern (case-sensitive)
- `bitand` - bitwise AND
- `regexp` - case-sensitive match of the regular expression given in `pattern`
- `iregexp` - case-insensitive match of the regular expression given in `pattern`

**pattern** (optional) - required pattern (string arguments must be double-quoted)

Supported value types: float, integer, string, text, log

Float items match with the precision of 2.22e-16; if database is not upgraded the precision is 0.000001.

With `bitand` as the third parameter, the fourth `pattern` parameter can be specified as two numbers, separated by `/`:

**number_to_compare_with/mask.**

`count()` calculates “bitwise AND” from the value and the mask and compares the result to `number_to_compare_with`. If the result of “bitwise AND” is equal to `number_to_compare_with`, the value is counted.

If `number_to_compare_with` and mask are equal, only the mask need be specified (without `/`).

With `regexp` or `iregexp` as the third parameter, the fourth `pattern` parameter can be an ordinary or global (starting with `@`) regular expression. In case of global regular expressions case sensitivity is inherited from global regular expression settings. For the purpose of regexp matching, float values will always be represented with 4 decimal digits after `.`. Also note that for large numbers difference in decimal (stored in database) and binary (used by Zabbix server) representation may affect the 4th decimal digit.

Examples:

```
=> count(/host/key,10m) → number of values for the last 10 minutes until now
=> count(/host/key,10m,"like","error") → number of values for the last 10 minutes until now that contain 'error'
=> count(/host/key,10m,12) → number of values for the last 10 minutes until now that equal '12'
=> count(/host/key,10m,"gt",12) → number of values for the last 10 minutes until now that are over '12'
=> count(/host/key,10m,"gt",12) → number of values within the last 10 values until now that are over '12'
=> count(/host/key,10m:now-1d,"gt",12) → number of values between 24 hours and 10 minutes and 24 hours ago from now that were over '12'
=> count(/host/key,10m,"bitand","6/7") → number of values for the last 10 minutes until now having '110' (in
```
FUNCTION

countunique
(/host/key,(sec|#num)<:time shift>,<operator>,<pattern>)
### Function

Number of unique values within the defined evaluation period.

See common parameters.

- **operator** (optional; must be double-quoted)

  Supported operators:
  - `eq` - equal (default)
  - `ne` - not equal
  - `gt` - greater
  - `ge` - greater or equal
  - `lt` - less
  - `le` - less or equal
  - `like` - matches if contains pattern (case-sensitive)
  - `bitand` - bitwise AND
  - `regexp` - case-sensitive match of the regular expression given in *pattern*
  - `iregexp` - case-insensitive match of the regular expression given in *pattern*

- **pattern** (optional) - required pattern (string arguments must be double-quoted)

Supported value types: float, integer, string, text, log

- Float items match with the precision of 2.22e-16; if database is not upgraded the precision is 0.000001.
- With `bitand` as the third parameter, the fourth *pattern* parameter can be specified as two numbers, separated by `'/`':
  - `number_to_compare_with/mask` count() calculates "bitwise AND" from the value and the mask and compares the result to `number_to_compare_with`. If the result of "bitwise AND" is equal to `number_to_compare_with`, the value is counted.
  - If `number_to_compare_with` and mask are equal, only the mask need be specified (without `'/`).
- With `regex` or `iregexp` as the third parameter, the fourth *pattern* parameter can be an ordinary or global (starting with `@`) regular expression. In case of global regular expressions case sensitivity is inherited from global regular expression settings. For the purpose of regexp matching, float values will always be represented with 4 decimal digits after `.`. Also note that for large numbers difference in decimal (stored in database) and binary (used by Zabbix server) representation may affect the 4th decimal digit.

Examples:

- `=> countunique(/host/key, 10m) → number of unique values for the last 10 minutes until now`
- `=> countunique(/host/key, 10m, "like", "error") → number of unique values for the last 10 minutes until now that contain 'error'`
- `=> countunique(/host/key, 10m, "gt", 12) → number of unique values for the last 10 minutes until now that are over '12'`
- `=> countunique(/host/key, #10, "gt", 12) → number of unique values within the last 10 values until now that are over '12'`
- `=> countunique(/host/key, 10m:now-1d, "gt", 12) → number of unique values between 24 hours and 10 minutes and 24 hours ago from now that were over '12'`
- `=> countunique(/host/key, 10m, "bitand", "6/7")`
**FUNCTION**

**find** (/host/key, <sec|#num><:time shift>, <operator>, <pattern>)

Find a value match. See common parameters.

- **sec** or **#num** (optional) - defaults to the latest value if not specified
- **operator** (optional; must be double-quoted)
  Supported operators:  
  - eq - equal (default)  
  - ne - not equal  
  - gt - greater  
  - ge - greater or equal  
  - lt - less  
  - le - less or equal  
  - like - value contains the string given in pattern (case-sensitive)  
  - bitand - bitwise AND  
  - regexp - case-sensitive match of the regular expression given in pattern  
  - iregexp - case-insensitive match of the regular expression given in pattern

- **pattern** - required pattern (string arguments must be double-quoted); Perl Compatible Regular Expression (PCRE) regular expression if operator is regexp, iregexp.

Supported value types: float, int, str, text, log

Returns:
- 1 - found  
- 0 - otherwise

If more than one value is processed, ‘1’ is returned if there is at least one matching value.

With regexp or iregexp as the third parameter, the fourth pattern parameter can be an ordinary or global (starting with ‘@’) regular expression. In case of global regular expressions case sensitivity is inherited from global regular expression settings.

Example:

```
=> find(/host/key, 10m, "like", "error")  
→ find a value that contains ‘error’ within the last 10 minutes until now
```

**first** (/host/key, sec<:time shift>)

The first (the oldest) value within the defined evaluation period. See common parameters.

Supported value types: float, int, str, text, log

Example:

```
=> first(/host/key, 1h) → retrieve the oldest value within the last hour until now
```

See also last().
FUNCTION

Checking how much the passive agent time differs from the Zabbix server/proxy time.

See common-parameters.

Supported value types: float, int

Returns:
1 - difference between the passive item value (as timestamp) and Zabbix server/proxy timestamp (clock of value collection) is less than or equal to T seconds
0 - otherwise

Usually used with the ‘system.localtime’ item to check that local time is in sync with the local time of Zabbix server. Note that ‘system.localtime’ must be configured as a passive check.

Can be used also with vfs.file.time[/path/file,modify] key to check that file didn’t get updates for long time.

Example:
=> fuzzytime(/host/key,60s)=0 → detect a problem if the time difference is over 60 seconds

This function is not recommended for use in complex trigger expressions (with multiple items involved), because it may cause unexpected results (time difference will be measured with the most recent metric), e.g. in fuzzytime(/Host/system.localtime,60s)=0 or last(/Host/trap)<>0

---

last (/host/key.<#num:<time shift>>)

The most recent value.

See common parameters.

#num (optional) - the Nth most recent value

Supported value types: float, int, str, text, log

Take note that a hash-tagged time period (#N) works differently here than with many other functions.

For example:
last() is always equal to last(#1)
last(#3) - third most recent value (not three latest values)

Zabbix does not guarantee the exact order of values if more than two values exist within one second in history.

Example:
=> last(/host/key) → retrieve the last value
=> last(/host/key,#2) → retrieve the previous value
=> last(/host/key,#1) <>

See also first().
FUNCTION

logeventid (/host/key,<#num<:time shift>,<pattern>)
Checking if event ID of the last log entry matches a regular expression.

See common parameters.

#num (optional) - the Nth most recent value
pattern (optional) - regular expression describing the required pattern, Perl Compatible Regular Expression (PCRE) style (string arguments must be double-quoted).

Supported value types: log

Returns:
0 - does not match
1 - matches

logseverity (/host/key,<#num<:time shift>)
Log severity of the last log entry.

See common parameters.

#num (optional) - the Nth most recent value

Supported value types: log

Returns:
0 - default severity
N - severity (integer, useful for Windows event logs: 1 - Information, 2 - Warning, 4 - Error, 7 - Failure Audit, 8 - Success Audit, 9 - Critical, 10 - Verbose).
Zabbix takes log severity from Information field of Windows event log.

logsource (/host/key,<#num<:time shift>,<pattern>)
Checking if log source of the last log entry matches a regular expression.

See common parameters.

#num (optional) - the Nth most recent value
pattern (optional) - regular expression describing the required pattern, Perl Compatible Regular Expression (PCRE) style (string arguments must be double-quoted).

Supported value types: log

Returns:
0 - does not match
1 - matches

Normally used for Windows event logs. For example, logsource("VMware Server").

monodec (/host/key,(sec|#num)<:time shift>,<mode>)
Check if there has been a monotonous decrease in values.

See common parameters.

mode (must be double-quoted) - weak (every value is smaller or the same as the previous one; default) or strict (every value has decreased)

Supported value types: int

Returns 1 if all elements in the time period continuously decrease, 0 otherwise.

Example:
=> monodec(/Host1/system.swap.size[all,free],60s) + monodec(/Host2/system.swap.size[all,free],60s) + monodec(/Host3/system.swap.size[all,free],60s) - calculate in how many hosts there has been a decrease in free swap size

monoinc (/host/key,(sec|#num)<:time shift>,<mode>)
FUNCTION

Check if there has been a monotonous increase in values.

See common parameters.

**mode** (must be double-quoted) - weak (every value is bigger or the same as the previous one; default) or strict (every value has increased)

Supported value types: int

Returns 1 if all elements in the time period continuously increase, 0 otherwise.

Example:

```
=> monoinc(/Host1/system.localtime,#3,"strict")=0
```

- check if system local time has been increasing consistently

**nodata** (/host/key,sec,<mode>)

Checking for no data received.

See common parameters.

**sec** period should not be less than 30 seconds because the history syncer process calculates this function only every 30 seconds.

**nodata**(/host/key,0) is disallowed.

**mode** - if set to strict (double-quoted), this function will be insensitive to proxy availability (see comments for details).

All value types are supported.

Returns:

1 - if no data received during the defined period of time
0 - otherwise

Since Zabbix 5.0, the 'nodata' triggers monitored by proxy are, by default, sensitive to proxy availability - if proxy becomes unavailable, the 'nodata' triggers will not fire immediately after a restored connection, but will skip the data for the delayed period. Note that for passive proxies suppression is activated if connection is restored more than 15 seconds and no less than 2 & ProxyUpdateFrequency seconds later. For active proxies suppression is activated if connection is restored more than 15 seconds later.

To turn off sensitiveness to proxy availability, use the third parameter, e.g.: `nodata(/host/key,5m,"strict")`; in this case the function will work the same as before 5.0.0 and fire as soon as the evaluation period (five minutes) without data has past.

Note that this function will display an error if, within the period of the 1st parameter:
- there's no data and Zabbix server was restarted
- there's no data and maintenance was completed
- there's no data and the item was added or re-enabled

Errors are displayed in the Info column in trigger configuration.

This function may not work properly if there are time differences between Zabbix server, proxy and agent. See also: Time synchronization requirement.

**percentile**

(/host/key,(sec|num)<:time shift>,.percentage)

1503
<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>Description</th>
<th>Function-specific parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>baselinedev (/host/key, data period:time shift,season_unit,num_seasons)</td>
<td>P-th percentile of a period, where P (percentage) is specified by the third parameter.</td>
<td>See common parameters.</td>
<td>Supported value types: float, int</td>
</tr>
<tr>
<td>rate (/host/key,sec&lt;:timeshift&gt;)</td>
<td>Per-second average rate of the increase in a monotonically increasing counter within the defined time period.</td>
<td>See common parameters.</td>
<td>Supported value types: float, int</td>
</tr>
</tbody>
</table>

**5 Trend functions**

Trend functions, in contrast to history functions, use trend data for calculations. Trends store hourly aggregate values. Trend functions use these hourly averages, and thus are useful for long-term analysis. Trend function results are cached so multiple calls to the same function with the same parameters fetch info from the database only once. The trend function cache is controlled by the TrendCacheSize server parameter.

Triggers that reference trend functions only are evaluated once per the smallest time period in the expression. For instance, a trigger like

```
trendavg(/host/key,1d::now/d) > 1 or trendavg(/host/key2,1w::now/w) > 2
```

will be evaluated once per day. If the trigger contains both trend and history (or time-based) functions, it is calculated in accordance with the usual principles.

All functions listed here are supported in:
- Trigger expressions
- Calculated items

Some general notes on function parameters:
- Function parameters are separated by a comma
- Optional function parameters (or parameter parts) are indicated by `< >`
- Function-specific parameters are described with each function
- `/host/key` and `time period:time shift` parameters must never be quoted

Common parameters:
- `/host/key` is a common mandatory first parameter
- `time period:time shift` is a common second parameter, where:
  - `time period` - the time period (minimum '1h'), defined as `<N><time unit>` where N - the number of time units, `time unit` - h (hour), d (day), w (week), M (month) or y (year).
  - `time shift` - the time period offset (see function examples)

