Documentation 5.0

ZABBIX

01.07.2025

Contents

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.

1. Introduction

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

1 Manual structure

Structure

The content of this Zabbix 5.0 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 web site 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
- custom screens and slide shows for 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, third-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).

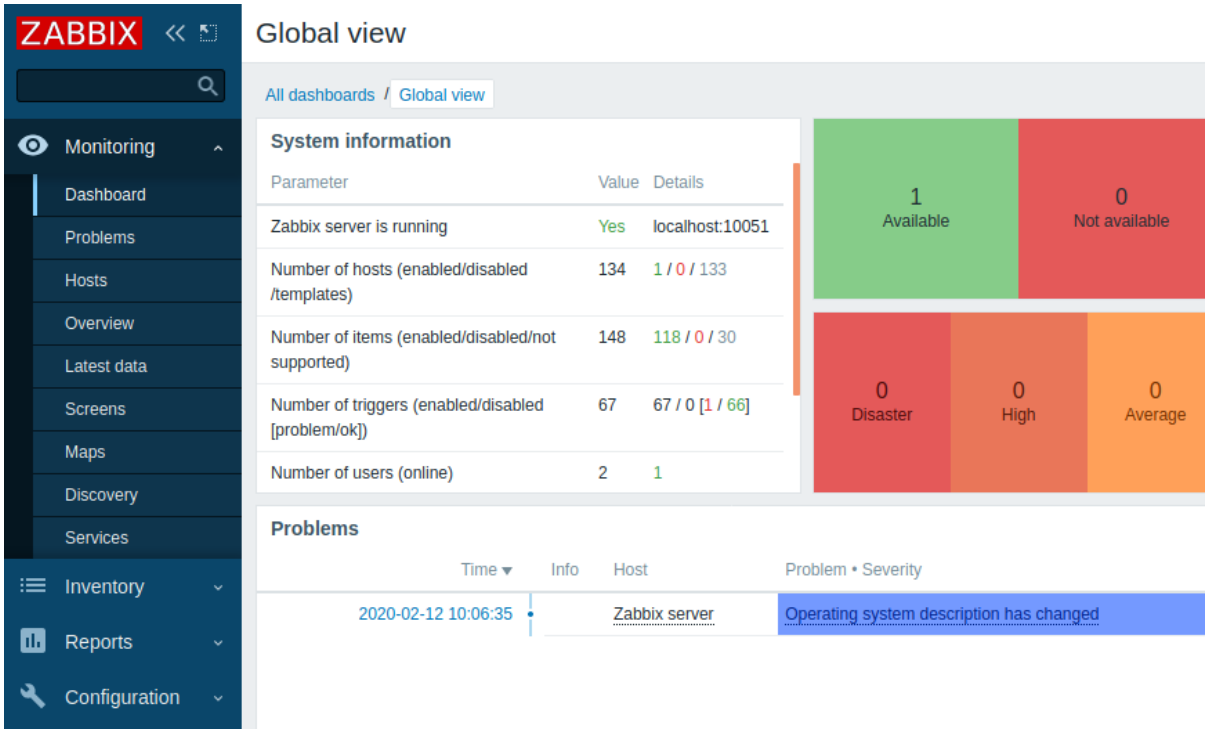
Data flow

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.