**Trend functions**

```
FUNCTION

Returns the number of deviations (by stddevpop algorithm) between the last data period and the same data periods in preceding seasons.

data period - the data gathering period within a season, defined as
< N > < time unit > where
N - number of time units
time unit - h (hour), d (day), w (week), M (month) or y (year), must be equal to or less than season

Time shift - the time period offset (see examples)

season unit - duration of one season (h, d, w, M, y), cannot be smaller than data period

num seasons - number of seasons to evaluate

Examples:
=> base-linedev(/host/key,1d:now/d,"M",6) → calculating the number of standard deviations (population) between the previous day and the same day in the previous 6 months. If the date doesn’t exist in a previous month, the last day of the month will be used (Jul,31 will be analysed against Jan,31, Feb, 28,... June, 30).

= base-linedev(/host/key,1h:now/h,"d",10) → calculating the number of standard deviations (population) between the previous hour and the same hours over the period of ten days before yesterday.

baselinewma (/host/key,data period:time shift,season unit,num seasons)
Calculates the baseline by averaging data from the same timeframe in multiple equal time periods (‘seasons’) using the weighted moving average algorithm.

data period - the data gathering period within a season, defined as
< N > < time unit > where
N - number of time units
time unit - h (hour), d (day), w (week), M (month) or y (year), must be equal to or less than season

Time shift - the time period offset, defines the end of data gathering time frame in seasons (see examples)

season unit - duration of one season (h, d, w, M, y), cannot be smaller than data period

num seasons - number of seasons to evaluate

Examples:
=> base-linewma(/host/key,1h:now/h,"d",3) → calculating baseline based on the last full hour within a 3-day period that ended yesterday. If “now” is Monday 13:30, the data for 12:00-12:59 on Friday, Saturday, and Sunday will be analyzed.

= base-linemwa(/host/key,2h:now/h,"d",3) → calculating baseline based on the last two hours within a 3-day period that ended yesterday. If “now” is Monday 13:30, the data for 10:00-11:59 on Friday, Saturday, and Sunday will be analyzed.

= base-linemwa(/host/key,1d:now/d,"M",4) → calculating baseline based on the same day of month as ‘yesterday’ in the 4 months preceding the last full month. If required date doesn’t exist, the last day of month is taken. If today is September 1st, the data for July 31st, June 30th, May 31st, April 30th will be analyzed.

trendavg (/host/key,time period:time shift)
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Examples:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>trendavg</strong></td>
<td>Average of trend values within the defined time period.</td>
<td>=&gt; trendavg(/host/key, 1h:now/h) → average for the previous hour (e.g. 12:00-13:00)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>=&gt; trendavg(/host/key, 1h:now/h-1h) → average for two hours ago (11:00-12:00)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>=&gt; trendavg(/host/key, 1h:now/h-2h) → average for three hours ago (10:00-11:00)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>=&gt; trendavg(/host/key, 1M:now/M-1y) → average for the previous month a year ago</td>
</tr>
<tr>
<td><strong>trendcount</strong></td>
<td>Number of successfully retrieved trend values within the defined time period.</td>
<td>=&gt; trendcount(/host/key, 1h:now/h) → count for the previous hour (e.g. 12:00-13:00)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>=&gt; trendcount(/host/key, 1h:now/h-1h) → count for two hours ago (11:00-12:00)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>=&gt; trendcount(/host/key, 1h:now/h-2h) → count for three hours ago (10:00-11:00)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>=&gt; trendcount(/host/key, 1M:now/M-1y) → count for the previous month a year ago</td>
</tr>
<tr>
<td><strong>trendmax</strong></td>
<td>The maximum in trend values within the defined time period.</td>
<td>=&gt; trendmax(/host/key, 1h:now/h) → maximum for the previous hour (e.g. 12:00-13:00)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>=&gt; trendmax(/host/key, 1h:now/h) - trendmin(/host/key, 1h:now/h) → calculate the difference</td>
</tr>
<tr>
<td></td>
<td></td>
<td>between the maximum and minimum values (trend delta) for the previous hour (12:00-13:00)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>=&gt; trendmax(/host/key, 1h:now/h-1h) → maximum for two hours ago (11:00-12:00)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>=&gt; trendmax(/host/key, 1h:now/h-2h) → maximum for three hours ago (10:00-11:00)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>=&gt; trendmax(/host/key, 1M:now/M-1y) → maximum for the previous month a year ago</td>
</tr>
</tbody>
</table>

**trendcount** (/host/key, time period: time shift)

Number of successfully retrieved trend values within the defined time period. See common parameters.

**trendmax** (/host/key, time period: time shift)

The maximum in trend values within the defined time period. See common parameters.
The minimum in trend values within the defined time period.

See common parameters.

Examples:

- `trendmin(/host/key, 1h:now/h) →` minimum for the previous hour (e.g. 12:00-13:00)
- `trendmin(/host/key, 1h:now/h) - trendmin(/host/key, 1h:now/h) →` calculate the difference between the maximum and minimum values (trend delta) for the previous hour (12:00-13:00)
- `trendmin(/host/key, 1h:now/h-1h) →` minimum for two hours ago (11:00-12:00)
- `trendmin(/host/key, 1h:now/h-2h) →` minimum for three hours ago (10:00-11:00)
- `trendmin(/host/key, 1M:now/M-1y) →` minimum for the previous month a year ago

`trendstl` (/host/key, eval period:time shift, detection period, season, <deviations>, <devalg>, <s_window>)
FUNCTION

Returns the rate of anomalies during the detection period - a decimal value between 0 and 1 that is \(((\text{the number of anomaly values})/\text{(total number of values)})\).

**eval period** - the time period that must be decomposed (minimum '1h'), defined as \(<N><\text{time unit}>\) where
- \(N\) - number of time units
- time unit - h (hour), d (day), w (week), M (month) or y (year).

**Time shift** - the time period offset (see examples)

**detection period** - the time period before the end of eval period for which anomalies are calculated (minimum '1h', cannot be longer than eval period), defined as \(<N><\text{time unit}>\) where
- \(N\) - number of time units
- time unit - h (hour), d (day), w (week).

**season** - the shortest time period where a repeating pattern ("season") is expected (minimum '2h', cannot be longer than eval period), number of entries in the eval period must be greater than the two times of the resulting frequency (season/h)), defined as \(<N><\text{time unit}>\) where
- \(N\) - number of time units
- time unit - h (hour), d (day), w (week).

**deviations** - the number of deviations (calculated by devalg) to count as anomaly (can be decimal), (must be greater than or equal to 1, default is 3)

**devalg** (must be double-quoted) - deviation algorithm, can be stddevpop, stddevsamp or mad (default)

**s_window** - the span (in lags) of the loess window for seasonal extraction (default is 10 * number of entries in eval period + 1)

**Examples:**

- `=> trendstl(/host/key,100h:now/h,10h,2h)` → analyse the last 100 hours of trend data, find the anomaly rate for the last 10 hours of that period, expecting the periodicity to be 2h, the remainder series values of the evaluation period are considered anomalies if they reach the value of 3 deviations of the MAD of that remainder series

- `=> trendstl(/host/key,100h:now/h,10h,100h,2h,2.1,"mad")` → analyse the period of 100 hours of trend data, up to 10 hours ago, find the anomaly rate for that entire period expecting the periodicity to be 2h, the remainder series values of the evaluation period are considered anomalies if they reach the value of 2.1 deviations of the MAD of that remainder series

- `=> trendstl(/host/key,100d:now/d-1d,10d,1d,4"10)→analyse100days`→ analyse 100 days of trend data up to a day ago, find the anomaly rate for the period of last 10d of that period, expecting the periodicity to be 1d, the remainder series values of the evaluation period are considered anomalies if they reach the value of 4 deviations of the MAD of that remainder series, overriding the default span of the loess window for seasonal extraction of "10 * number of entries in eval period + 1" with the span of 10 lags

- `=> trendstl(/host/key,1M:now/M-1y,1d,2h,"stddevsamp")` → analyse the previous month a year ago, find the anomaly rate of the last day of that period expecting the periodicity to be 2h, the remainder series values of the evaluation period are considered anomalies if they reach the value of 3 deviation of the sample standard deviation of that remainder series

**trendsum** (/host/key,time period:time shift)
FUNCTION
Sum of trend values within the defined time period.
See common parameters.

Examples:
=> \texttt{trendsum(/host/key,1h:now/h)} \rightarrow \text{sum for the previous hour (e.g. 12:00-13:00)}
=> \texttt{trendsum(/host/key,1h:now/h-1h)} \rightarrow \text{sum for two hours ago (11:00-12:00)}
=> \texttt{trendsum(/host/key,1h:now/h-2h)} \rightarrow \text{sum for three hours ago (10:00-11:00)}
=> \texttt{trendsum(/host/key,1M:now/M-1y)} \rightarrow \text{sum for the previous month a year ago}

6 Mathematical functions

All functions listed here are supported in:

\begin{itemize}
  \item Trigger expressions
  \item Calculated items
\end{itemize}

Mathematical functions are supported with float and integer value types, unless stated otherwise.

Some general notes on function parameters:

\begin{itemize}
  \item Function parameters are separated by a comma
  \item Expressions are accepted as parameters
  \item Optional function parameters (or parameter parts) are indicated by < >
\end{itemize}

FUNCTION

<table>
<thead>
<tr>
<th>Description</th>
<th>Function-specific parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs (value)</td>
<td>value - value to check</td>
<td>Supported value types: float, int, str, text, log For strings returns: 0 - values are equal 1 - values differ Example: =&gt; \texttt{abs(last(/host/key))}&gt;10 Absolute numeric difference will be calculated, as seen with these incoming example values ('previous' and 'latest' value = absolute difference): '1' and '5' = 4 '3' and '1' = 2 '0' and '-2.5' = 2.5 The value must be between -1 and 1. For example, the arccosine of a value '0.5' will be '2.0943951'. Example: =&gt; \texttt{acos(last(/host/key))}</td>
</tr>
<tr>
<td>acos (value)</td>
<td>value - value to check</td>
<td></td>
</tr>
<tr>
<td>asin (value)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Function Overview

**Function**: The arcsine of a value as an angle, expressed in radians.  
**Value**: value - value to check  
**Example**: The value must be between -1 and 1.  
*For example, the arcsine of a value '0.5' will be '0.523598776'.*

### atan (value)

**Function**: The arctangent of a value as an angle, expressed in radians.  
**Value**: value - value to check  
**Example**: For example, the arctangent of a value '1' will be '0.785398163'.

### atan2 (value,abscissa)

**Function**: The arctangent of the ordinate (exprue) and abscissa coordinates specified as an angle, expressed in radians.  
**Value**: value - value to check  
**Abscissa**: abscissa - abscissa value  
**Example**: For example, the arctangent of the ordinate and abscissa coordinates of a value '1' will be '2.21429744'.

### avg (<value1>,<value2>,...)

**Function**: Average value of the referenced item values.  
**Value**: valueX - value returned by one of history functions  
**Example**: For example, the cube root of '64' will be '4', of '63' will be '3.97905721'.

### ceil (value)

**Function**: Round the value up to the nearest greater or equal integer.  
**Value**: value - value to check  
**Example**: For example, '2.4' will be rounded up to '3'.

### cos (value)

**Function**: The cosine of a value, where the value is an angle expressed in radians.  
**Value**: value - value to check  
**Example**: For example, the cosine of a value '1' will be '0.54030230586'.

### cbrt (value)

**Function**: Cuberoot of a value.  
**Value**: value - value to check  
**Example**: For example, the cube root of '64' will be '4', of '63' will be '3.97905721'.

### cosh (value)

**Function**: The hyperbolic cosine of a value.  
**Value**: value - value to check  
**Example**: For example, the hyperbolic cosine of a value '1' will be '1.54308063482'.

### cot (value)

**Function**: The cotangent of a value, where the value is an angle, expressed in radians.  
**Value**: value - value to check  
**Example**: For example, the cotangent of a value '1' will be '0.54030230586'.

### degrees (value)

**Function**: The arcsine of a value as an angle, expressed in radians.
### FUNCTION

Converts a value from radians to degrees.

<table>
<thead>
<tr>
<th>function</th>
<th>description</th>
<th>value - value to check</th>
<th>examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>degrees()</strong></td>
<td>Converts a value from radians to</td>
<td>value</td>
<td>For example, a value ‘1’ converted to degrees will be ‘57.2957795’.</td>
</tr>
<tr>
<td></td>
<td>degrees</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>e</strong></td>
<td>Euler’s number (2.718281828459045)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>exp(value)</strong></td>
<td>Euler’s number at a power of a value</td>
<td>value</td>
<td>For example, Euler’s number at a power of a value ‘2’ will be ‘7.38905609893065’.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>expm1(value)</strong></td>
<td>Euler’s number at a power of a value minus 1.</td>
<td>value</td>
<td>For example, Euler’s number at a power of a value ‘2’ minus 1 will be ‘6.38905609893065’.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>floor(value)</strong></td>
<td>Round the value down to the nearest smaller or equal integer.</td>
<td>value</td>
<td>For example, ‘2.6’ will be rounded down to ‘2’.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>log(value)</strong></td>
<td>Natural logarithm.</td>
<td>value</td>
<td>For example, the natural logarithm of a value ‘2’ will be ‘0.69314718055994529’.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>log10(value)</strong></td>
<td>Decimal logarithm.</td>
<td>value</td>
<td>For example, the decimal logarithm of a value ‘5’ will be ‘0.69897000433’.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>max(&lt;value1&gt;,&lt;value2&gt;,....)</strong></td>
<td>Highest value of the referenced item values.</td>
<td>valueX - value returned by one of history functions</td>
<td>For example: max(avg(/host/key),avg(/host2/key2))</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>min(&lt;value1&gt;,&lt;value2&gt;,....)</strong></td>
<td>Lowest value of the referenced item values.</td>
<td>valueX - value returned by one of history functions</td>
<td>For example: min(avg(/host/key),avg(/host2/key2))</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>mod(value,denominator)</strong></td>
<td>Division remainder.</td>
<td>value - value to check denominator - division denominator</td>
<td>For example, division remainder of a value ‘5’ with division denominator ‘2’ will be ‘1’.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>pi</strong></td>
<td>Pi constant (3.14159265358979)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example:

- `=> degrees(last(/host/key))`
- `=> e()`
- `=> exp(last(/host/key))`
- `=> expm1(last(/host/key))`
- `=> floor(last(/host/key))`
- `=> log(last(/host/key))`
- `=> log10(last(/host/key))`
- `=> max(avg(/host/key),avg(/host2/key2))`
- `=> min(avg(/host/key),avg(/host2/key2))`
- `=> mod(last(/host/key),2)`
- `=> pi()`
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>power</strong> (value, power value)</td>
<td>The power of a value. value - value to check, power value - the Nth power to use</td>
<td>For example, the 3rd power of a value '2' will be '8'. Example: =&gt; power(last(/host/key), 3)</td>
</tr>
<tr>
<td><strong>radians</strong> (value)</td>
<td>Convert a value from degrees to radians. value - value to check</td>
<td>For example, a value '1' converted to radians will be '0.0174532925'. Example: =&gt; radians(last(/host/key))</td>
</tr>
<tr>
<td><strong>rand</strong></td>
<td>Return a random integer value.</td>
<td>A pseudo-random generated number using time as seed (enough for mathematical purposes, but not cryptography). Example: =&gt; rand()</td>
</tr>
<tr>
<td><strong>round</strong> (value, decimal places)</td>
<td>Round the value to decimal places. value - value to check, decimal places - specify decimal places for rounding (0 is also possible)</td>
<td>For example, a value '2.5482' rounded to 2 decimal places will be '2.55'. Example: =&gt; round(last(/host/key), 2)</td>
</tr>
<tr>
<td><strong>signum</strong> (value)</td>
<td>Returns '-1' if a value is negative, '0' if a value is zero, '1' if a value is positive. value - value to check</td>
<td>Example: =&gt; signum(last(/host/key))</td>
</tr>
<tr>
<td><strong>sin</strong> (value)</td>
<td>The sine of a value, where the value is an angle expressed in radians. value - value to check</td>
<td>For example, the sine of a value '1' will be '0.8414709848'. Example: =&gt; sin(last(/host/key))</td>
</tr>
<tr>
<td><strong>sinh</strong> (value)</td>
<td>The hyperbolical sine of a value. value - value to check</td>
<td>For example, the hyperbolical sine of a value '1' will be '1.17520119364'. Example: =&gt; sinh(last(/host/key))</td>
</tr>
<tr>
<td><strong>sqrt</strong> (value)</td>
<td>Square root of a value. value - value to check</td>
<td>This function will fail with a negative value. For example, the square root of a value '3.5' will be '1.87082869339'. Example: =&gt; sqrt(last(/host/key))</td>
</tr>
<tr>
<td><strong>sum</strong> (&lt;value1&gt;, &lt;value2&gt;,...)</td>
<td>Sum of the referenced item values. valueX - value returned by one of history functions</td>
<td>Example: =&gt; sum(avg(/host/key), avg(/host2/key2))</td>
</tr>
<tr>
<td><strong>tan</strong> (value)</td>
<td>The tangent of a value. value - value to check</td>
<td>For example, the tangent of a value '1' will be '1.55740772465'. Example: =&gt; tan(last(/host/key))</td>
</tr>
</tbody>
</table>
FUNCTION

Truncate the value to decimal places.

value - value to check
decimal places - specify decimal places for truncating (0 is also possible)

Example:
=> truncate(last(/host/key),2)

7 Operator functions

All functions listed here are supported in:

• Trigger expressions
• Calculated items

Some general notes on function parameters:

• Function parameters are separated by a comma
• Expressions are accepted as parameters

FUNCTION

<table>
<thead>
<tr>
<th>Description</th>
<th>Function-specific parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>between (value,min,max)</td>
<td>value - value to check min - minimum value max - maximum value</td>
<td>Supported value types: integer, float Returns: 1 - in range 0 - otherwise Example: =&gt; between(last(/host/key),1,10)=1 - trigger if the value is between 1 and 10.</td>
</tr>
<tr>
<td>in (value,value1,value2,...,valueN)</td>
<td>value - value to check value1,value2,...,valueN - listed values (string values must be double-quoted)</td>
<td>Supported value types: all Returns: 1 - if equal 0 - otherwise The value is compared to the listed values as numbers, if all of these values can be converted to numeric; otherwise compared as strings. Example: =&gt; in(last(/host/key),5,10)=1 - trigger if the last value is equal to 5 or 10 =&gt; in(&quot;text&quot;, last(/host/key),last(/host/key,#2))=1 - trigger if &quot;text&quot; is equal to either of the last 2 values.</td>
</tr>
</tbody>
</table>

8 Prediction functions

All functions listed here are supported in:

• Trigger expressions
• Calculated items

Some general notes on function parameters:

• Function parameters are separated by a comma
• Optional function parameters (or parameter parts) are indicated by < >
• Function-specific parameters are described with each function
• `/host/key` and `(sec|num)<:time shift>` parameters must never be quoted

Common parameters
• `/host/key` is a common mandatory first parameter for the functions referencing the host item history
• `(sec|num)<:time shift>` is a common second parameter for the functions referencing the host item history, where:
  - `sec` - maximum evaluation period in seconds (time suffixes can be used), or
  - `num` - maximum evaluation range in latest collected values (if preceded by a hash mark)
  - `time shift` (optional) allows to move the evaluation point back in time. See more details on specifying time shift.

Prediction functions

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>Description</th>
<th>Function-specific parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>forecast</td>
<td>Future value, max, min, delta or avg of the item.</td>
<td>See common parameters.</td>
<td>Supported value types: float, int</td>
</tr>
</tbody>
</table>

- **time** - forecasting horizon in seconds (time suffixes can be used); negative values are supported
- **fit** (optional; must be double-quoted) - function used to fit historical data

Supported fits:
- linear - linear function
- polynomialN - polynomial of degree N (1 <= N <= 6)
- exponential - exponential function
- logarithmic - logarithmic function
- power - power function

Note that:
- linear is default, polynomial1 is equivalent to linear
- **mode** (optional; must be double-quoted) - demanded output

Supported modes:
- value - value (default)
- max - maximum
- min - minimum
- delta - max-min
- avg - average

Note that:
- value estimates item value at the moment `now + time`
- max, min, delta and avg investigate item value estimate on the interval between `now` and `now + time`

If value to return is larger than 1.7976931348623157E+308 or less than -1.7976931348623157E+308, return value is cropped to 1.7976931348623157E+308 or -1.7976931348623157E+308 correspondingly.

Becomes unsupported only if misused in expression (wrong item type, invalid parameters), otherwise returns -1 in case of errors.

Examples:
- `=> forecast(/host/key,#10,1h)` → forecast item value in one hour based on the last 10 values
- `=> forecast(/host/key,1h,30m)` → forecast item value in 30 minutes based on the last hour data
- `=> forecast(/host/key,1h:now-1d,12h)` → forecast item value in 12 hours based on one hour one day ago
- `=> forecast(/host/key,1h,10m,“exponential”)` → forecast item value in 10 minutes based on the last hour data and exponential function
- `=> forecast(/host/key,1h,2h,”polynomial3”,“max”)` → forecast the maximum value the item can reach in the next two hours based on last hour data and cubic (third degree) polynomial
- `=> forecast(/host/key,#2,-20m)` → estimate the item value 20 minutes ago based on the last two values (this can be more precise than using last(), especially if item is updated rarely, say, once an hour)

See also additional information on predictive trigger functions.
**FUNCTION**

**timeleft** (/host/key,(sec|#num)<:time shift>,threshold,<fit>)

Time in seconds needed for an item to reach a specified threshold.

- **threshold** - value to reach (unit suffixes can be used)
- **fit** (optional; must be double-quoted) - see forecast()

See [common parameters](#).

Supported value types: float, int

If value to return is larger than 1.7976931348623157E+308, return value is cropped to 1.7976931348623157E+308.

Returns 1.7976931348623157E+308 if threshold cannot be reached.

Becomes unsupported only if misused in the expression (wrong item type, invalid parameters), otherwise returns -1 in case of errors.

Examples:

- `=> timeleft(/host/key,#10,0) →` time until the item value reaches zero based on the last 10 values
- `=> timeleft(/host/key,1h,100) →` time until the item value reaches 100 based on the last hour data
- `=> timeleft(/host/key,1h:now-1d,100) →` time until the item value reaches 100 based on one hour one day ago
- `=> timeleft(/host/key,1h,200,"polynomial2") →` time until the item value reaches 200 based on the last hour data and assumption that the item behaves like quadratic (second degree) polynomial

See also additional information on [predictive trigger functions](#).

---

**9 String functions**

All functions listed here are supported in:

- Trigger expressions
- Calculated items

Some general notes on function parameters:

- Function parameters are separated by a comma
- Expressions are accepted as parameters
- String parameters must be double-quoted; otherwise they might get misinterpreted
- Optional function parameters (or parameter parts) are indicated by `< >`

---

**FUNCTION**

<table>
<thead>
<tr>
<th>Description</th>
<th>Function-specific parameters</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ascii (value)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FUNCTION</td>
<td>Description</td>
<td>Supported value types:</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td><strong>The ASCII code of the leftmost character of the value.</strong></td>
<td>value - value to check</td>
<td>string, text, log</td>
</tr>
<tr>
<td><strong>bitlength (value)</strong></td>
<td>The length of value in bits.</td>
<td>string, text, log, integer</td>
</tr>
<tr>
<td><strong>bytelength (value)</strong></td>
<td>The length of value in bytes.</td>
<td>string, text, log, integer</td>
</tr>
<tr>
<td><strong>char (value)</strong></td>
<td>Return the character by interpreting the value as ASCII code.</td>
<td>integer</td>
</tr>
<tr>
<td><strong>concat (&lt;value1&gt;,&lt;value2&gt;,...)</strong></td>
<td>The string resulting from concatenating referenced item values or constant values.</td>
<td>string, text, log, float, integer</td>
</tr>
<tr>
<td><strong>left (value,count)</strong></td>
<td>Insert specified characters or spaces into the character string beginning at the specified position in the string.</td>
<td>string, text, log</td>
</tr>
</tbody>
</table>

**Example:**

- `= ` ascii(last(/host/key))

- `= ` bitlength(last(/host/key))

- `= ` bytelength(last(/host/key))

- `= ` char(last(/host/key))

- `= ` concat(last(/host/key),"bix")

- `= ` concat("1 min:",last(/host/system.cpu.load[all,avg1]),","15 min:",last(/host/system.cpu.load[all,avg15]))

- `= ` insert(last(/host/key),3,2,"b")
FUNCTION

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Value</th>
<th>Count</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>left(value, count)</td>
<td>The leftmost characters of the value.</td>
<td>value - value to check</td>
<td>count - number of characters to return</td>
<td>Supported value types: string, text, log</td>
</tr>
<tr>
<td>For example, you may return 'Zab' from 'Zabbix' by specifying 3 leftmost characters to return.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>$=&gt; \text{left(last(/host/key),3)}$ - return three leftmost characters</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Value</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>length(value)</td>
<td>The length of value in characters.</td>
<td>value - value to check</td>
<td>Supported value types: str, text, log</td>
</tr>
<tr>
<td>Example:</td>
<td>$=&gt; \text{length(last(/host/key))}$ → length of the latest value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$=&gt; \text{length(last(/host/key,#3))}$ → length of the third most recent value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$=&gt; \text{length(last(/host/key,#1:now-1d))}$ → length of the most recent value one day ago</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Value</th>
<th>Count</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>ltrim(value, chars)</td>
<td>Remove specified characters from the beginning of string.</td>
<td>value - value to check</td>
<td>chars - (optional) specify characters to remove</td>
<td></td>
</tr>
<tr>
<td>Supported value types: string, text, log</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>$=&gt; \text{ltrim(last(/host/key))}$ - remove whitespace from the beginning of string</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$=&gt; \text{ltrim(last(/host/key),&quot;Z&quot;)}$ - remove any 'Z' from the beginning of string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$=&gt; \text{ltrim(last(/host/key), &quot;Z&quot;)}$ - remove any space and 'Z' from the beginning of string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>See also: rtrim(), trim()</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Value</th>
<th>Count</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>mid(value, start, length)</td>
<td>Return a substring of N characters beginning at the character position specified by 'start'.</td>
<td>value - value to check</td>
<td>start - start position of substring</td>
<td>Supported value types: string, text, log</td>
</tr>
<tr>
<td>length - positions to return in substring</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>For example, it is possible return 'abbi' from a value like 'Zabbix' if starting position is 2, and positions to return is 4).</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>$=&gt; \text{mid(last(/host/key),2,4)}$=&quot;abbi&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Value</th>
<th>Count</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>repeat(value, count)</td>
<td>Repeat a string.</td>
<td>value - value to check</td>
<td>count - number of times to repeat</td>
<td>Supported value types: string, text, log</td>
</tr>
<tr>
<td>Example:</td>
<td>$=&gt; \text{repeat(last(/host/key),2)}$ - repeat the value two times</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Value</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>replace(value, pattern, replacement)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### FUNCTION

Find pattern in the value and replace with replacement. All occurrences of the pattern will be replaced.

- **value** - value to check
- **pattern** - pattern to find
- **replacement** - string to replace the pattern with

**Supported value types:** string, text, log

**Example:**

```text
=> replace(last(/host/key), "ibb", "abb")
```
- replace all `ibb` with `abb`

#### right (value,count)

The rightmost characters of the value.

- **value** - value to check
- **count** - number of characters to return

**Supported value types:** string, text, log

**Example:**

```text
=> right(last(/host/key), 3) - return three rightmost characters
```

#### rtrim (value,<chars>)

Remove specified characters from the end of string.

- **value** - value to check
- **chars** - (optional) specify characters to remove

Whitespace is right-trimmed by default (if no optional characters are specified).

**Supported value types:** string, text, log

**Example:**

```text
=> rtrim(last(/host/key)) - remove whitespace from the end of string
=> rtrim(last(/host/key), "x") - remove any 'x' from the end of string
```

#### trim (value,<chars>)

Remove specified characters from the beginning and end of string.

- **value** - value to check
- **chars** - (optional) specify characters to remove

Whitespace is trimmed from both sides by default (if no optional characters are specified).

**Supported value types:** string, text, log

**Example:**

```text
=> trim(last(/host/key)) - remove whitespace from the beginning and end of string
=> trim(last(/host/key), "_.") - remove "." from the beginning and end of string
```

See also: ltrim(), trim()

---

### 7 Macros

#### 1 Supported macros

**Overview**

The table contains a complete list of macros supported by Zabbix out-of-the-box.

To see all macros supported in a location (for example, in "map URL"), you may paste the location name into the search box at the bottom of your browser window (accessible by pressing CTRL+F) and do a search for next.
<table>
<thead>
<tr>
<th>Macro</th>
<th>Supported in</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ACTION.ID}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Numeric ID of the triggered action.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Service-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Service update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Discovery notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Autoregistration notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td>{ACTION.NAME}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Name of the triggered action.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Service-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Service update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Discovery notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Autoregistration notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td>{ALERT.MESSAGE}</td>
<td>→ Alert script parameters</td>
<td>‘Default message’ value from action configuration. Supported since 3.0.0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{ALERT.SENDTO}</td>
<td>→ Alert script parameters</td>
<td>‘Send to’ value from user media configuration. Supported since 3.0.0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{ALERT.SUBJECT}</td>
<td>→ Alert script parameters</td>
<td>‘Default subject’ value from action configuration. Supported since 3.0.0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{DATE}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Current date in yyyy.mm.dd. format.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Service-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Service update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Discovery notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Autoregistration notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td>{DISCOVERY.DEVICE.IPADDRESS}</td>
<td>→ Discovery notifications and commands</td>
<td>IP address of the discovered device. Available always, does not depend on host being added.</td>
</tr>
<tr>
<td>{DISCOVERY.DEVICE.DNS}</td>
<td>→ Discovery notifications and commands</td>
<td>DNS name of the discovered device.</td>
</tr>
<tr>
<td>{DISCOVERY.DEVICE.STATUS}</td>
<td>→ Discovery notifications and commands</td>
<td>Status of the discovered device: can be either UP or DOWN.</td>
</tr>
<tr>
<td>{DISCOVERY.DEVICE.uptime}</td>
<td>→ Discovery notifications and commands</td>
<td>Time since the last change of discovery status for a particular device, with precision down to a second. For example: 1h 29m 01s. For devices with status DOWN, this is the period of their downtime.</td>
</tr>
<tr>
<td>{DISCOVERY.RULE.NAME}</td>
<td>→ Discovery notifications and commands</td>
<td>Name of the discovery rule that discovered the presence or absence of the device or service.</td>
</tr>
<tr>
<td>{DISCOVERY.SERVICE.NAME}</td>
<td>→ Discovery notifications and commands</td>
<td>Name of the service that was discovered.                                    For example: HTTP.</td>
</tr>
<tr>
<td>{DISCOVERY.SERVICE.PORT}</td>
<td>→ Discovery notifications and commands</td>
<td>Port of the service that was discovered.                                   For example: 80.</td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>{DISCOVERY.SERVICE.STATUS}</td>
<td>Trigger-based notifications and commands</td>
<td>Status of the discovered service:// can be either UP or DOWN.</td>
</tr>
<tr>
<td>{DISCOVERY.SERVICE.UPTIME}</td>
<td>Discovery notifications and commands</td>
<td>Time since the last change of discovery status for a particular service, with precision down to a second. For example: 1h 29m 01s. For services with status DOWN, this is the period of their downtime.</td>
</tr>
<tr>
<td>{ESC.HISTORY}</td>
<td>Trigger-based notifications and commands</td>
<td>Escalation history. Log of previously sent messages. Shows previously sent notifications, on which escalation step they were sent and their status (sent, in progress* or failed).</td>
</tr>
<tr>
<td>{EVENT.ACK.STATUS}</td>
<td>Acknowledgment status of the event (Yes/No).</td>
<td></td>
</tr>
<tr>
<td>{EVENT.AGE}</td>
<td>Age of the event that triggered an action, with precision down to a second. Useful in escalated messages.</td>
<td></td>
</tr>
<tr>
<td>{EVENT.DATE}</td>
<td>Date of the event that triggered an action.</td>
<td></td>
</tr>
<tr>
<td>{EVENT.DURATION}</td>
<td>Duration of the event (time difference between problem and recovery events), with precision down to a second. Useful in problem recovery messages. Supported since 5.0.0.</td>
<td></td>
</tr>
<tr>
<td>{EVENT.ID}</td>
<td>Numeric ID of the event that triggered an action.</td>
<td></td>
</tr>
<tr>
<td>{EVENT.NAME}</td>
<td>Name of the problem event that triggered an action. Supported since 4.0.0.</td>
<td></td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>{EVENT.SEVERITY}</td>
<td></td>
<td>→ Trigger-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Numeric value of the event severity. Possible values: 0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supportedsince 4.0.0.</td>
</tr>
<tr>
<td>{EVENT.OBJECT}</td>
<td></td>
<td>→ Trigger-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Discovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Autoregistration notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Internal notifications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Numeric value of the event object. Possible values: 0 - Trigger, 1 - Discovered host, 2 - Discovered service, 3 - Autoregistration, 4 - Item, 5 - Low-level discovery rule.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supportedsince 4.4.0.</td>
</tr>
<tr>
<td>{EVENT.OPDATA}</td>
<td></td>
<td>→ Trigger-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Operational data of the underlying trigger of a problem.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supportedsince 4.4.0.</td>
</tr>
<tr>
<td>{EVENT.RECOVERY.DATE}</td>
<td></td>
<td>→ Problem recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date of the recovery event.</td>
</tr>
<tr>
<td>{EVENT.RECOVERY.ID}</td>
<td></td>
<td>→ Problem recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Numeric ID of the recovery event.</td>
</tr>
<tr>
<td>{EVENT.RECOVERY.NAME}</td>
<td></td>
<td>→ Problem recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Name of the recovery event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supportedsince 4.4.1.</td>
</tr>
<tr>
<td>{EVENT.RECOVERY.STATUS}</td>
<td></td>
<td>→ Problem recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Verbal value of the recovery event.</td>
</tr>
<tr>
<td>{EVENT.RECOVERY.TAGS}</td>
<td></td>
<td>→ Problem recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A comma separated list of recovery event tags. Expanded to an empty string if no tags exist. Supportedsince 3.2.0.</td>
</tr>
<tr>
<td>{EVENT.RECOVERY.TAGS.JSON}</td>
<td></td>
<td>→ Problem recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A JSON array containing event tag objects. Expanded to an empty array if no tags exist. Supportedsince 5.0.0.</td>
</tr>
<tr>
<td>{EVENT.RECOVERY.TIME}</td>
<td></td>
<td>→ Problem recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Time of the recovery event.</td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>{EVENT.RECOVERY.VALUE}</td>
<td></td>
<td>Numeric value of the recovery event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands (if recovery took place)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands (if recovery took place)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td>{EVENT.SEVERITY}</td>
<td></td>
<td>Name of the event severity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Trigger-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported since 4.0.0.</td>
</tr>
<tr>
<td>{EVENT.SOURCE}</td>
<td></td>
<td>Numeric value of the event source. Possible values: 0 - Trigger, 1 - Discovery, 2 - Autoregistration, 3 - Internal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Trigger-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Discovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Autoregistration notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Internal notifications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td>{EVENT.STATUS}</td>
<td></td>
<td>Verbal value of the event that triggered an action.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Trigger-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Internal notifications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td>{EVENT.TAGS}</td>
<td></td>
<td>A comma separated list of event tags. Expanded to an empty string if no tags exist.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Trigger-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td>{EVENT.TAGS.JSON}</td>
<td></td>
<td>A JSON array containing event tag objects. Expanding to an empty array if no tags exist.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Trigger-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td>{EVENT.TAGS.&lt;tag name&gt;}</td>
<td></td>
<td>Event tag value referenced by the tag name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Trigger-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Webhook media type URL names and URLs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A tag name containing non-alphanumeric characters (including non-English multibyte-UTF characters) should be double quoted. Quotes and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>backslashes inside a quoted tag name must be escaped with a backslash.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supported since 4.4.2.</td>
</tr>
<tr>
<td>{EVENT.TIME}</td>
<td></td>
<td>Time of the event that triggered an action.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Trigger-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service-based notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service update notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Service recovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Discovery notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Autoregistration notifications and commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Internal notifications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>→ Manual event action scripts</td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>{EVENT.UPDATE.ACTION}</code></td>
<td>Problem update notifications and commands</td>
<td>Human-readable name of the action(s) performed during problem update. Resolves to the following values: acknowledged, commented, changed severity from (original severity) to (updated severity) and closed (depending on how many actions are performed in one update). Supported since 4.0.0.</td>
</tr>
<tr>
<td><code>{EVENT.UPDATE.DATE}</code></td>
<td>Problem update notifications and commands</td>
<td>Date of event update (acknowledgment, etc). Deprecated name: <code>{ACK.DATE}</code></td>
</tr>
<tr>
<td><code>{EVENT.UPDATE.HISTORY}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Log of problem updates (acknowledgments, etc). Deprecated name: <code>{EVENT.ACK.HISTORY}</code></td>
</tr>
<tr>
<td><code>{EVENT.UPDATE.MESSAGE}</code></td>
<td>Manual event action scripts</td>
<td>Problem update message. Deprecated name: <code>{ACK.MESSAGE}</code></td>
</tr>
<tr>
<td><code>{EVENT.UPDATE.STATUS}</code></td>
<td>Problem update notifications and commands</td>
<td>Numeric value of the problem update status. Possible values: 0 - Webhook was called because of problem/recovery event, 1 - Update operation. Supported since 4.4.0.</td>
</tr>
<tr>
<td><code>{EVENT.UPDATE.TIME}</code></td>
<td>Problem update notifications and commands</td>
<td>Time of event update (acknowledgment, etc). Deprecated name: <code>{ACK.TIME}</code></td>
</tr>
<tr>
<td><code>{EVENT.VALUE}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Numeric value of the event that triggered an action (1 for problem, 0 for recovering).</td>
</tr>
<tr>
<td><code>{FUNCTION.VALUE&lt;1-9&gt;}</code></td>
<td>Problem update notifications and commands</td>
<td>Results of the Nth item-based function in the trigger expression at the time of the event. Only functions with /host/key as the first parameter are counted. See indexed macros.</td>
</tr>
<tr>
<td><code>FUNCTION.RECOVERY.VALUE&lt;1-9&gt;</code></td>
<td>Recovery notifications and commands</td>
<td>Results of the Nth item-based function in the recovery expression at the time of the event. Only functions with /host/key as the first parameter are counted. See indexed macros.</td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>{HOST.CONN}</code></td>
<td>→ Trigger-based notifications and commands</td>
<td>Host IP address or DNS name, depending on host settings.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Item key parameters</td>
<td>May be used with a numeric index as {HOST.CONN&lt;1-9&gt;} to point to the first,</td>
</tr>
<tr>
<td></td>
<td>→ Host interface IP/DNS</td>
<td>second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Trapper item “Allowed hosts” field</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Database monitoring additional parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ JMX item endpoint field</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Web monitoring</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Low-level discovery rule filter regular expressions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ URL field of dynamic URL dashboard widget</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger names, event names, operational data and descriptions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger URLs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type item, item prototype and discovery rule parameter names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ HTTP agent type item, item prototype and discovery rule fields:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>URL, Query fields, Request body, Headers, Proxy, SSL certificate file, SSL key file, Allowed hosts.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts (including confirmation text)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts (including confirmation text)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{HOST.DESCRIPTION}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Host description.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td>This macro may be used with a numeric index e.g. {HOST.DESCRIPTION&lt;1-9&gt;} to</td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td>point to the first, second, third, etc. host in a trigger expression. See</td>
</tr>
<tr>
<td></td>
<td></td>
<td>indexed macros.</td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>{HOST.DNS}</td>
<td>→ Trigger-based notifications and commands&lt;br&gt;→ Problem update notifications and commands&lt;br&gt;→ Internal notifications&lt;br&gt;→ Map element labels, map URL names and values&lt;br&gt;→ Item key parameters&lt;br&gt;→ Host interface IP/DNS&lt;br&gt;→ Trapper item “Allowed hosts” field&lt;br&gt;→ Database monitoring additional parameters&lt;br&gt;→ SSH and Telnet scripts&lt;br&gt;→ JMX item endpoint field&lt;br&gt;→ Web monitoring&lt;br&gt;→ Low-level discovery rule filter regular expressions&lt;br&gt;→ URL field of dynamic URL dashboard widget&lt;br&gt;→ Trigger names, event names, operational data and descriptions&lt;br&gt;→ Trigger URLs&lt;br&gt;→ Tag names and values&lt;br&gt;→ Script-type item, item prototype and discovery rule parameter names and values&lt;br&gt;→ HTTP agent type item, item prototype and discovery rule fields: URL, Query fields, Request body, Headers, Proxy, SSL certificate file, SSL key file, Allowed hosts.</td>
<td>Host DNS name(^2). This macro may be used with a numeric index e.g. {HOST.DNS&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td>{HOST.HOST}</td>
<td>→ Trigger-based notifications and commands&lt;br&gt;→ Problem update notifications and commands&lt;br&gt;→ Autoregistration notifications and commands&lt;br&gt;→ Internal notifications&lt;br&gt;→ Item key parameters&lt;br&gt;→ Map element labels, map URL names and values&lt;br&gt;→ Host interface IP/DNS&lt;br&gt;→ Trapper item “Allowed hosts” field&lt;br&gt;→ Database monitoring additional parameters&lt;br&gt;→ SSH and Telnet scripts&lt;br&gt;→ JMX item endpoint field&lt;br&gt;→ Web monitoring&lt;br&gt;→ Low-level discovery rule filter regular expressions&lt;br&gt;→ URL field of dynamic URL dashboard widget&lt;br&gt;→ Trigger names, event names, operational data and descriptions&lt;br&gt;→ Trigger URLs&lt;br&gt;→ Tag names and values&lt;br&gt;→ Script-type item, item prototype and discovery rule parameter names and values&lt;br&gt;→ HTTP agent type item, item prototype and discovery rule fields: URL, Query fields, Request body, Headers, Proxy, SSL certificate file, SSL key file, Allowed hosts.</td>
<td>Host name. This macro may be used with a numeric index e.g. {HOST.HOST&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td>{HOSTNAME&lt;1-9&gt;}</td>
<td>→ Description of item value widget</td>
<td>{} is deprecated.</td>
</tr>
</tbody>
</table>

\(^1\) Item key parameters include: Host interface IP/DNS, Trapper item “Allowed hosts” field, Database monitoring additional parameters, SSH and Telnet scripts, JMX item endpoint field, Web monitoring, URL field of dynamic URL dashboard widget, Trigger names, event names, operational data and descriptions, Trigger URLs, Tag names and values, Script-type item, item prototype and discovery rule parameter names and values, HTTP agent type item, item prototype and discovery rule fields: URL, Query fields, Request body, Headers, Proxy, SSL certificate file, SSL key file, Allowed hosts, Manual host action scripts (including confirmation text), Manual event action scripts (including confirmation text), Description of item value widget.

\(^2\) Host name.
<table>
<thead>
<tr>
<th>Macro</th>
<th>Supported in</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>{HOST.ID}</strong></td>
<td>→ Trigger-based notifications and commands</td>
<td>Host ID.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td>May be used with a numeric index as {HOST.ID&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ URL field of dynamic URL dashboard widget</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger URLs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><strong>{HOST.IP}</strong></td>
<td>→ Trigger-based notifications and commands</td>
<td>Host IP address(^3).</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td>This macro may be used with a numeric index e.g. {HOST.IP&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Autoregistration notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Item key parameters(^1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Host interface IP/DNS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trapper item “Allowed hosts” field</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Database monitoring additional parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ SSH and Telnet scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ JMX item endpoint field</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Web monitoring(^4)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Low-level discovery rule filter regular expressions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ URL field of dynamic URL dashboard widget</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger names, event names, operational data and descriptions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger URLs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type item, item prototype and discovery rule parameter names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ HTTP agent type item, item prototype and discovery rule fields:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>URL, Query fields, Request body, Headers, Proxy,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SSL certificate file, SSL key file, Allowed hosts.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts (including confirmation text)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts (including confirmation text)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><strong>{HOST.METADATA}</strong></td>
<td>→ Autoregistration notifications and commands</td>