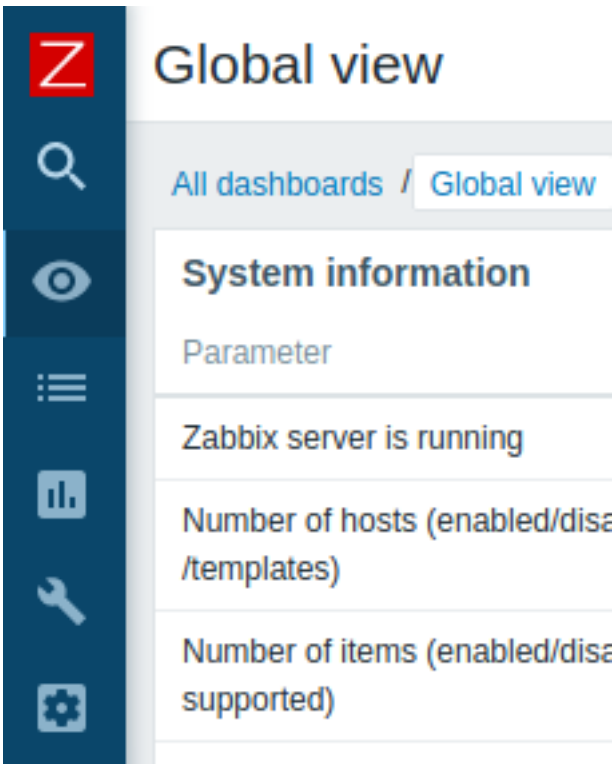
5 What's new in Zabbix 5.0.0

Vertical menu

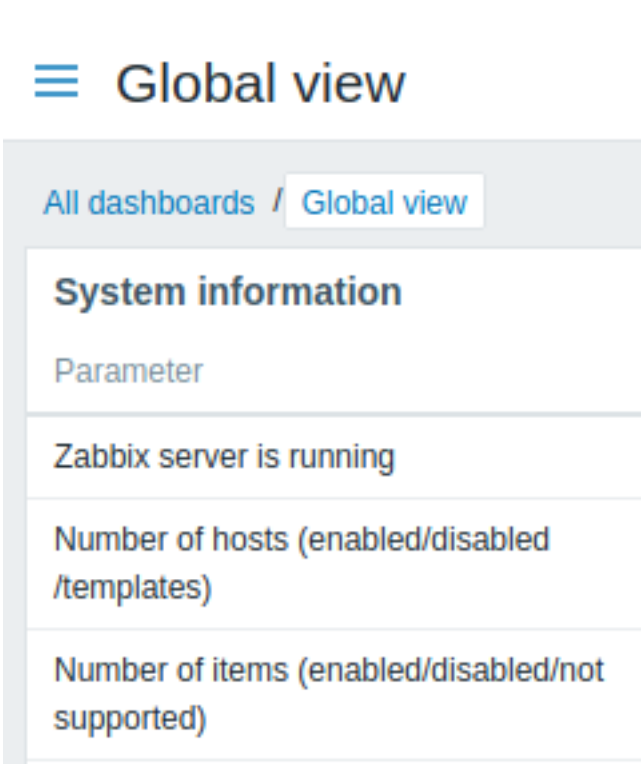
A modern vertical menu in a sidebar replaces the horizontal menu in the new version.



The menu can be collapsed or hidden completely:



Collapsed menu with icons only.



Hidden menu.

When the menu is collapsed, a full menu reappears as soon as the mouse cursor is placed upon it. Even when the menu is hidden completely, a full menu is just one mouse click away. See additional details.

String comparison allowed

String comparison is now allowed in triggers, using the **=** (equal) and **<>** (not equal) operators.

So, for example, it is now possible to define triggers that create alarm if the strings returned by two items are different:

```
{Local Zabbix server:vfs.file.contents[/etc/os-release].last()}<>{Remote Zabbix server:vfs.file.contents[/
```

String comparison is also possible in calculated items.

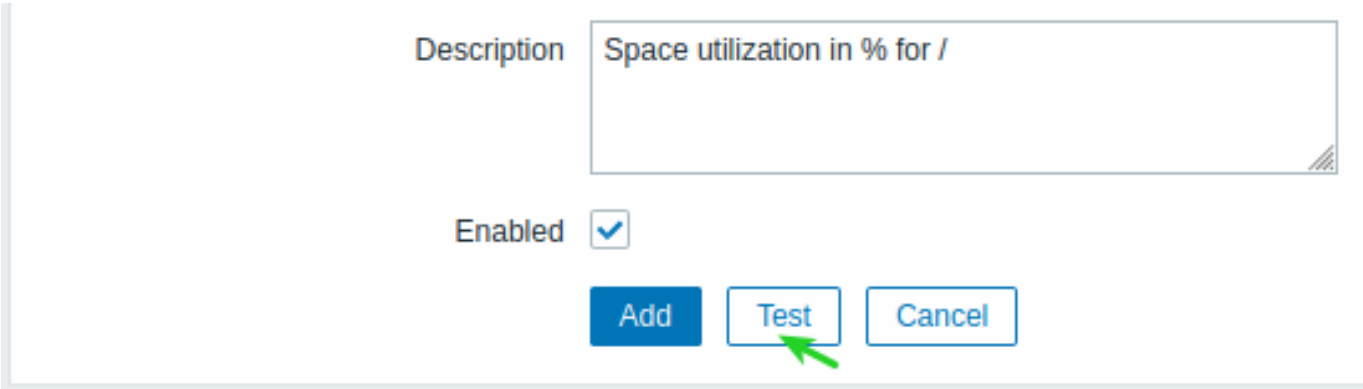
Test item from UI

In previous Zabbix versions, it was difficult to tell if a newly-configured item was configured correctly or not. For that you needed to wait until the item tried to gather some data.

In the new version it is possible to test the item (template item, item prototype, low-level discovery rule) from the user interface even before saving and, if configured correctly, get a real value in return.

Item testing is not supported for active items and some simple checks (`icmpping*`, `vmware.*` items).

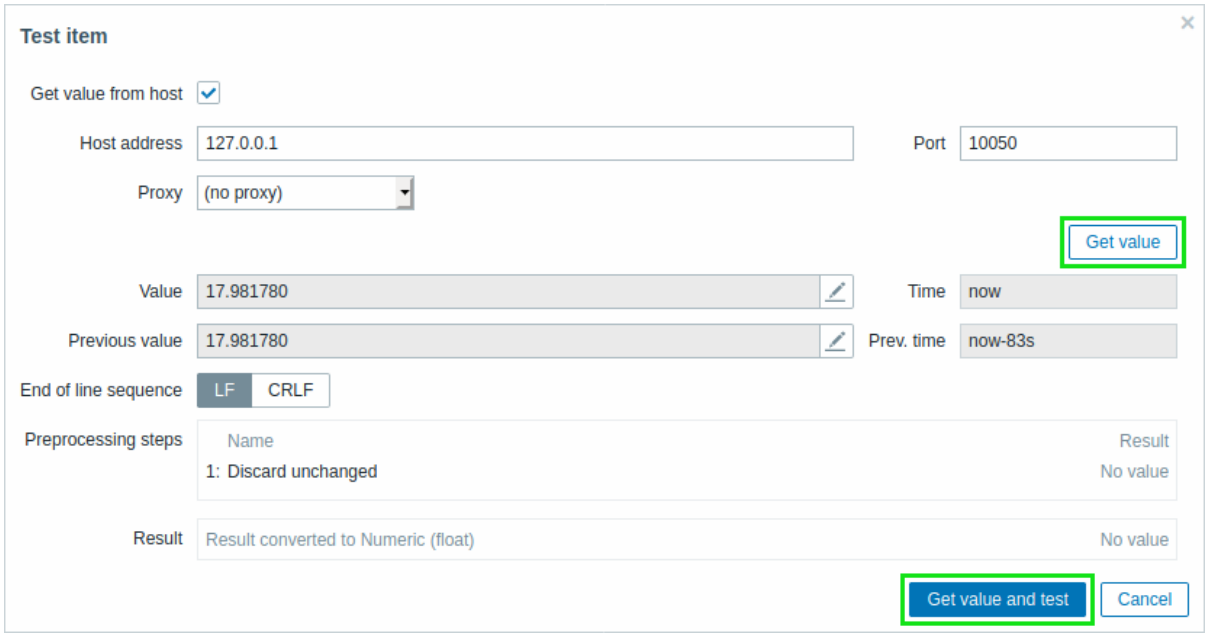
To test the item, click on the *Test* button at the bottom of the item configuration form.



The item testing form has fields for the required host parameters (host address, port, proxy name/no proxy). 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
- The fields are disabled when not needed in the context of the item type (e.g. the host address field is disabled for calculated and aggregate items, the proxy field is disabled for calculated items)

To test the item, click on *Get value*. If the value is retrieved successfully, it will fill the *Value* field.



A successfully retrieved value from host can also be used to test the preprocessing steps.

In fact, the item testing form is an extension of the preprocessing testing form already known in recent Zabbix versions. So if previously you could test preprocessing steps only against a hypothetical input value, now it is also possible to test preprocessing against a real test value just received.

To test the preprocessing steps against the real value, click on *Get value and test*.

See also:

- Testing an item
- Testing preprocessing steps

Execute now

In a related development the *Check now* option has been renamed to *Execute now*, to avoid confusing it with the item testing functionality.

No data triggers sensitive to proxy availability

No data triggers are now, by default, sensitive to proxy availability - the 'nodata' triggers will not fire immediately after a restored connection, but will skip the data for the delayed period.