<td>Host metadata.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used only for active agent autoregistration.</td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>{HOST.NAME}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Visible host name.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Item key parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Host interface IP/DNS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trapper item “Allowed hosts” field</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Database monitoring additional parameters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ SSH and Telnet scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Web monitoring</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Low-level discovery rule filter regular expressions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ URL field of dynamic URL dashboard widget</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger names, event names, operational data and descriptions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger URLs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type item, item prototype and discovery rule parameter names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ HTTP agent type item, item prototype and discovery rule fields:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>URL, Query fields, Request body, Headers, Proxy, SSL certificate file, SSL key file, Allowed hosts.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts (including confirmation text)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts (including confirmation text)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{HOST.PORT}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Host (agent) port.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Autoregistration notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger names, event names, operational data and descriptions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger URLs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ JMX item endpoint field</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{HOST.TARGETCONN}</td>
<td>→ Trigger-based commands</td>
<td>IP address or DNS name of the target host, depending on host settings.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update commands</td>
<td>Supported since 5.4.0.</td>
</tr>
<tr>
<td></td>
<td>→ Discovery commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Autoregistration commands</td>
<td></td>
</tr>
<tr>
<td>{HOST.TARGETDNS}</td>
<td>→ Trigger-based commands</td>
<td>DNS name of the target host.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update commands</td>
<td>Supported since 5.4.0.</td>
</tr>
<tr>
<td></td>
<td>→ Discovery commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Autoregistration commands</td>
<td></td>
</tr>
<tr>
<td>{HOST.TARGETHOST}</td>
<td>→ Trigger-based commands</td>
<td>Technical name of the target host.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update commands</td>
<td>Supported since 5.4.0.</td>
</tr>
<tr>
<td></td>
<td>→ Discovery commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Autoregistration commands</td>
<td></td>
</tr>
<tr>
<td>{HOST.TARGETIP}</td>
<td>→ Trigger-based commands</td>
<td>IP address of the target host.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update commands</td>
<td>Supported since 5.4.0.</td>
</tr>
<tr>
<td></td>
<td>→ Discovery commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Autoregistration commands</td>
<td></td>
</tr>
<tr>
<td>{HOST.TARGETNAME}</td>
<td>→ Trigger-based commands</td>
<td>Visible name of the target host.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update commands</td>
<td>Supported since 5.4.0.</td>
</tr>
<tr>
<td></td>
<td>→ Discovery commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Autoregistration commands</td>
<td></td>
</tr>
<tr>
<td>{HOSTGROUP.ID}</td>
<td>→ Map element labels, map URL names and values</td>
<td>Host group ID.</td>
</tr>
</tbody>
</table>

4. This macro may be used with a numeric index e.g. {HOST.NAME<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

5. This macro may be used with a numeric index e.g. {HOST.PORT<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
<table>
<thead>
<tr>
<th>Macro</th>
<th>Supported in</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{INVENTORY.ALIAS}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Alias field in host inventory. This macro may be used with a numeric index e.g. {INVENTORY.ALIAS&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{INVENTORY.ASSET.TAG}</td>
<td>Asset tag field in host inventory. This macro may be used with a numeric index e.g. {INVENTORY.ASSET.TAG&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Trigger-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{INVENTORY.CHASSIS}</td>
<td>Chassis field in host inventory. This macro may be used with a numeric index e.g. {INVENTORY.CHASSIS&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Trigger-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{INVENTORY.CONTACT}</td>
<td>Contact field in host inventory. This macro may be used with a numeric index e.g. {INVENTORY.CONTACT&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Trigger-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{PROFILE.CONTACT&lt;1-9&gt;}</td>
<td>{PROFILE.CONTACT&lt;1-9&gt;} is deprecated.</td>
</tr>
<tr>
<td></td>
<td>→ Trigger-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{INVENTORY.CONTRACT.NUMBER}</td>
<td>Contract number field in host inventory. This macro may be used with a numeric index e.g. {INVENTORY.CONTRACT.NUMBER&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Trigger-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{INVENTORY.DEPLOYMENT.STATUS}</td>
<td>Deployment status field in host inventory. This macro may be used with a numeric index e.g. {INVENTORY.DEPLOYMENT.STATUS&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Trigger-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>{INVENTORY_HARDWARE}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Hardware field in host inventory.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td>This macro may be used with a numeric index e.g. <code>{INVENTORY_HARDWARE&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td><code>{PROFILE_HARDWARE&lt;1-9&gt;}</code> is deprecated.</td>
</tr>
<tr>
<td><code>{INVENTORY_HARDWARE.FULL}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Hardware (Full details) field in host inventory.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td>This macro may be used with a numeric index e.g. <code>{INVENTORY_HARDWARE.FULL&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY_HOST.NETMASK}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Host subnet mask field in host inventory.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td>This macro may be used with a numeric index e.g. <code>{INVENTORY_HOST.NETMASK&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY_HOST.NETWORKS}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Host networks field in host inventory.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td>This macro may be used with a numeric index e.g. <code>{INVENTORY_HOST.NETWORKS&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY_HOST.ROUTER}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Host router field in host inventory.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td>This macro may be used with a numeric index e.g. <code>{INVENTORY_HOST.ROUTER&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY_HW.ARCH}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Hardware architecture field in host inventory.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td>This macro may be used with a numeric index e.g. <code>{INVENTORY_HW.ARCH&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>{INVENTORY.HW.DATE.DECOMM}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Date hardware decommissioned field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.HW.DATE.DECOMM&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td><code>{INVENTORY.HW.DATE.EXPIRY}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Date hardware maintenance expires field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.HW.DATE.EXPIRY&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td><code>{INVENTORY.HW.DATE.INSTALL}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Date hardware installed field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.HW.DATE.INSTALL&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td><code>{INVENTORY.HW.DATE.PURCHASE}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Date hardware purchased field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.HW.DATE.PURCHASE&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td><code>{INVENTORY.INSTALLER.NAME}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Installer name field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.INSTALLER.NAME&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td><code>{INVENTORY.LOCATION}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Location field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.LOCATION&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td><code>{PROFILE.LOCATION&lt;1-9&gt;}</code></td>
<td></td>
<td>{PROFILE.LOCATION&lt;1-9&gt;} is deprecated.</td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| {INVENTORY.LOCATION.LAT} | Trigger-based notifications and commands  
→ Problem update notifications and commands  
→ Internal notifications  
→ Tag names and values  
→ Map element labels, map URL names and values  
→ Script-type items  
→ Manual host action scripts  
→ Manual event action scripts  
→ Description of item value widget | Location latitude field in host inventory. This macro may be used with a numeric index e.g. {INVENTORY.LOCATION.LAT<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. |
| {INVENTORY.LOCATION.LON} | Trigger-based notifications and commands  
→ Problem update notifications and commands  
→ Internal notifications  
→ Tag names and values  
→ Map element labels, map URL names and values  
→ Script-type items  
→ Manual host action scripts  
→ Manual event action scripts  
→ Description of item value widget | Location longitude field in host inventory. This macro may be used with a numeric index e.g. {INVENTORY.LOCATION.LON<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. |
| {INVENTORY.MACADDRESS.A} | Trigger-based notifications and commands  
→ Problem update notifications and commands  
→ Internal notifications  
→ Tag names and values  
→ Map element labels, map URL names and values  
→ Script-type items  
→ Manual host action scripts  
→ Manual event action scripts  
→ Description of item value widget | MAC address A field in host inventory. This macro may be used with a numeric index e.g. {INVENTORY.MACADDRESS.A<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. {PROFILE.MACADDRESS<1-9>} is deprecated. |
| {INVENTORY.MACADDRESS.B} | Trigger-based notifications and commands  
→ Problem update notifications and commands  
→ Internal notifications  
→ Tag names and values  
→ Map element labels, map URL names and values  
→ Script-type items  
→ Manual host action scripts  
→ Manual event action scripts  
→ Description of item value widget | MAC address B field in host inventory. This macro may be used with a numeric index e.g. {INVENTORY.MACADDRESS.B<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. |
| {INVENTORY.MODEL} | Trigger-based notifications and commands  
→ Problem update notifications and commands  
→ Internal notifications  
→ Tag names and values  
→ Map element labels, map URL names and values  
→ Script-type items  
→ Manual host action scripts  
→ Manual event action scripts  
→ Description of item value widget | Model field in host inventory. This macro may be used with a numeric index e.g. {INVENTORY.MODEL<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. |
| {INVENTORY.NAME} | Trigger-based notifications and commands  
→ Problem update notifications and commands  
→ Internal notifications  
→ Tag names and values  
→ Map element labels, map URL names and values  
→ Script-type items  
→ Manual host action scripts  
→ Manual event action scripts  
→ Description of item value widget | Name field in host inventory. This macro may be used with a numeric index e.g. {INVENTORY.NAME<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. {PROFILE.NAME<1-9>} is deprecated. |
<table>
<thead>
<tr>
<th>Macro</th>
<th>Supported in</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{INVENTORY.NOTES}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Notes field in host inventory.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY.OOB.IP}</code></td>
<td>Trigger-based notifications and commands</td>
<td>OOB IP address field in host inventory.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY.OOB.NETMASK}</code></td>
<td>Trigger-based notifications and commands</td>
<td>OOB subnet mask field in host inventory.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY.OOB.ROUTER}</code></td>
<td>Trigger-based notifications and commands</td>
<td>OOB router field in host inventory.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY.OS}</code></td>
<td>Trigger-based notifications and commands</td>
<td>OS field in host inventory.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY.OS.FULL}</code></td>
<td>Trigger-based notifications and commands</td>
<td>OS (Full details) field in host inventory.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
</tbody>
</table>

This macro may be used with a numeric index e.g. `{INVENTORY.NOTES<1-9>}` to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

{PROFILE.NOTES<1-9>}, {INVENTORY.OOB.IP<1-9>}, {INVENTORY.OOB.NETMASK<1-9>}, {INVENTORY.OOB.ROUTER<1-9>}, {INVENTORY.OS<1-9>}, {INVENTORY.OS.FULL<1-9>} are deprecated.
<table>
<thead>
<tr>
<th>Macro</th>
<th>Supported in</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{INVENTORY.OS.SHORT}</code></td>
<td>Trigger-based notifications and commands</td>
<td>OS (Short) field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.OS.SHORT&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY.POC.PRIMARY.CELL}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Primary POC cell field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.POC.PRIMARY.CELL&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY.POC.PRIMARY.EMAIL}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Primary POC email field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.POC.PRIMARY.EMAIL&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY.POC.PRIMARY.NAME}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Primary POC name field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.POC.PRIMARY.NAME&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY.POC.PRIMARY.NOTES}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Primary POC notes field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.POC.PRIMARY.NOTES&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td><code>{INVENTORY.POC.PRIMARY.PHONE.A}</code></td>
<td>Trigger-based notifications and commands</td>
<td>Primary POC phone A field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.POC.PRIMARY.PHONE.A&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| `{INVENTORY.POC.PRIMARY.PHONE.B}` | → Problem update notifications and commands  
→ Internal notifications  
→ Tag names and values  
→ Map element labels, map URL names and values  
→ Script-type items  
→ Manual host action scripts  
→ Manual event action scripts  
→ Description of item value widget | Primary POC phone B field in host inventory.  
This macro may be used with a numeric index e.g. `{INVENTORY.POC.PRIMARY.PHONE.B<1-9>}` to point to the first, second, third, etc. host in a trigger expression. See indexed macros. |
| `{INVENTORY.POC.PRIMARY.SCREEN}` | → Problem update notifications and commands  
→ Internal notifications  
→ Tag names and values  
→ Map element labels, map URL names and values  
→ Script-type items  
→ Manual host action scripts  
→ Manual event action scripts  
→ Description of item value widget | Primary POC screen name field in host inventory.  
This macro may be used with a numeric index e.g. `{INVENTORY.POC.PRIMARY.SCREEN<1-9>}` to point to the first, second, third, etc. host in a trigger expression. See indexed macros. |
| `{INVENTORY.POC.SECONDARY.CELL}` | → Problem update notifications and commands  
→ Internal notifications  
→ Tag names and values  
→ Map element labels, map URL names and values  
→ Script-type items  
→ Manual host action scripts  
→ Manual event action scripts  
→ Description of item value widget | Secondary POC cell field in host inventory.  
This macro may be used with a numeric index e.g. `{INVENTORY.POC.SECONDARY.CELL<1-9>}` to point to the first, second, third, etc. host in a trigger expression. See indexed macros. |
| `{INVENTORY.POC.SECONDARY.EMAIL}` | → Problem update notifications and commands  
→ Internal notifications  
→ Tag names and values  
→ Map element labels, map URL names and values  
→ Script-type items  
→ Manual host action scripts  
→ Manual event action scripts  
→ Description of item value widget | Secondary POC email field in host inventory.  
This macro may be used with a numeric index e.g. `{INVENTORY.POC.SECONDARY.EMAIL<1-9>}` to point to the first, second, third, etc. host in a trigger expression. See indexed macros. |
| `{INVENTORY.POC.SECONDARY.NAME}` | → Problem update notifications and commands  
→ Internal notifications  
→ Tag names and values  
→ Map element labels, map URL names and values  
→ Script-type items  
→ Manual host action scripts  
→ Manual event action scripts  
→ Description of item value widget | Secondary POC name field in host inventory.  
This macro may be used with a numeric index e.g. `{INVENTORY.POC.SECONDARY.NAME<1-9>}` to point to the first, second, third, etc. host in a trigger expression. See indexed macros. |
| `{INVENTORY.POC.SECONDARY.NOTES}` | → Problem update notifications and commands  
→ Internal notifications  
→ Tag names and values  
→ Map element labels, map URL names and values  
→ Script-type items  
→ Manual host action scripts  
→ Manual event action scripts  
→ Description of item value widget | Secondary POC notes field in host inventory.  
This macro may be used with a numeric index e.g. `{INVENTORY.POC.SECONDARY.NOTES<1-9>}` to point to the first, second, third, etc. host in a trigger expression. See indexed macros. |
<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{INVENTORY.POC.SECONDARY.PHONE.A} → Trigger-based notifications and commands</td>
<td>Secondary POC phone A field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.POC.SECONDARY.PHONE.A&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td>Internal notifications</td>
</tr>
<tr>
<td></td>
<td>Tag names and values</td>
</tr>
<tr>
<td></td>
<td>Map element labels, map URL names and values</td>
</tr>
<tr>
<td></td>
<td>Script-type items⁶</td>
</tr>
<tr>
<td></td>
<td>Manual host action scripts⁶</td>
</tr>
<tr>
<td></td>
<td>Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td>Description of item value widget</td>
</tr>
<tr>
<td>{INVENTORY.POC.SECONDARY.PHONE.B} → Trigger-based notifications and commands</td>
<td>Secondary POC phone B field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.POC.SECONDARY.PHONE.B&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td>Internal notifications</td>
</tr>
<tr>
<td></td>
<td>Tag names and values</td>
</tr>
<tr>
<td></td>
<td>Map element labels, map URL names and values</td>
</tr>
<tr>
<td></td>
<td>Script-type items⁶</td>
</tr>
<tr>
<td></td>
<td>Manual host action scripts⁶</td>
</tr>
<tr>
<td></td>
<td>Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td>Description of item value widget</td>
</tr>
<tr>
<td>{INVENTORY.POC.SECONDARY.SCREEN} → Trigger-based notifications and commands</td>
<td>Secondary POC screen name field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.POC.SECONDARY.SCREEN&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td>Internal notifications</td>
</tr>
<tr>
<td></td>
<td>Tag names and values</td>
</tr>
<tr>
<td></td>
<td>Map element labels, map URL names and values</td>
</tr>
<tr>
<td></td>
<td>Script-type items⁶</td>
</tr>
<tr>
<td></td>
<td>Manual host action scripts⁶</td>
</tr>
<tr>
<td></td>
<td>Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td>Description of item value widget</td>
</tr>
<tr>
<td>{INVENTORY.SERIALNO.A} → Trigger-based notifications and commands</td>
<td>Serial number A field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.SERIALNO.A&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td>Internal notifications</td>
</tr>
<tr>
<td></td>
<td>Tag names and values</td>
</tr>
<tr>
<td></td>
<td>Map element labels, map URL names and values</td>
</tr>
<tr>
<td></td>
<td>Script-type items⁶</td>
</tr>
<tr>
<td></td>
<td>Manual host action scripts⁶</td>
</tr>
<tr>
<td></td>
<td>Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td>Description of item value widget</td>
</tr>
<tr>
<td>{PROFILE.SERIALNO&lt;1-9&gt;}</td>
<td>is deprecated.</td>
</tr>
<tr>
<td>{INVENTORY.SERIALNO.B} → Trigger-based notifications and commands</td>
<td>Serial number B field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.SERIALNO.B&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td>Internal notifications</td>
</tr>
<tr>
<td></td>
<td>Tag names and values</td>
</tr>
<tr>
<td></td>
<td>Map element labels, map URL names and values</td>
</tr>
<tr>
<td></td>
<td>Script-type items⁶</td>
</tr>
<tr>
<td></td>
<td>Manual host action scripts⁶</td>
</tr>
<tr>
<td></td>
<td>Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td>Description of item value widget</td>
</tr>
<tr>
<td>{INVENTORY.SITE.ADDRESS.A} → Trigger-based notifications and commands</td>
<td>Site address A field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.SITE.ADDRESS.A&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>Problem update notifications and commands</td>
</tr>
<tr>
<td></td>
<td>Internal notifications</td>
</tr>
<tr>
<td></td>
<td>Tag names and values</td>
</tr>
<tr>
<td></td>
<td>Map element labels, map URL names and values</td>
</tr>
<tr>
<td></td>
<td>Script-type items⁶</td>
</tr>
<tr>
<td></td>
<td>Manual host action scripts⁶</td>
</tr>
<tr>
<td></td>
<td>Manual event action scripts</td>
</tr>
<tr>
<td></td>
<td>Description of item value widget</td>
</tr>
<tr>
<td>Macro</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>{INVENTORY.SITE.ADDRESS.B}</code></td>
<td>Site address B field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.SITE.ADDRESS.B&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td><code>{INVENTORY.SITE.ADDRESS.C}</code></td>
<td>Site address C field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.SITE.ADDRESS.C&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td><code>{INVENTORY.SITE.CITY}</code></td>
<td>Site city field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.SITE.CITY&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td><code>{INVENTORY.SITE.COUNTRY}</code></td>
<td>Site country field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.SITE.COUNTRY&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td><code>{INVENTORY.SITE.NOTES}</code></td>