Suppression is turned on:

- for passive proxies - if connection is restored more than 15 seconds and no less than 2 & ProxyUpdateFrequency seconds later
- for active proxies - if connection is restored more than 15 seconds later

You may also turn off sensitiveness to proxy availability, using the new second parameter, e.g.: nodata(5m,strict). In this case the function will work the same as before and fire as soon as the evaluation period (five minutes, in this case) without data is past.

It is also possible to monitor how long data are delayed on the proxy using the new `zabbix[proxy,<proxy name>,delay]` internal item.

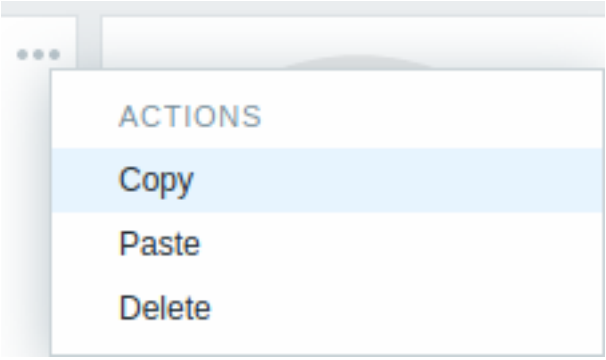
Custom frontend modules

It is now possible to enhance Zabbix frontend functionality by adding third-party modules or by developing your own modules without the need to change the source code of Zabbix. See Modules for more information.

Copy and paste widgets

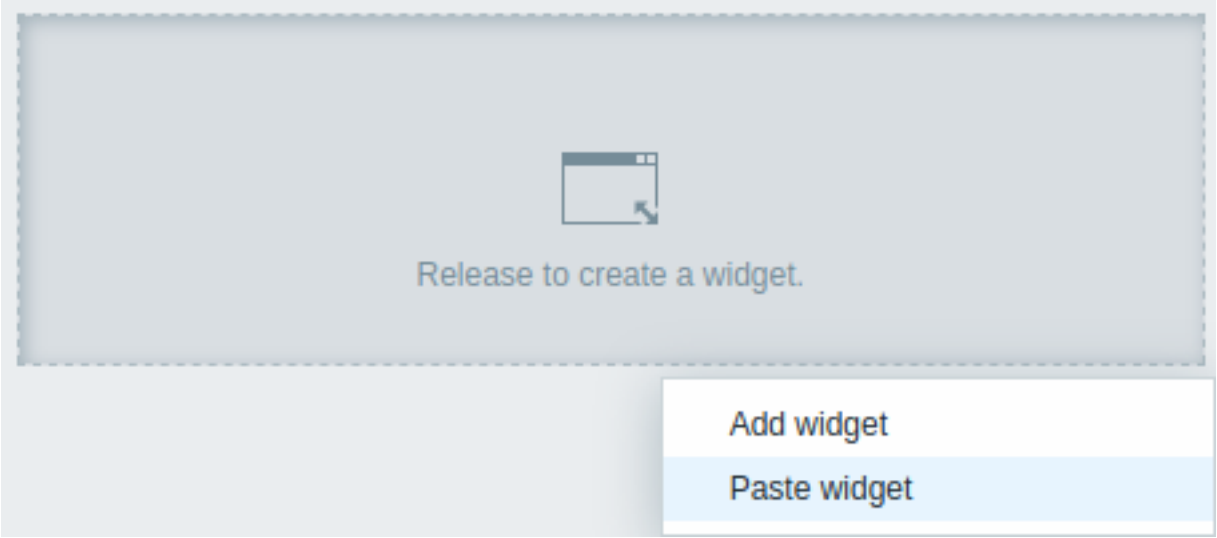
Dashboard widgets can be copied and pasted in the new version. They can be copy-pasted within the same dashboard, or between dashboards opened in different tabs.

A widget can be copied using the widget menu:



Then the copied widget can be used to create a new widget with the same properties. To paste a widget:

- use the *Paste widget* button when editing the dashboard
- use the *Paste widget* option when adding a new widget by selecting some area in the dashboard (a widget must copied first for the paste option to become available)



A copied widget can also be used to paste over an existing widget using the *Paste* option in the widget menu.

Managing large numbers of hosts

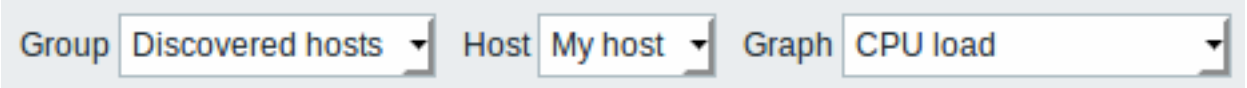
Several improvements have been made to make it easier to work with large numbers of hosts and other elements.

A consistent Zabbix feature in previous versions was the presence of many dropdowns for host and host group selection, and sometimes also for selecting other elements like graphs. Typical locations for these dropdowns included top of the page and popups. In the new version that has been changed in many locations (see the list of locations below):

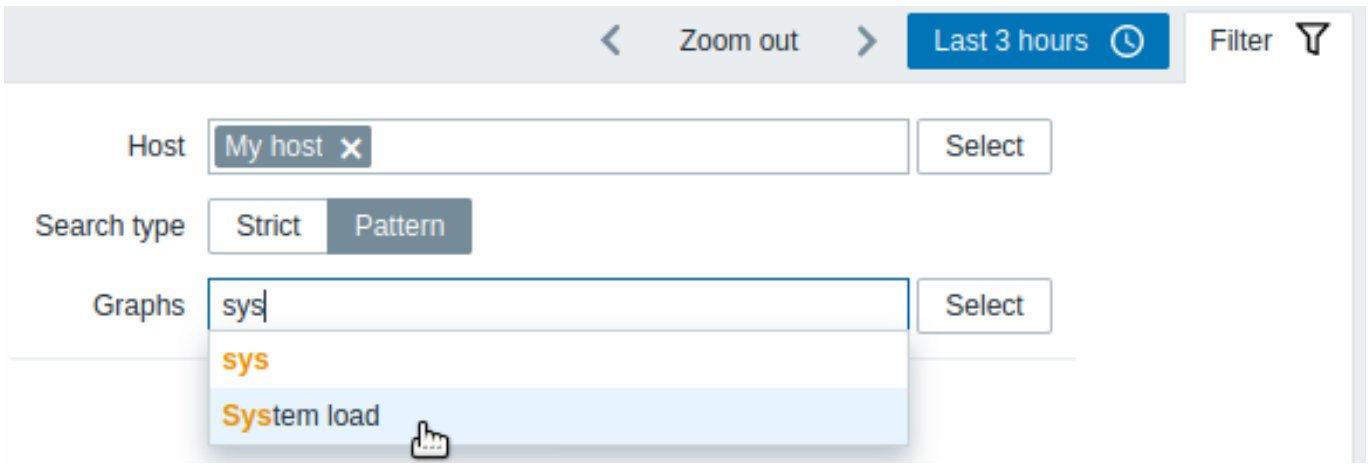
- multi-select fields have replaced dropdowns in popups
- multi-select fields have replaced many dropdowns that were located top of the page; many of these fields have also been moved into filters

Note that:

- two host group and host dropdowns are in places changed to a single host multi-select field followed by a *Select* popup for host group selection
- there is also a new option to search for graph name pattern:



Top-of-th



Multi-se

- filtering by host has been added to trigger/data overview pages and widgets
- the *Rows per page* setting from user profile is applied to web monitoring, host inventory overview and availability report pages
- hardcoded limit of 50 records without pagination is applied to trigger/data overview pages and widgets

Please see individual pages for details on changed host/host group/graph/etc selection:

- Monitoring:
  - Dashboard (host selection with dynamic widgets)
  - Host graphs
  - Host web scenarios
  - Trigger overview
  - Data overview
  - Screens/slide shows (host selection with dynamic screen elements)
- Inventory:
  - Host inventory overview
  - Host inventory
- Reports:
  - Availability report (by host)
- Configuration lists:
  - Hosts
  - Templates
  - Applications
  - Graphs
  - Web scenarios

Overriding in LLD rules

It is now possible to filter out items, triggers, hosts and graphs or override their attributes during low-level discovery based on LLD object and prototype name.

IPMI sensor discovery

A new `ipmi.get` IPMI item has been added that returns a JSON with IPMI-sensor related information. This item can be used for the discovery of IPMI sensors.

Item key limit raised

The maximum allowed length of an item key has been raised from 256 to 2048 characters.

Extended range of numeric (float) values

Numeric (float) data type now supports precision of approximately 15 digits and range from approximately -1.79E+308 to 1.79E+308 (with exception of PostgreSQL 11 and earlier versions). This is by default for new installations. For upgraded installations, a manual patch must be applied.