<td>Site notes field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.SITE.NOTES&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td><code>{INVENTORY.SITE.RACK}</code></td>
<td>Site rack location field in host inventory. This macro may be used with a numeric index e.g. <code>{INVENTORY.SITE.RACK&lt;1-9&gt;}</code> to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
</tr>
<tr>
<td>{INVENTORY.SITE.STATE}</td>
<td>Trigger-based notifications and commands</td>
</tr>
<tr>
<td>{INVENTORY.SITE.ZIP}</td>
<td>Trigger-based notifications and commands</td>
</tr>
<tr>
<td>{INVENTORY.SOFTWARE}</td>
<td>Trigger-based notifications and commands</td>
</tr>
<tr>
<td>{INVENTORY.SOFTWARE.APP.A}</td>
<td>Trigger-based notifications and commands</td>
</tr>
<tr>
<td>{INVENTORY.SOFTWARE.APP.B}</td>
<td>Trigger-based notifications and commands</td>
</tr>
<tr>
<td>{INVENTORY.SOFTWARE.APP.C}</td>
<td>Trigger-based notifications and commands</td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>{INVENTORY.SOFTWARE.APP.D}</td>
<td></td>
</tr>
<tr>
<td>{INVENTORY.SOFTWARE.APP.E}</td>
<td></td>
</tr>
<tr>
<td>{INVENTORY.SOFTWARE.FULL}</td>
<td></td>
</tr>
<tr>
<td>{INVENTORY.TAG}</td>
<td></td>
</tr>
<tr>
<td>{INVENTORY.TYPE}</td>
<td></td>
</tr>
<tr>
<td>{INVENTORY.TYPE.FULL}</td>
<td></td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>{INVENTORY.URL.A} → Trigger-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{INVENTORY.URL.B} → Trigger-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{INVENTORY.URL.C} → Trigger-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{INVENTORY.VENDOR} → Trigger-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td>→ Tag names and values</td>
<td></td>
</tr>
<tr>
<td>→ Map element labels, map URL names and values</td>
<td></td>
</tr>
<tr>
<td>→ Script-type items</td>
<td></td>
</tr>
<tr>
<td>→ Manual host action scripts</td>
<td></td>
</tr>
<tr>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{ITEM.DESCRIPTION} → Trigger-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{ITEM.DESCRIPTION.ORIG} → Trigger-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td>→ Description of item value widget</td>
<td></td>
</tr>
</tbody>
</table>

Supported since 5.2.0.
<table>
<thead>
<tr>
<th>Macro</th>
<th>Supported in</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ITEM.ID}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Numeric ID of the Nth item in the trigger expression that caused a notification. This macro may be used with a numeric index e.g. {ITEM.ID&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td>{ITEM.KEY}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Key of the Nth item in the trigger expression that caused a notification. This macro may be used with a numeric index e.g. {ITEM.KEY&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. {TRIGGER.KEY} is deprecated.</td>
</tr>
<tr>
<td>{ITEM.KEY.ORIG}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Original key (with macros not expanded) of the Nth item in the trigger expression that caused a notification. This macro may be used with a numeric index e.g. {ITEM.KEY.ORIG&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td>{ITEM.LASTVALUE}</td>
<td>→ Trigger-based notifications and commands</td>
<td>The latest value of the Nth item in the trigger expression that caused a notification. It will resolve to <em>UNKNOWN</em> in the frontend if the latest history value has been collected more than the Max history display period time ago (set in the Administration→General menu section). Note that since 4.0, when used in the problem name, it will not resolve to the latest item value when viewing problem events, instead it will keep the item value from the time of problem happening. It is alias to last (/{HOST.HOST}/{ITEM.KEY}). Customizing the macro value is supported for this macro; starting with Zabbix 3.2.0.</td>
</tr>
<tr>
<td>{ITEM.LOG.AGE}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Age of the log item event, with precision down to a second. This macro may be used with a numeric index e.g. {ITEM.LOG.AGE&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>{ITEM.LOG.DATE}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Date of the log item event.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger names, operational data and descriptions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger URLs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Event tags and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{ITEM.LOG.EVENTID}</td>
<td>→ Problem update notifications and commands</td>
<td>ID of the event in the event log.</td>
</tr>
<tr>
<td></td>
<td>→ Trigger names, operational data and descriptions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger URLs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Event tags and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{ITEM.LOG.NSEVERITY}</td>
<td>→ Problem update notifications and commands</td>
<td>Numeric severity of the event in the event log.</td>
</tr>
<tr>
<td></td>
<td>→ Trigger names, operational data and descriptions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger URLs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Event tags and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{ITEM.LOG.SEVERITY}</td>
<td>→ Problem update notifications and commands</td>
<td>Verbal severity of the event in the event log.</td>
</tr>
<tr>
<td></td>
<td>→ Trigger names, operational data and descriptions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger URLs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Event tags and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{ITEM.LOG.SOURCE}</td>
<td>→ Problem update notifications and commands</td>
<td>Source of the event in the event log.</td>
</tr>
<tr>
<td></td>
<td>→ Trigger names, operational data and descriptions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger URLs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Event tags and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{ITEM.LOG.TIME}</td>
<td>→ Problem update notifications and commands</td>
<td>Time of the log item event.</td>
</tr>
<tr>
<td></td>
<td>→ Trigger names, operational data and descriptions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger URLs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Event tags and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{ITEM.NAME}</td>
<td>→ Problem update notifications and commands</td>
<td>Name of the Nth item in the trigger expression that caused a notification.</td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Description of item value widget</td>
<td></td>
</tr>
</tbody>
</table>

---

This macro may be used with a numeric index e.g. 
{ITEM.LOG.DATE<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

For Windows event log monitoring only.

This macro may be used with a numeric index e.g. 
{ITEM.LOG.NSEVERITY<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

For Windows event log monitoring only.

This macro may be used with a numeric index e.g. 
{ITEM.LOG.SOURCE<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

For Windows event log monitoring only.
### Macro Description

<table>
<thead>
<tr>
<th>Macro</th>
<th>Supported in</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ITEM.NAME.ORIG}</td>
<td>Trigger-based notifications and commands</td>
<td>This macro is deprecated since Zabbix 6.0. It used to resolve to the original name (i.e. without macros resolved) of the item in pre-6.0 Zabbix versions when user macros and positional macros were supported in the item name.</td>
</tr>
<tr>
<td></td>
<td>Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{ITEM.STATE}</td>
<td>Item-based internal notifications</td>
<td>This macro may be used with a numeric index e.g. {ITEM.STATE&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{ITEM.STATE.ERROR}</td>
<td>Item-based internal notifications</td>
<td>Error message with details why an item became unsupported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If an item goes into the unsupported state and then immediately gets supported again the error field can be empty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resolved to either:</td>
</tr>
<tr>
<td></td>
<td>1) the historical (at-the-time-of-event) value of the Nth item in the trigger expression, if used in the context of trigger status change, for example, when displaying events or sending notifications.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2) the latest value of the Nth item in the trigger expression, if used without the context of trigger status change, for example, when displaying a list of triggers in a pop-up selection window. In this case works the same as {ITEM.LASTVALUE}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>In the first case it will resolve to <em>UNKNOWN</em> if the history value has already been deleted or has never been stored.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In the second case, and in the frontend only, it will resolve to <em>UNKNOWN</em> if the latest history value has been collected more than the Max history display period time ago (set in the Administration→General menu section).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Customizing</strong> the macro value is supported for this macro, starting with Zabbix 3.2.0.</td>
</tr>
<tr>
<td>{ITEM.VALUE}</td>
<td>Trigger-based notifications and commands</td>
<td>This macro may be used with a numeric index e.g. {ITEM.VALUE&lt;1-9&gt;} to point to the first, second, third, etc. item in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Trigger names, event names, operational data and descriptions</td>
<td>Value type of the Nth item in the trigger expression that caused a notification. Possible values: 0 - numeric float, 1 - character, 2 - log, 3 - numeric unsigned, 4 - text.</td>
</tr>
<tr>
<td></td>
<td>Tag names and values</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Trigger URLs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Description of item value widget</td>
<td></td>
</tr>
<tr>
<td>{ITEM.VALUETYPE}</td>
<td>Trigger-based notifications and commands</td>
<td>This macro may be used with a numeric index e.g. {ITEM.VALUETYPE&lt;1-9&gt;} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.</td>
</tr>
<tr>
<td></td>
<td>Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Description of item value widget</td>
<td></td>
</tr>
</tbody>
</table>

Customizing the macro value is supported for this macro, starting with Zabbix 3.2.0.

Supported since 5.4.0.
Macro Support in Description

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{LLDRULE.DESCRIPTION} → LLD-rule based internal notifications</td>
<td>Description of the low-level discovery rule which caused a notification.</td>
</tr>
<tr>
<td>{LLDRULE.DESCRIPTION.ORIG} → LLD-rule based internal notifications</td>
<td>Description (with macros unresolved) of the low-level discovery rule which caused a notification. Supported since 5.2.0.</td>
</tr>
<tr>
<td>{LLDRULE.ID} → LLD-rule based internal notifications</td>
<td>Numeric ID of the low-level discovery rule which caused a notification.</td>
</tr>
<tr>
<td>{LLDRULE.KEY} → LLD-rule based internal notifications</td>
<td>Key of the low-level discovery rule which caused a notification.</td>
</tr>
<tr>
<td>{LLDRULE.KEY.ORIG} → LLD-rule based internal notifications</td>
<td>Original key (with macros not expanded) of the low-level discovery rule which caused a notification.</td>
</tr>
<tr>
<td>{LLDRULE.NAME} → LLD-rule based internal notifications</td>
<td>Name of the low-level discovery rule (with macros resolved) that caused a notification.</td>
</tr>
<tr>
<td>{LLDRULE.NAME.ORIG} → LLD-rule based internal notifications</td>
<td>Original name (i.e. without macros resolved) of the low-level discovery rule that caused a notification.</td>
</tr>
<tr>
<td>{LLDRULE.STATE} → LLD-rule based internal notifications</td>
<td>The latest state of the low-level discovery rule. Possible values: Not supported and Normal. Error message: Not supported and Normal.</td>
</tr>
<tr>
<td>{LLDRULE.STATE.ERROR} → LLD-rule based internal notifications</td>
<td>Errormessage with details why an LLD rule became unsupported.</td>
</tr>
</tbody>
</table>

If an LLD rule goes into the unsupported state and then immediately gets supported again the error field can be empty.

{MAP.ID} → Map element labels, map URL names and values
{MAP.NAME} → Map element labels, map URL names and values → Text field in map shapes

Network map ID.
Network map name.
Supported since 3.4.0.

Description of the proxy. Resolves to either:
1) proxy of the Nth item in the trigger expression (in trigger-based notifications). You may use indexed macros here.
2) proxy, which executed discovery (in discovery notifications). Use {PROXY.DESCRIPTION} here, without indexing.
3) proxy to which an active agent registered (in autoregistration notifications). Use {PROXY.DESCRIPTION} here, without indexing.

This macro may be used with a numeric index e.g. {PROXY.DESCRIPTION<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

{PROXY.NAME} → Trigger-based notifications and commands
{PROXY.NAME} → Problem update notifications and commands
{PROXY.NAME} → Discovery notifications and commands
{PROXY.NAME} → Autoregistration notifications and commands
{PROXY.NAME} → Internal notifications
{PROXY.NAME} → Manual event action scripts

Name of the proxy. Resolves to either:
1) proxy of the Nth item in the trigger expression (in trigger-based notifications). You may use indexed macros here.
2) proxy, which executed discovery (in discovery notifications). Use {PROXY.NAME} here, without indexing.
3) proxy to which an active agent registered (in autoregistration notifications). Use {PROXY.NAME} here, without indexing.

This macro may be used with a numeric index e.g. {PROXY.NAME<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

{SERVICE.DESCRIPTION} → Service-based notifications and commands
{SERVICE.DESCRIPTION} → Service update notifications and commands

Description of the service (with macros resolved).
<table>
<thead>
<tr>
<th>Macro</th>
<th>Supported in</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{SERVICE.NAME}</td>
<td>→ Service-based notifications and commands</td>
<td>Name of the service (with macros resolved).</td>
</tr>
<tr>
<td></td>
<td>→ Service update notifications and commands</td>
<td></td>
</tr>
<tr>
<td>{SERVICE.ROOTCAUSE}</td>
<td>→ Service-based notifications and commands</td>
<td>List of trigger problem events that caused a service to fail, sorted by severity and host name. Includes the following details: host name, event name, severity, age, service tags and values.</td>
</tr>
<tr>
<td></td>
<td>→ Service update notifications and commands</td>
<td></td>
</tr>
<tr>
<td>{SERVICE.TAGS}</td>
<td>→ Service-based notifications and commands</td>
<td>A comma separated list of service event tags.</td>
</tr>
<tr>
<td></td>
<td>→ Service update notifications and commands</td>
<td>Service event tags can be defined in the service configuration section Tags. Expanded to an empty string if no tags exist.</td>
</tr>
<tr>
<td>{SERVICE.TAGSJSON}</td>
<td>→ Service-based notifications and commands</td>
<td>A JSON array containing service event tag objects.</td>
</tr>
<tr>
<td></td>
<td>→ Service update notifications and commands</td>
<td>Service event tags can be defined in the service configuration section Tags. Expanded to an empty array if no tags exist.</td>
</tr>
<tr>
<td></td>
<td>→ Discovery notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Autoregistration notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger event names</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td>{TIME}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Current time in hh:mm:ss.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Service-based notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Service update notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Discovery notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Autoregistration notifications and commands</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Trigger event names</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td>{TRIGGER.DESCRIPTION}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Trigger description.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td>All macros supported in a trigger description will be expanded if {TRIGGER.DESCRIPTION} is used in notification text.</td>
</tr>
<tr>
<td></td>
<td>→ Trigger-based internal notifications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td>{TRIGGER.COMMENT} is deprecated.</td>
</tr>
<tr>
<td>{TRIGGER.EXPRESSION.EXPLAIN}</td>
<td>→ Trigger-based notifications and commands</td>
<td>Partially evaluated trigger expression.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td>Item-based functions are evaluated and replaced by the results at the time of event generation whereas all other functions are displayed as written in the expression. Can be used for debugging trigger expressions.</td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Event names</td>
<td></td>
</tr>
<tr>
<td>{TRIGGER.EXPRESSION.RECOVERY.EXPLAIN}</td>
<td>→ Problem-based notifications and commands</td>
<td>Partially evaluated trigger recovery expression.</td>
</tr>
<tr>
<td></td>
<td>→ Problem update notifications and commands</td>
<td>Item-based functions are evaluated and replaced by the results at the time of event generation whereas all other functions are displayed as written in the expression. Can be used for debugging trigger recovery expressions.</td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td>{TRIGGER.EVENTS.ACK}</td>
<td>→ Problem update notifications and commands</td>
<td>Number of acknowledged events for a map element in maps, or for the trigger which generated current event in notifications.</td>
</tr>
<tr>
<td></td>
<td>→ Map element labels</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td>{TRIGGER.EVENTS.PROBLEM.ACK}</td>
<td>→ Problem update notifications and commands</td>
<td>Number of acknowledged PROBLEM events for all triggers disregarding their state.</td>
</tr>
<tr>
<td></td>
<td>→ Map element labels</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td>{TRIGGER.EVENTS.PROBLEM.UNACK}</td>
<td>→ Problem update notifications and commands</td>
<td>Number of unacknowledged PROBLEM events for all triggers disregarding their state.</td>
</tr>
<tr>
<td></td>
<td>→ Map element labels</td>
<td></td>
</tr>
<tr>
<td></td>
<td>→ Manual event action scripts</td>
<td></td>
</tr>
<tr>
<td>Macro</td>
<td>Supported in</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>{TRIGGER.EVENTS.UNACK}</code></td>
<td>Trigger-based notifications and commands → Problem update notifications and commands → Map element labels → Manual event action scripts</td>
<td>Number of unacknowledged events for a map element in maps, or for the trigger which generated current event in notifications.</td>
</tr>
<tr>
<td><code>{TRIGGER.HOSTGROUP.NAME}</code></td>
<td>Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Manual event action scripts</td>
<td>A sorted (by SQL query), comma-space separated list of host groups in which the trigger is defined.</td>
</tr>
<tr>
<td><code>{TRIGGER.PROBLEM.EVENTS.PROBLEM.ACK}</code></td>
<td>Number of acknowledged PROBLEM events for triggers in PROBLEM state.</td>
<td></td>
</tr>
<tr>
<td><code>{TRIGGER.PROBLEM.EVENTS.PROBLEM.UNACK}</code></td>
<td>Number of unacknowledged PROBLEM events for triggers in PROBLEM state.</td>
<td></td>
</tr>
<tr>
<td><code>{TRIGGER.EXPRESSION}</code></td>
<td>Trigger expression.</td>
<td></td>
</tr>
<tr>
<td><code>{TRIGGER.EXPRESSION.RECOVERY}</code></td>
<td>Trigger recovery expression if OK event generation in trigger configuration is set to ‘Recovery expression’; otherwise an empty string is returned. Supported since 3.2.0.</td>
<td></td>
</tr>
<tr>
<td><code>{TRIGGER.ID}</code></td>
<td>Numeric trigger ID which triggered this action. Supported in trigger tag values since 4.4.1.</td>
<td></td>
</tr>
<tr>
<td><code>{TRIGGER.NAME}</code></td>
<td>Name of the trigger (with macros resolved). Note that since 4.0.0 <code>{EVENT.NAME}</code> can be used in actions to display the triggered event/problem name with macros resolved.</td>
<td></td>
</tr>
<tr>
<td><code>{TRIGGER.NAME.ORIG}</code></td>
<td>Original name of the trigger (i.e. without macros resolved).</td>
<td></td>
</tr>
<tr>
<td><code>{TRIGGER.NSEVERITY}</code></td>
<td>Numerical trigger severity. Possible values: 0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster.</td>
<td></td>
</tr>
<tr>
<td><code>{TRIGGER.SEVERITY}</code></td>
<td>Trigger severity name. Can be defined in Administration → General → Trigger displaying options.</td>
<td></td>
</tr>
<tr>
<td><code>{TRIGGER.STATE}</code></td>
<td>The latest state of the trigger. Possible values: Unknown and Normal. Error message with details why a trigger became unsupported.</td>
<td></td>
</tr>
<tr>
<td><code>{TRIGGER.STATUS}</code></td>
<td>Trigger value at the time of operation step execution. Can be either PROBLEM or OK. <code>{STATUS}</code> is deprecated.</td>
<td></td>
</tr>
<tr>
<td><code>{TRIGGER.TEMPLATE.NAME}</code></td>
<td>A sorted (by SQL query), comma-space separated list of templates in which the trigger is defined, or &quot;UNKNOWN&quot; if the trigger is defined in a host.</td>
<td></td>
</tr>
</tbody>
</table>

1545
<table>
<thead>
<tr>
<th>Macro</th>
<th>Supported in</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{TRIGGER.URL}</td>
<td></td>
<td>Trigger URL.</td>
</tr>
<tr>
<td>{TRIGGER.VALUE}</td>
<td></td>
<td>Current trigger numeric value: 0 - trigger is in OK state, 1 - trigger is in PROBLEM state.</td>
</tr>
<tr>
<td>{TRIGGERS.UNACK}</td>
<td></td>
<td>Number of unacknowledged triggers for a map element, disregarding trigger state. A trigger is considered to be unacknowledged if at least one of its PROBLEM events is unacknowledged.</td>
</tr>
<tr>
<td>{TRIGGERS.PROBLEM.UNACK}</td>
<td></td>
<td>Number of unacknowledged PROBLEM triggers for a map element. A trigger is considered to be unacknowledged if at least one of its PROBLEM events is unacknowledged.</td>
</tr>
<tr>
<td>{TRIGGERS.ACK}</td>
<td></td>
<td>Number of acknowledged triggers for a map element, disregarding trigger state. A trigger is considered to be acknowledged if all of it's PROBLEM events are acknowledged.</td>
</tr>
<tr>
<td>{TRIGGERS.PROBLEM.ACK}</td>
<td></td>
<td>Number of acknowledged PROBLEM triggers for a map element. A trigger is considered to be acknowledged if all of it’s PROBLEM events are acknowledged.</td>
</tr>
<tr>
<td>{USER.FULLNAME}</td>
<td></td>
<td>Name, surname and username of the user who added event acknowledgment or started the script. Supported for problem updates since 3.4.0, for global scripts since 5.0.2</td>