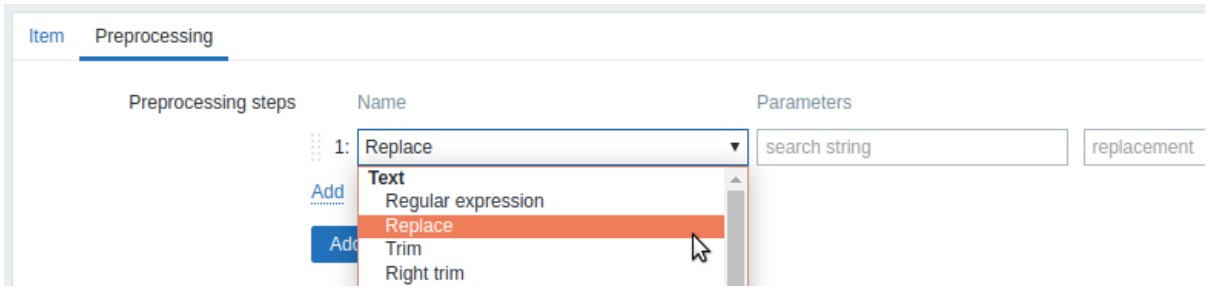
ODBC monitoring without DSN

A new `connection string` parameter has been added to `db.odbc.*` items. Now *Database monitor* items can be configured in two ways:

- Using the data source name as set in /etc/odbc.ini
- Using a connection string

Find and replace preprocessing step

A new item value preprocessing option has been added allowing to find and replace a specified string with another:



This step has two parameters:

- *search string* - the string to search for
- *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.

Nanosecond support by Zabbix sender input file

A new Zabbix sender option

`-N, --with-ns`

allows to support nanoseconds in a Zabbix sender input file. This option can be only used together with the `--with-timestamps` option, e.g.:

```
zabbix_sender -z 127.0.0.1 --with-timestamps --with-ns -i values.txt
```

This option specifies that each line of the input file contains the following, whitespace-delimited: `<host> <key> <timestamp> <ns> <value>`, e.g.:

```
Zabbix server" trap001 1429533600 748744024 43
Zabbix server" trap001 1429533600 748791234 44
```

Secure connections to Zabbix database

It is now possible to configure secure TLS connections to MySQL and PostgreSQL databases from:

- Zabbix frontend
- Zabbix server or proxy

Restricting agent checks

It is possible to restrict checks on the agent side by creating a whitelist or blacklist of allowed item keys.

Whitelist/blacklist is created using a combination of two new 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

See also: Restricting agent checks

Stronger cryptography for passwords

A stronger bcrypt cryptography is now used for hashing user passwords instead of MD5. The change to the stronger cryptography after the upgrade is automatic, i.e. no effort on the user side is required. Note that passwords longer than 72 characters will be truncated.

Using HTTP proxy in webhooks

It is now possible to specify an HTTP proxy when configuring a webhook. The new HTTPProxy parameter is listed in the webhook parameter list by default with an empty value.

When specifying the proxy value the same functionality as in the item configuration HTTP proxy field is supported.