</tr>
<tr>
<td>{USER.NAME}</td>
<td></td>
<td>Name of the user who started the script. Supported since 5.0.2.</td>
</tr>
<tr>
<td>{USER.SURNAME}</td>
<td></td>
<td>Surname of the user who started the script. Supported since 5.0.2.</td>
</tr>
<tr>
<td>{USER.USERNAME}</td>
<td></td>
<td>Username of the user who started the script. Supported since 5.0.2. {USER.ALIAS}, supported before Zabbix 5.4.0, is now deprecated.</td>
</tr>
<tr>
<td>{$MACRO}</td>
<td></td>
<td>See: User macros supported by location</td>
</tr>
<tr>
<td>{#MACRO}</td>
<td></td>
<td>See: Low-level discovery macros</td>
</tr>
<tr>
<td>{?EXPRESSION}</td>
<td></td>
<td>Customizing the macro value is supported for this macro, starting with Zabbix 4.0.0. See expression macros. Supported since 5.2.0.</td>
</tr>
</tbody>
</table>

Footnotes

1 The {HOST.*} macros supported in item key parameters will resolve to the interface that is selected for the item. When used in items without interfaces they will resolve to either the Zabbix agent, SNMP, JMX or IPMI interface of the host in this order of priority or to 'UNKNOWN' if the host does not have any interface.
In global scripts, interface IP/DNS fields and web scenarios the macro will resolve to the main agent interface, however, if it is not present, the main SNMP interface will be used. If SNMP is also not present, the main JMX interface will be used. If JMX is not present either, the main IPMI interface will be used. If the host does not have any interface, the macro resolves to ‘UNKNOWN’.

Only the \texttt{avg}, \texttt{last}, \texttt{max} and \texttt{min} functions, with seconds as parameter are supported in this macro in map labels.

\{HOST.*\} macros are supported in web scenario Variables, Headers, SSL certificate file and SSL key file fields and in scenario step URL, Post, Headers and Required string fields. Since Zabbix 5.4.0, \{HOST.*\} macros are no longer supported in web scenario Name and web scenario step Name fields.

Only the \texttt{avg}, \texttt{last}, \texttt{max} and \texttt{min} functions, with seconds as parameter are supported within this macro in graph names. The \{HOST.HOST<1-9>\} macro can be used as host within the macro. For example:
- \texttt{last(/Cisco switch/ifAlias[#{SNMPINDEX}])}
- \texttt{last(/\{HOST.HOST\}/ifAlias[#{SNMPINDEX}])}

Only the \texttt{avg}, \texttt{last}, \texttt{max} and \texttt{min} functions, with seconds as parameter are supported within this macro in map labels.

Indexed macros

The indexed macro syntax of \{MACRO<1-9>\} works only in the context of \textit{trigger expressions}. It can be used to reference hosts or functions in the order in which they appear in the expression. Macros like \{HOST.IP1\}, \{HOST.IP2\}, \{HOST.IP3\} will resolve to the IP of the first, second, and third host in the trigger expression (providing the trigger expression contains those hosts). Macros like \{FUNCTION.VALUE1\}, \{FUNCTION.VALUE2\}, \{FUNCTION.VALUE3\} will resolve to the value of the first, second, and third item-based function in the trigger expression at the time of the event (providing the trigger expression contains those functions).

Additionally the \{HOST.HOST<1-9>\} macro is also supported within the \{?func(/host/key,param)\} expression macro in \textit{graph names}. For example, \{?func(/\{HOST.HOST2\}/key,param)\} in the graph name will refer to the host of the second item in the graph.

Indexed macros will not resolve in any other context, except the two cases mentioned here. For other contexts, use macros \textbf{without} index (i.e. \{HOST.HOST\}, \{HOST.IP\}, etc) instead.

\section*{2 User macros supported by location}

Overview

This section contains a list of locations, where \textit{user-definable} macros are supported.

Only global-level user macros are supported for Actions, Network discovery, Proxies and all locations listed under Other locations section of this page. In the mentioned locations, host-level and template-level macros will not be resolved.

Actions

In \textbf{actions}, user macros can be used in the following fields:

<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger-based notifications and commands</td>
<td>yes</td>
</tr>
<tr>
<td>Trigger-based internal notifications</td>
<td>yes</td>
</tr>
<tr>
<td>Problem update notifications</td>
<td>yes</td>
</tr>
<tr>
<td>Service-based notifications and commands</td>
<td>yes</td>
</tr>
<tr>
<td>Service update notifications</td>
<td>yes</td>
</tr>
<tr>
<td>Time period condition</td>
<td>no</td>
</tr>
<tr>
<td>Operations</td>
<td></td>
</tr>
<tr>
<td>Default operation step duration</td>
<td>no</td>
</tr>
<tr>
<td>Step duration</td>
<td>no</td>
</tr>
</tbody>
</table>

Hosts/host prototypes

In a \textbf{host} and \textbf{host prototype} configuration, user macros can be used in the following fields:

<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface IP/DNS</td>
<td>DNS only</td>
</tr>
<tr>
<td>Interface port</td>
<td>no</td>
</tr>
<tr>
<td>SNMP v1, v2</td>
<td>SNMP community yes</td>
</tr>
<tr>
<td>SNMP v3</td>
<td></td>
</tr>
</tbody>
</table>
Items / item prototypes

In an item or an item prototype configuration, user macros can be used in the following fields:

<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context name</td>
<td>yes</td>
</tr>
<tr>
<td>Security name</td>
<td>yes</td>
</tr>
<tr>
<td>Authentication passphrase</td>
<td>yes</td>
</tr>
<tr>
<td>Privacy passphrase</td>
<td>yes</td>
</tr>
<tr>
<td>Username</td>
<td>yes</td>
</tr>
<tr>
<td>Password</td>
<td>yes</td>
</tr>
<tr>
<td>Tag names</td>
<td>yes</td>
</tr>
<tr>
<td>Tag values</td>
<td>yes</td>
</tr>
<tr>
<td>Item key parameters</td>
<td>yes</td>
</tr>
<tr>
<td>Update interval</td>
<td>no</td>
</tr>
<tr>
<td>Custom intervals</td>
<td>no</td>
</tr>
<tr>
<td>History storage period</td>
<td>no</td>
</tr>
<tr>
<td>Trend storage period</td>
<td>no</td>
</tr>
<tr>
<td>Description</td>
<td>yes</td>
</tr>
<tr>
<td>Calculated item</td>
<td>Formula</td>
</tr>
<tr>
<td>Database monitor</td>
<td>Username</td>
</tr>
<tr>
<td>HTTP agent</td>
<td>URL</td>
</tr>
<tr>
<td>HTTP authentication username</td>
<td>yes</td>
</tr>
<tr>
<td>HTTP authentication password</td>
<td>yes</td>
</tr>
<tr>
<td>SSI certificate file</td>
<td>yes</td>
</tr>
<tr>
<td>SSI key file</td>
<td>yes</td>
</tr>
<tr>
<td>SSI key password</td>
<td>yes</td>
</tr>
<tr>
<td>Allowed hosts</td>
<td>yes</td>
</tr>
<tr>
<td>JMX endpoint</td>
<td>JMX agent</td>
</tr>
</tbody>
</table>
### Low-level discovery

In a **low-level discovery rule**, user macros can be used in the following fields:

<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text^1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key parameters</td>
<td>yes</td>
</tr>
<tr>
<td>Update interval</td>
<td>no</td>
</tr>
<tr>
<td>Custom interval</td>
<td>no</td>
</tr>
<tr>
<td>Keep lost resources period</td>
<td>no</td>
</tr>
<tr>
<td>Description</td>
<td>yes</td>
</tr>
<tr>
<td>SNMP agent</td>
<td>SNMP OID</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>SSH agent</td>
<td>Username</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>Public key file</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>Private key file</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>Password</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>Script</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>TELNET agent</td>
<td>Username</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>Password</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>Script</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Zabbix trapper</td>
<td>Allowed hosts</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Database monitor</td>
<td>Username</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>Password</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>SQL query</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>JMX agent</td>
<td>JMX endpoint</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>HTTP agent</td>
<td>URL^3</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>Query fields</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>Timeout</td>
</tr>
<tr>
<td></td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>Request body</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>Headers (names and values)</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>Required status codes</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>HTTP authentication username</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>HTTP authentication password</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Filters</td>
<td>Regular expression</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Overrides</td>
<td>Filters: regular expression</td>
</tr>
<tr>
<td></td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>Operations: update interval (for item prototypes)</td>
</tr>
<tr>
<td></td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>Operations: history storage period (for item prototypes)</td>
</tr>
<tr>
<td></td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>Operations: trend storage period (for item prototypes)</td>
</tr>
<tr>
<td></td>
<td>no</td>
</tr>
</tbody>
</table>

### Network discovery

In a **network discovery rule**, user macros can be used in the following fields:

<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text^1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update interval</td>
<td>no</td>
</tr>
</tbody>
</table>

---

1. Multiple macros/mix with text
2. Tags
3. URL
<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP v1, v2</td>
<td></td>
</tr>
<tr>
<td>SNMP community</td>
<td>yes</td>
</tr>
<tr>
<td>SNMP OID</td>
<td>yes</td>
</tr>
<tr>
<td>SNMP v3</td>
<td></td>
</tr>
<tr>
<td>Context name</td>
<td>yes</td>
</tr>
<tr>
<td>Security name</td>
<td>yes</td>
</tr>
<tr>
<td>Authentication passphrase</td>
<td>yes</td>
</tr>
<tr>
<td>Privacy passphrase</td>
<td>yes</td>
</tr>
<tr>
<td>SNMP OID</td>
<td>yes</td>
</tr>
</tbody>
</table>

Proxies

In a **proxy** configuration, user macros can be used in the following field:

<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface port (for passive proxy)</td>
<td>no</td>
</tr>
</tbody>
</table>

Templates

In a **template** configuration, user macros can be used in the following fields:

<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tags²</td>
<td></td>
</tr>
<tr>
<td>Tag names</td>
<td>yes</td>
</tr>
<tr>
<td>Tag values</td>
<td>yes</td>
</tr>
</tbody>
</table>

Triggers

In a **trigger** configuration, user macros can be used in the following fields:

<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>yes</td>
</tr>
<tr>
<td>Operational data</td>
<td>yes</td>
</tr>
<tr>
<td>Expression (only in constants and function parameters; secret macros are not supported). Description URL³</td>
<td>yes</td>
</tr>
</tbody>
</table>

³ Multiple macros/mix with text
### Location

<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag for matching</td>
<td>yes</td>
</tr>
<tr>
<td>Tags 2</td>
<td></td>
</tr>
<tr>
<td>Tag names</td>
<td>yes</td>
</tr>
<tr>
<td>Tag values</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Web scenario

In a web scenario configuration, user macros can be used in the following fields:

<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>yes</td>
</tr>
<tr>
<td>Update interval</td>
<td>no</td>
</tr>
<tr>
<td>Agent</td>
<td>yes</td>
</tr>
<tr>
<td>HTTP proxy</td>
<td>yes</td>
</tr>
<tr>
<td>Variables (values only)</td>
<td>yes</td>
</tr>
<tr>
<td>Headers (names and values)</td>
<td>yes</td>
</tr>
<tr>
<td>Steps</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>yes</td>
</tr>
<tr>
<td>URL</td>
<td>yes</td>
</tr>
<tr>
<td>Variables (values only)</td>
<td>yes</td>
</tr>
<tr>
<td>Headers (names and values)</td>
<td>yes</td>
</tr>
<tr>
<td>Timeout</td>
<td>no</td>
</tr>
<tr>
<td>Required string</td>
<td>yes</td>
</tr>
<tr>
<td>Required status codes</td>
<td>no</td>
</tr>
</tbody>
</table>

### Authentication

<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>yes</td>
</tr>
<tr>
<td>Password</td>
<td>yes</td>
</tr>
<tr>
<td>SSL certificate</td>
<td>yes</td>
</tr>
<tr>
<td>SSL key file</td>
<td>yes</td>
</tr>
<tr>
<td>SSL key password</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Tags 2

<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag names</td>
<td>yes</td>
</tr>
<tr>
<td>Tag values</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Other locations

In addition to the locations listed here, user macros can be used in the following fields:

<table>
<thead>
<tr>
<th>Location</th>
<th>Multiple macros/mix with text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global scripts (script, SSH, Telnet, IPMI), including confirmation text</td>
<td>yes</td>
</tr>
</tbody>
</table>
For a complete list of all macros supported in Zabbix, see supported macros.

Footnotes

1 If multiple macros in a field or macros mixed with text are not supported for the location, a single macro has to fill the whole field.

2 Macros used in tag names and values are resolved only during event generation process.

3 URLs that contain a secret macro will not work, as the macro in them will be resolved as “*****”.

8 Unit symbols

Overview

Having to use some large numbers, for example ‘86400’ to represent the number of seconds in one day, is both difficult and error-prone. This is why you can use some appropriate unit symbols (or suffixes) to simplify Zabbix trigger expressions and item keys.

Instead of ‘86400’ for the number of seconds you can simply enter ‘1d’. Suffixes function as multipliers.

Time suffixes

For time you can use:

- s - seconds (when used, works the same as the raw value)
- m - minutes
• h - hours
• d - days
• w - weeks

Time suffixes support only integer numbers (so ‘1h’ is supported, ‘1.5h’ or ‘1.5h’ are not; use ‘90m’ instead).

Time suffixes are supported in:

• trigger expression constants and function parameters
• constants of calculated item formulas
• parameters of the zabbix[queue,<from>,<to>] internal item
• time period parameter of aggregate calculations
• item configuration (‘Update interval’, ‘Custom intervals’, ‘History storage period’ and ‘Trend storage period’ fields)
• item prototype configuration (‘Update interval’, ‘Custom intervals’, ‘History storage period’ and ‘Trend storage period’ fields)
• low-level discovery rule configuration (‘Update interval’, ‘Custom intervals’, ‘Keep lost resources’ fields)
• network discovery configuration (‘Update interval’ field)
• web scenario configuration (‘Update interval’, ‘Timeout’ fields)
• action operation configuration (‘Default operation step duration’, ‘Step duration’ fields)
• user profile settings (‘Auto-logout’, ‘Refresh’, ‘Message timeout’ fields)
• graph widget of Monitoring → Dashboard (‘Time shift’ field)
• Administration → General → Housekeeping (storage period fields)
• Administration → General → Trigger displaying options (‘Display OK triggers for’, ‘On status change triggers blink for’ fields)
• Administration → General → Other (‘Login blocking interval’ field and fields related to communication with Zabbix server)
• Zabbix server ha_set_failover_delay=delay runtime control option

Memory suffixes

Memory size suffixes are supported in:

• trigger expression constants and function parameters
• constants of calculated item formulas

For memory size you can use:

• K - kilobyte
• M - megabyte
• G - gigabyte
• T - terabyte

Other uses

Unit symbols are also used for a human-readable representation of data in the frontend.

In both Zabbix server and frontend these symbols are supported:

• K - kilo
• M - mega
• G - giga
• T - tera

When item values in B, Bps are displayed in the frontend, base 2 is applied (1K = 1024). Otherwise a base of 10 is used (1K = 1000).

Additionally the frontend also supports the display of:

• P - peta
• E - exa
• Z - zetta
• Y - yotta

Usage examples

By using some appropriate suffixes you can write trigger expressions that are easier to understand and maintain, for example these expressions:

last(/host/system.uptime[])<86400s
avg(/host/system.cpu.load,600s)<10
last(/host/vm.memory.size[available])<20971520

could be changed to:

last(/host/system.uptime[])<1d
avg(/host/system.cpu.load,10m)<10

1553
last(/host/vm.memory.size[available])<20M

9 Time period syntax

Overview
To set a time period, the following format has to be used:

\[d-d, hh:mm-hh:mm\]

where the symbols stand for the following:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>Day of the week: 1 - Monday, 2 - Tuesday ..., 7 - Sunday</td>
</tr>
<tr>
<td>hh</td>
<td>Hours: 00-24</td>
</tr>
<tr>
<td>mm</td>
<td>Minutes: 00-59</td>
</tr>
</tbody>
</table>

You can specify more than one time period using a semicolon (;) separator:

\[d-d, hh:mm-hh:mm; d-d, hh:mm-hh:mm\]

Leaving the time period empty equals 01-07,00:00-24:00, which is the default value.

The upper limit of a time period is not included. Thus, if you specify 09:00-18:00 the last second included in the time period is 17:59:59.

Examples

Working hours. Monday - Friday from 9:00 till 18:00:

\[1-5, 09:00-18:00\]

Working hours plus weekend. Monday - Friday from 9:00 till 18:00 and Saturday, Sunday from 10:00 till 16:00:

\[1-5, 09:00-18:00; 6-7, 10:00-16:00\]

10 Command execution

Zabbix uses common functionality for external checks, user parameters, system.run items, custom alert scripts, remote commands and user scripts.

Execution steps

The command/script is executed similarly on both Unix and Windows platforms:

1. Zabbix (the parent process) creates a pipe for communication
2. Zabbix sets the pipe as the output for the to-be-created child process
3. Zabbix creates the child process (runs the command/script)
4. A new process group (in Unix) or a job (in Windows) is created for the child process
5. Zabbix reads from the pipe until timeout occurs or no one is writing to the other end (ALL handles/file descriptors have been closed). Note that the child process can create more processes and exit before they exit or close the handle/file descriptor.
6. If the timeout has not been reached, Zabbix waits until the initial child process exits or timeout occurs
7. If the initial child process exited and the timeout has not been reached, Zabbix checks exit code of the initial child process and compares it to 0 (non-zero value is considered as execution failure, only for custom alert scripts, remote commands and user scripts executed on Zabbix server and Zabbix proxy)
8. At this point it is assumed that everything is done and the whole process tree (i.e. the process group or the job) is terminated

Zabbix assumes that a command/script has done processing when the initial child process has exited AND no other process is still keeping the output handle/file descriptor open. When processing is done, ALL created processes are terminated.

All double quotes and backslashes in the command are escaped with backslashes and the command is enclosed in double quotes.

Exit code checking

Exit code are checked with the following conditions:

- Only for custom alert scripts, remote commands and user scripts executed on Zabbix server and Zabbix proxy.
Any exit code that is different from 0 is considered as execution failure.

Contents of standard error and standard output for failed executions are collected and available in frontend (where execution result is displayed).

Additional log entry is created for remote commands on Zabbix server to save script execution output and can be enabled using LogRemoteCommands agent parameter.

Possible frontend messages and log entries for failed commands/scripts:

- Contents of standard error and standard output for failed executions (if any).
- “Process exited with code: N.” (for empty output, and exit code not equal to 0).
- “Process killed by signal: N.” (for process terminated by a signal, on Linux only).
- “Process terminated unexpectedly.” (for process terminated for unknown reasons).

Read more about:
- External checks
- User parameters
- system.run items
- Custom alert scripts
- Remote commands
- Global scripts

13 Version compatibility

Supported agents

To be compatible with Zabbix 6.2, Zabbix agent must not be older than version 1.4 and must not be newer than 6.2.

You may need to review the configuration of older agents as some parameters have changed, for example, parameters related to logging for versions before 3.0.

To take full advantage of the latest metrics, improved performance and reduced memory usage, use the latest supported agent.

Supported agents 2

Older Zabbix agents 2 from version 4.4 onwards are compatible with Zabbix 6.2; Zabbix agent 2 must not be newer than 6.2.

Note that when using Zabbix agent 2 versions 4.4 and 5.0, the default interval of 10 minutes is used for refreshing unsupported items.

To take full advantage of the latest metrics, improved performance and reduced memory usage, use the latest supported agent 2.

Supported Zabbix proxies

To be compatible with Zabbix 6.2, the proxy must be of the same major version; thus only Zabbix 6.2.x proxies can work with Zabbix 6.2.x server.

It is no longer possible to start the upgraded server and have older, yet unupgraded proxies report data to a newer server. This approach, which was never recommended nor supported by Zabbix, now is officially disabled, as the server will ignore data from unupgraded proxies. See also the upgrade procedure.

Warnings about using incompatible Zabbix daemon versions are logged.

Supported XML files

XML files not older than version 1.8 are supported for import in Zabbix 6.2.

In the XML export format, trigger dependencies are stored by name only. If there are several triggers with the same name (for example, having different severities and expressions) that have a dependency defined between them, it is not possible to import them. Such dependencies must be manually removed from the XML file and re-added after import.

14 Database error handling

If Zabbix detects that the backend database is not accessible, it will send a notification message and continue the attempts to connect to the database. For some database engines, specific error codes are recognized.

MySQL
15 Zabbix sender dynamic link library for Windows

In a Windows environment applications can send data to Zabbix server/proxy directly by using the Zabbix sender dynamic link library (zabbix_sender.dll) instead of having to launch an external process (zabbix_sender.exe).

The dynamic link library with the development files is located in bin\winXX\dev folders. To use it, include the zabbix_sender.h header file and link with the zabbix_sender.lib library. An example file with Zabbix sender API usage can be found in build\win32\examples\zabbix_sender folder.

The following functionality is provided by the Zabbix sender dynamic link library:

```c
int zabbix_sender_send_values(const char *address, unsigned short port, const char *source, const zabbix_sender_value_t *values, int count, char **result);
```

The following data structures are used by the Zabbix sender dynamic link library:

```c
typedef struct
{
    /* host name, must match the name of target host in Zabbix */
    char    *host;
    /* the item key */
    char    *key;
    /* the item value */
    char    *value;
} zabbix_sender_value_t;

typedef struct
{
    /* number of total values processed */
    int total;
    /* number of failed values */
    int failed;
    /* time in seconds the server spent processing the sent values */
    double time_spent;
} zabbix_sender_info_t;
```

17 Service monitoring upgrade

**Overview**  In Zabbix 6.0, service monitoring functionality has been reworked significantly (see What's new in Zabbix 6.0.0 for the list of changes).

This page describes how services and SLAs, defined in earlier Zabbix versions, are changed during an upgrade to Zabbix 6.0 or newer.
Services

In older Zabbix versions, services had two types of dependencies: soft and hard. After an upgrade, all dependencies will become equal.

If a service "Child service" has been previously linked to "Parent service 1" via hard dependency and additionally "Parent service 2" via soft dependency, after an upgrade the "Child service" will have two parent services "Parent service 1" and "Parent service 2".

Trigger-based mapping between problems and services has been replaced by tag-based mapping. In Zabbix 6.0 and newer, service configuration form has a new parameter Problem tags, which allows specifying one or multiple tag name and value pairs for problem matching. Triggers that have been linked to a service will get a new tag ServiceLink: <trigger ID>:<trigger name> (tag value will be truncated to 32 characters). Linked services will get ServiceLink problem tag with the same value.

Status calculation rules

The ‘Status calculation algorithm’ will be upgraded using the following rules:

- Do not calculate → Set status to OK
- Problem, if at least one child has a problem → Most critical of child nodes
- Problem, if all children have problems → Most critical if all children have problems

If you have upgraded from Zabbix pre-6.0 to Zabbix 6.0.0, 6.0.1 or 6.0.2, see Known issues for Zabbix 6.0 documentation.

SLAs

Previously, SLA targets had to be defined for each service separately. Since Zabbix 6.0, SLA has become a separate entity, which contains information about service schedule, expected service level objective (SLO) and downtime periods to exclude from the calculation. Once configured, an SLA can be assigned to multiple services through service tags.

During an upgrade:

- Identical SLAs defined for each service will be grouped and one SLA per each group will be created.
- Each affected service will get a special tag SLA:<ID> and the same tag will be specified in the Service tags parameter of the corresponding SLA.
- Service creation time, a new metric in SLA reports, will be set to 01/01/2000 00:00 for existing services.

18 Other issues

Login and systemd

We recommend creating a zabbix user as system user, that is, without ability to log in. Some users ignore this recommendation and use the same account to log in (e. g. using SSH) to host running Zabbix. This might crash Zabbix daemon on log out. In this case you will get something like the following in Zabbix server log:

zabbix_server [27730]: [file:'selfmon.c',line:375] lock failed: [22] Invalid argument
zabbix_server [27716]: [file:'dbcconfig.c',line:5266] lock failed: [22] Invalid argument
zabbix_server [27706]: [file:'log.c',line:238] lock failed: [22] Invalid argument

and in Zabbix agent log:

zabbix_agentd [27796]: [file:'log.c',line:238] lock failed: [22] Invalid argument

This happens because of default systemd setting RemoveIPC=yes configured in /etc/systemd/logind.conf. When you log out of the system the semaphores created by Zabbix previously are removed which causes the crash.

A quote from systemd documentation:

RemoveIPC=

Controls whether System V and POSIX IPC objects belonging to the user shall be removed when the user fully logs out. Takes a boolean argument. If enabled, the user may not consume IPC resources after the last of the user's sessions terminated. This covers System V semaphores, shared memory and message queues, as well as POSIX shared memory and message queues. Note that IPC objects of the root user and other system users are excluded from the effect of this setting. Defaults to "yes".

There are 2 solutions to this problem:

1. (recommended) Stop using zabbix account for anything else than Zabbix processes, create a dedicated account for other things.
2. (not recommended) Set RemoveIPC=no in /etc/systemd/logind.conf and reboot the system. Note that RemoveIPC is a system-wide parameter, changing it will affect the whole system.
Using Zabbix frontend behind proxy

If Zabbix frontend runs behind proxy server, the cookie path in the proxy configuration file needs to be rewritten in order to match the reverse-proxyed path. See examples below. If the cookie path is not rewritten, users may experience authorization issues, when trying to login to Zabbix frontend.

Example configuration for nginx

```plain
# ..
location / {
# ..
proxy_cookie_path /zabbix /
proxy_pass http://192.168.0.94/zabbix/;
# ..
```

Example configuration for Apache

```plain
# ..
ProxyPass "/" http://host/zabbix/
ProxyPassReverse "/" http://host/zabbix/
ProxyPassReverseCookiePath /zabbix /
ProxyPassReverseCookieDomain host zabbix.example.com
# ..
```

19 Agent vs agent 2 comparison

This section describes the differences between the Zabbix agent and the Zabbix agent 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Zabbix agent</th>
<th>Zabbix agent 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming</td>
<td>C</td>
<td>Go with some parts in C</td>
</tr>
<tr>
<td></td>
<td>language</td>
<td></td>
</tr>
<tr>
<td>Daemonization</td>
<td>yes</td>
<td>by systemd only (yes on Windows)</td>
</tr>
<tr>
<td>Supported</td>
<td>Custom loadable modules in C.</td>
<td>Custom plugins in Go.</td>
</tr>
<tr>
<td>extensions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements</td>
<td>Linux, IBM AIX, FreeBSD, NetBSD, OpenBSD, HP-UX, Mac OS X, Solaris: 9, 10, 11, Windows: all desktop and server versions since XP</td>
<td>Linux, Windows: all desktop and server versions, on which an up-to-date supported Go version can be installed.</td>
</tr>
<tr>
<td>Supported</td>
<td>GnuTLS 3.1.18 and newer</td>
<td>Linux: OpenSSL 1.0.1 and later is supported since Zabbix 4.4.8.</td>
</tr>
<tr>
<td>platforms</td>
<td>OpenSSL 1.0.1, 1.0.2, 1.1.0, 1.1.1, 3.0.x LibreSSL - tested with versions 2.7.4, 2.8.2 (certain limitations apply, see the Encryption page for details).</td>
<td>MS Windows: OpenSSL 1.1.1 or later. The OpenSSL library must have PSK support enabled. LibreSSL is not supported.</td>
</tr>
<tr>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td>crypto libraries</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monitoring</td>
<td></td>
<td></td>
</tr>
<tr>
<td>processes</td>
<td>A separate active check process for each server/proxy record.</td>
<td>Single process with automatically created threads. The maximum number of threads is determined by the GOMAXPROCS environment variable.</td>
</tr>
<tr>
<td>Processes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metrics</td>
<td>UNIX: see a list of supported items.</td>
<td>UNIX: All metrics supported by Zabbix agent. Additionally, the agent 2 provides Zabbix-native monitoring solution for: Docker, Memcached, MySQL, PostgreSQL, Redis, systemd, and other monitoring targets - see a full list of agent 2 specific items.</td>
</tr>
<tr>
<td></td>
<td>Windows: see a list of additional Windows-specific items.</td>
<td>Windows: All metrics supported by Zabbix agent, and also net.tcp.service* checks of HTTPS, LDAP. Additionally, the agent 2 provides Zabbix-native monitoring solution for: PostgreSQL, Redis. Checks from different plugins or multiple checks within one plugin can be executed concurrently. Supported for passive and active checks.</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Active checks for single server are executed sequentially.</td>
<td></td>
</tr>
<tr>
<td>Scheduled/flexible</td>
<td>Supported for passive checks only.</td>
<td></td>
</tr>
<tr>
<td>intervals</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Parameter | Zabbix agent | Zabbix agent 2
--- | --- | ---
Third party traps | no | yes
Additional features |  | 
Persistent storage | no | yes
Persistent files for log*[] metrics | yes (only on Unix) | no
Timeout settings | Defined on an agent level only. | Plugin timeout can override the timeout defined on an agent level.
Changes user at runtime | yes (Unix-like systems only) | no (controlled by systemd)
User- configurable ciphersuites | yes | no

See also:
- Zabbix processes description: Zabbix agent, Zabbix agent 2
- Configuration parameters: Zabbix agent UNIX / Windows, Zabbix agent 2 UNIX / Windows

Zabbix manpages

These are Zabbix manpages for Zabbix processes.

zabbix_agent2

Section: Maintenance Commands (8)
Updated: 2019-01-29

NAME
zabbix_agent2 - Zabbix agent 2

SYNOPSIS
zabbix_agent2 [-c config-file]
zabbix_agent2 [-c config-file] -p
zabbix_agent2 [-c config-file] -t item-key
zabbix_agent2 [-c config-file] -R runtime-option
zabbix_agent2 -h
zabbix_agent2 -V

DESCRIPTION
zabbix_agent2 is an application for monitoring parameters of various services.
OPTIONS

-\(c\), \(--\text{config}\) config-file
  Use the alternate config-file instead of the default one.

-\(R\), \(--\text{runtime-control}\) runtime-option
  Perform administrative functions according to runtime-option.

Runtime control options: userparameter reload
Reload user parameters from the configuration file

loglevel increase
Increase log level

loglevel decrease
Decrease log level

help
List available runtime control options

metrics
List available metrics

version
Display version

-p, \(--\text{print}\)
Print known items and exit. For each item either generic defaults are used, or specific defaults for testing are supplied. These defaults are listed in square brackets as item key parameters. Returned values are enclosed in square brackets and prefixed with the type of the returned value, separated by a pipe character. For user parameters type is always \(t\), as the agent can not determine all possible return values. Items, displayed as working, are not guaranteed to work from the Zabbix server or zabbix_get when querying a running agent daemon as permissions or environment may be different. Returned value types are:

\(d\)
Number with a decimal part.

\(m\)
Not supported. This could be caused by querying an item that only works in the active mode like a log monitoring item or an item that requires multiple collected values. Permission issues or incorrect user parameters could also result in the not supported state.

\(s\)
Text. Maximum length not limited.

\(t\)
Text. Same as \(s\).

\(u\)
Unsigned integer.

-t, \(--\text{test}\) item-key
Test single item and exit. See \(--\text{print}\) for output description.

-h, \(--\text{help}\)
Display this help and exit.

-V, \(--\text{version}\)
Output version information and exit.

FILES
/usr/local/etc/zabbix_agent2.conf
Default location of Zabbix agent 2 configuration file (if not modified during compile time).
NAME

zabbix_agentd - Zabbix agent daemon

SYNOPSIS

zabbix_agentd [-c config-file]
zabbix_agentd [-c config-file] -p
zabbix_agentd [-c config-file] -t item-key
zabbix_agentd [-c config-file] -R runtime-option
zabbix_agentd -h
zabbix_agentd -V

DESCRIPTION

zabbix_agentd is a daemon for monitoring various server parameters.

OPTIONS

-c, --config config-file
Use the alternate config-file instead of the default one.

-f, --foreground
Run Zabbix agent in foreground.
-R, --runtime-control runtime-option
Perform administrative functions according to runtime-option.

Runtime control options

userparameter_reload[=target]
Reload user parameters from the configuration file

log_level_increase[=target]
Increase log level, affects all processes if target is not specified

log_level_decrease[=target]
Decrease log level, affects all processes if target is not specified

Log level control targets

process-type
All processes of specified type (active checks, collector, listener)

process-type,N
Process type and number (e.g., listener,3)

pid
Process identifier, up to 65535. For larger values specify target as “process-type,N”

-p, --print
Print known items and exit. For each item either generic defaults are used, or specific defaults for testing are supplied. These defaults are listed in square brackets as item key parameters. Returned values are enclosed in square brackets and prefixed with the type of the returned value, separated by a pipe character. For user parameters type is always t, as the agent can not determine all possible return values. Items, displayed as working, are not guaranteed to work from the Zabbix server or zabbix_get when querying a running agent daemon as permissions or environment may be different. Returned value types are:

d
Number with a decimal part.

m
Not supported. This could be caused by querying an item that only works in the active mode like a log monitoring item or an item that requires multiple collected values. Permission issues or incorrect user parameters could also result in the not supported state.

s
Text. Maximum length not limited.

t
Text. Same as s.

u
Unsigned integer.

-t, --test item-key
Test single item and exit. See --print for output description.

-h, --help
Display this help and exit.

-V, --version
Output version information and exit.

FILES
/usr/local/etc/zabbix_agentd.conf
Default location of Zabbix agent configuration file (if not modified during compile time).
NAME
zabbix_get - Zabbix get utility

SYNOPSIS
zabbix_get -s host-name-or-IP [-p port-number] [-l IP-address] [-t timeout] -k item-key
zabbix_get -h
zabbix_get -V

DESCRIPTION
zabbix_get is a command line utility for getting data from Zabbix agent.
OPTIONS

-s, --host host-name-or-IP
Specify host name or IP address of a host.

-p, --port port-number
Specify port number of agent running on the host. Default is 10050.

-l, --source-address IP-address
Specify source IP address.

-t, --timeout seconds
Specify timeout. Valid range: 1-30 seconds (default: 30)

-k, --key item-key
Specify key of item to retrieve value for.

--tls-connect value
How to connect to agent. Values:

    unencrypted
    connect without encryption (default)

    psk
    connect using TLS and a pre-shared key

    cert
    connect using TLS and a certificate

--tls-ca-file CA-file
Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification.

--tls-crl-file CRL-file
Full pathname of a file containing revoked certificates.

--tls-agent-cert-issuer cert-issuer
Allowed agent certificate issuer.

--tls-agent-cert-subject cert-subject
Allowed agent certificate subject.

--tls-cert-file cert-file
Full pathname of a file containing the certificate or certificate chain.

--tls-key-file key-file
Full pathname of a file containing the private key.

--tls-psk-identity PSK-identity
PSK-identity string.

--tls-psk-file PSK-file
Full pathname of a file containing the pre-shared key.

--tls-cipher13 cipher-string
Cipher string for OpenSSL 1.1.1 or newer for TLS 1.3. Override the default ciphersuite selection criteria. This option is not available if OpenSSL version is less than 1.1.1.

--tls-cipher cipher-string
GnuTLS priority string (for TLS 1.2 and up) or OpenSSL cipher string (only for TLS 1.2). Override the default ciphersuite selection criteria.

-h, --help
Display this help and exit.

-V, --version
Output version information and exit.
**NAME**
zabbix_js - Zabbix JS utility

**SYNOPSIS**
zabbix_js -s script-file -p input-param [-l log-level] [-t timeout]
zabbix_js -s script-file -i input-file [-l log-level] [-t timeout]
zabbix_js -h
zabbix_js -V

**DESCRIPTION**
zabbix_js is a command line utility that can be used for embedded script testing.
OPTIONS

-s, --script script-file
Specify the file name of the script to execute. If '-' is specified as file name, the script will be read from stdin.

-p, --param input-param
Specify the input parameter.

-i, --input input-file
Specify the file name of the input parameter. If '-' is specified as file name, the input will be read from stdin.

-l, --loglevel log-level
Specify the log level.

-t, --timeout timeout
Specify the timeout in seconds.

-h, --help
Display this help and exit.

-V, --version
Output version information and exit.

EXAMPLES

zabbix_js -s script-file.js -p example

SEE ALSO

Documentation https://www.zabbix.com/manuals
zabbix_agent2(8), zabbix_agentd(8), zabbix_get(1), zabbix_proxy(8), zabbix_sender(1), zabbix_server(8)

Index
NAME
SYNOPSIS
DESCRIPTION
OPTIONS
EXAMPLES
SEE ALSO

This document was created by man2html, using the manual pages.
Time: 21:23:35 GMT, March 18, 2020

zabbix_proxy

Section: Maintenance Commands (8)
Updated: 2020-09-04
Index Return to Main Contents

NAME

zabbix_proxy - Zabbix proxy daemon
SYNOPSIS

zabbix_proxy [-c config-file]
zabbix_proxy [-c config-file] -R runtime-option
zabbix_proxy -h
zabbix_proxy -V

DESCRIPTION

zabbix_proxy is a daemon that collects monitoring data from devices and sends it to Zabbix server.

OPTIONS

-c, --config config-file
Use the alternate config-file instead of the default one.

-f, --foreground
Run Zabbix proxy in foreground.

-R, --runtime-control runtime-option
Perform administrative functions according to runtime-option.

Runtime control options

config_cache_reload
Reload configuration cache. Ignored if cache is being currently loaded. Active Zabbix proxy will connect to the Zabbix server and request configuration data. Default configuration file (unless -c option is specified) will be used to find PID file and signal will be sent to process, listed in PID file.

snmp_cache_reload
Reload SNMP cache.

housekeeper_execute
Execute the housekeeper. Ignored if housekeeper is being currently executed.

diaginfo[=section]
Log internal diagnostic information of the specified section. Section can be historycache, preprocessing. By default diagnostic information of all sections is logged.

log_level_increase[=target]
Increase log level, affects all processes if target is not specified.

log_level_decrease[=target]
Decrease log level, affects all processes if target is not specified.

Log level control targets

process-type
All processes of specified type (configuration syncer, data sender, discoverer, heartbeat sender, history syncer, housekeeper, http poller, icmp pinger, ipmi manager, ipmi poller, java poller, poller, self-monitoring, snmp trapper, task manager, trapper, unreachable poller, vmware collector)

process-type,N
Process type and number (e.g., poller,3)

pid
Process identifier, up to 65535. For larger values specify target as "process-type,N"

-h, --help
Display this help and exit.

-V, --version
Output version information and exit.
AUTHOR
Alexei Vladishev <alex@zabbix.com>

This document was created by man2html, using the manual pages.
Time: 16:12:22 GMT, September 04, 2020

DESCRIPTION

zabbix_sender is a command line utility for sending monitoring data to Zabbix server or proxy. On the Zabbix server an item of type Zabbix_trapper should be created with corresponding key. Note that incoming values will only be accepted from hosts specified in Allowed hosts field for this item.

OPTIONS

-c, --config config-file
Use config-file. Zabbix sender reads server details from the agentd configuration file. By default Zabbix sender does not read any configuration file. Only parameters Hostname, ServerActive, SourceIP, TLSConnect, TLSCAFile, TLSConsole, TLSServerCertIssuer, TLSServerCertSubject, TLSKeyFile, TLSPSKIdentity and TLSPSKFile are supported. All addresses defined in the agent ServerActive configuration parameter are used for sending data. If sending of batch data fails to one address, the following batches are not sent to this address.

-z, --zabbix-server server
Hostname or IP address of Zabbix server. If a host is monitored by a proxy, proxy hostname or IP address should be used instead. When used together with --config, overrides the entries of ServerActive parameter specified in agentd configuration file.

-p, --port port
Specify port number of Zabbix server trapper running on the server. Default is 10051. When used together with --config, overrides the port entries of ServerActive parameter specified in agentd configuration file.

-l, --source-address IP-address
Specify source IP address. When used together with --config, overrides SourceIP parameter specified in agentd configuration file.

-t, --timeout seconds
Specify timeout. Valid range: 1-300 seconds (default: 60)

-s, --host host
Specify host name the item belongs to (as registered in Zabbix frontend). Host IP address and DNS name will not work. When used together with --config, overrides Hostname parameter specified in agentd configuration file.

-k, --key key
Specify item key to send value to.

-o, --value value
Specify item value.

-I, --input-file input-file
Load values from input file. Specify - as <input-file> to read values from standard input. Each line of file contains whitespace delimited: <hostname> <key> <value>. Each value must be specified on its own line. Each line must contain 3 whitespace delimited entries: <hostname> <key> <value>, where "hostname" is the name of monitored host as registered in Zabbix frontend, "key" is target item key and "value" - the value to send. Specify - as <hostname> to use hostname from agent configuration file or from --host argument.

An example of a line of an input file:

"Linux DB3" db.connections 43

The value type must be correctly set in item configuration of Zabbix frontend. Zabbix sender will send up to 250 values in one connection. Contents of the input file must be in the UTF-8 encoding. All values from the input file are sent in a sequential order top-down. Entries must be formatted using the following rules:
Quoted and non-quoted entries are supported.

Double-quote is the quoting character.

Entries with whitespace must be quoted.

Double-quote and backslash characters inside quoted entry must be escaped with a backslash.

Escaping is not supported in non-quoted entries.

Linefeed escape sequences (\n) are supported in quoted strings.

Linefeed escape sequences are trimmed from the end of an entry.

-T, --with-timestamps
This option can be only used with --input-file option.

Each line of the input file must contain 4 whitespace delimited entries: <hostname> <key> <timestamp> <value>. Timestamp should be specified in Unix timestamp format. If target item has triggers referencing it, all timestamps must be in an increasing order, otherwise event calculation will not be correct.

An example of a line of the input file:
"Linux DB3" db.connections 1429533600 43
For more details please see option --input-file.

If a timestamped value is sent for a host that is in a “no data” maintenance type then this value will be dropped; however, it is possible to send a timestamped value in for an expired maintenance period and it will be accepted.

-N, --with-ns
This option can be only used with --with-timestamps option.

Each line of the input file must contain 5 whitespace delimited entries: <hostname> <key> <timestamp> <ns> <value>.

An example of a line of the input file:
"Linux DB3" db.connections 1429533600 7402561 43
For more details please see option --input-file.

-r, --real-time
Send values one by one as soon as they are received. This can be used when reading from standard input.

--tls-connect value
How to connect to server or proxy. Values:

  unencrypted
  connect without encryption (default)

  psk
  connect using TLS and a pre-shared key

  cert
  connect using TLS and a certificate

--tls-ca-file CA-file
Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification.

--tls-crl-file CRL-file
Full pathname of a file containing revoked certificates.

--tls-server-cert-issuer cert-issuer
Allowed server certificate issuer.

--tls-server-cert-subject cert-subject
Allowed server certificate subject.
--tls-cert-file cert-file
Full pathname of a file containing the certificate or certificate chain.

--tls-key-file key-file
Full pathname of a file containing the private key.

--tls-psk-identity PSK-identity
PSK-identity string.

--tls-psk-file PSK-file
Full pathname of a file containing the pre-shared key.

--tls-cipher13 cipher-string
Cipher string for OpenSSL 1.1.1 or newer for TLS 1.3. Override the default ciphersuite selection criteria. This option is not available if OpenSSL version is less than 1.1.1.

--tls-cipher cipher-string
GnuTLS priority string (for TLS 1.2 and up) or OpenSSL cipher string (only for TLS 1.2). Override the default ciphersuite selection criteria.

-v, --verbose
Verbose mode, -vv for more details.

-h, --help
Display this help and exit.

-V, --version
Output version information and exit.

EXIT STATUS
The exit status is 0 if the values were sent and all of them were successfully processed by server. If data was sent, but processing of at least one of the values failed, the exit status is 2. If data sending failed, the exit status is 1.

EXAMPLES
zabbix_sender -c /etc/zabbix/zabbix_agentd.conf -k mysql.queries -o 342.45
Send 342.45 as the value for mysql.queries item of monitored host. Use monitored host and Zabbix server defined in agent configuration file.

zabbix_sender -c /etc/zabbix/zabbix_agentd.conf -s "Monitored Host" -k mysql.queries -o 342.45
Send 342.45 as the value for mysql.queries item of Monitored Host host using Zabbix server defined in agent configuration file.

zabbix_sender -z 192.168.1.113 -i data_values.txt
Send values from file data_values.txt to Zabbix server with IP 192.168.1.113. Host names and keys are defined in the file.

echo "- hw.serial.number 1287872261 SQ4321ASDF" | zabbix_sender -c /usr/local/etc/zabbix_agentd.conf -T -i -
Send a timestamped value from the commandline to Zabbix server, specified in the agent configuration file. Dash in the input data indicates that hostname also should be used from the same configuration file.

echo ""Zabbix server" trapper.item """" | zabbix_sender -z 192.168.1.113 -p 10000 -i -
Send empty value of an item to the Zabbix server with IP address 192.168.1.113 on port 10000 from the commandline. Empty values must be indicated by empty double quotes.
Send 342.45 as the value for mysql.queries item in Monitored Host host to server with IP 192.168.1.113 using TLS with certificate.

zabbix_sender -z 192.168.1.113 -s "Monitored Host" -k mysql.queries -o 342.45 --tls-connect cert --tls-ca-file /home/zabbix/zabbix_ca_file --tls-cert-file /home/zabbix/zabbix_agentd.crt --tls-key-file /home/zabbix/zabbix_agentd.key

Send 342.45 as the value for mysql.queries item in Monitored Host host to server with IP 192.168.1.113 using TLS with pre-shared key (PSK).

zabbix_sender -z 192.168.1.113 -s "Monitored Host" -k mysql.queries -o 342.45 --tls-connect psk --tls-psk-identity "PSK ID Zabbix agentd" --tls-psk-file /home/zabbix/zabbix_agentd.psk

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agentd(8), zabbix_get(1), zabbix_proxy(8), zabbix_server(8), zabbix_js(1), zabbix_agent2(8), zabbix_web_service(8)

AUTHOR

Alexei Vladishev <[email protected]>

This document was created by man2html, using the manual pages.
Time: 08:42:39 GMT, June 11, 2021

zabbix_server

Section: Maintenance Commands (8)
Updated: 2020-09-04

NAME

zabbix_server - Zabbix server daemon
SYNOPSIS

zabbix_server [-c config-file]
zabbix_server [-c config-file] -R runtime-option
zabbix_server -h
zabbix_server -V

DESCRIPTION

zabbix_server is the core daemon of Zabbix software.

OPTIONS

- c, --config config-file
Use the alternate config-file instead of the default one.

-f, --foreground
Run Zabbix server in foreground.

-R, --runtime-control runtime-option
Perform administrative functions according to runtime-option.

Runtime control options

config_cache_reload
Reload configuration cache. Ignored if cache is being currently loaded. Default configuration file (unless -c option is specified) will be used to find PID file and signal will be sent to process, listed in PID file.

snmp_cache_reload
Reload SNMP cache.

housekeeper_execute
Execute the housekeeper. Ignored if housekeeper is being currently executed.

diaginfo[=section]
Log internal diagnostic information of the specified section. Section can be historycache, preprocessing, alerting, lld, valuecache. By default diagnostic information of all sections is logged.

log_level_increase[=target]
Increase log level, affects all processes if target is not specified

log_level_decrease[=target]
Decrease log level, affects all processes if target is not specified

Log level control targets

process-type
All processes of specified type (alerter, alert manager, configuration syncer, discoverer, escalator, history syncer, housekeeper, http poller, icmp pinger, ipmi manager, ipmi poller, java poller, lld manager, lld worker, poller, preprocessing manager, preprocessing worker, proxy poller, self-monitoring, snmp trapper, task manager, timer, trapper, unreachable poller, vmware collector)

process-type,N
Process type and number (e.g., poller,3)

pid
Process identifier, up to 65535. For larger values specify target as “process-type,N”

-h, --help
Display this help and exit.

-V, --version
Output version information and exit.
NAME
zabbix_web_service - Zabbix web service

SYNOPSIS
zabbix_web_service [-c config-file]
zabbix_web_service -h
zabbix_web_service -V

DESCRIPTION
zabbix_web_service is an application for providing web services to Zabbix components.
OPTIONS

-c, --config config-file
Use the alternate config-file instead of the default one.

-h, --help
Display this help and exit.

-V, --version
Output version information and exit.

FILES

/usr/local/etc/zabbix_web_service.conf
Default location of Zabbix web service configuration file (if not modified during compile time).

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agentd(8), zabbix_get(1), zabbix_proxy(8), zabbix_sender(1), zabbix_server(8), zabbix_js(1), zabbix_agent2(8)

AUTHOR

Zabbix LLC