Discovery rule filtering

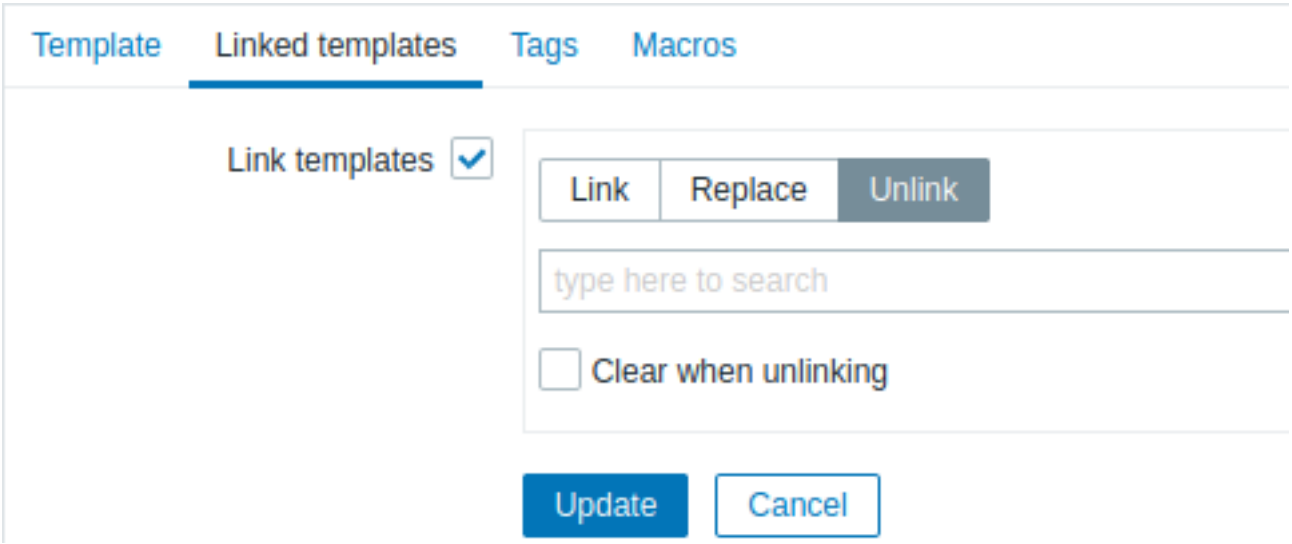
The list of low-level discovery rules previously was always linked to a single host, making it impossible to view all discovery rules in one place or filter the ones of a specific host group or those having errors.

In the new version, the list of low-level discovery rules contains a filter allowing to filter by host group, host, discovery item type, discovery rule state and other parameters. Additionally, the added first column in the list now always displays the host of the discovery rule.

New mass update options

It is now possible to:

- Mass update user macros defined on a host or template level
- Mass unlink templates when using host or template mass update:

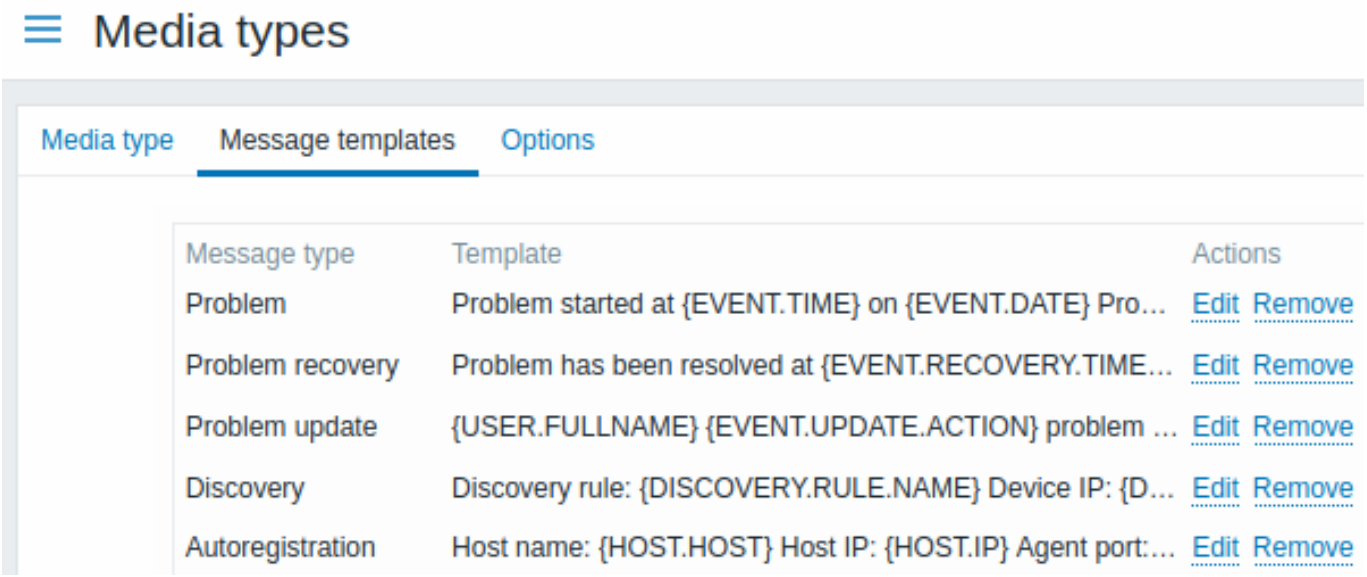


See also:

- Host mass update
- Template mass update

Default messages for each media type

It is now possible to specify default message templates for each event type when defining media types.



Thus default message editing no longer takes place when configuring action operations.

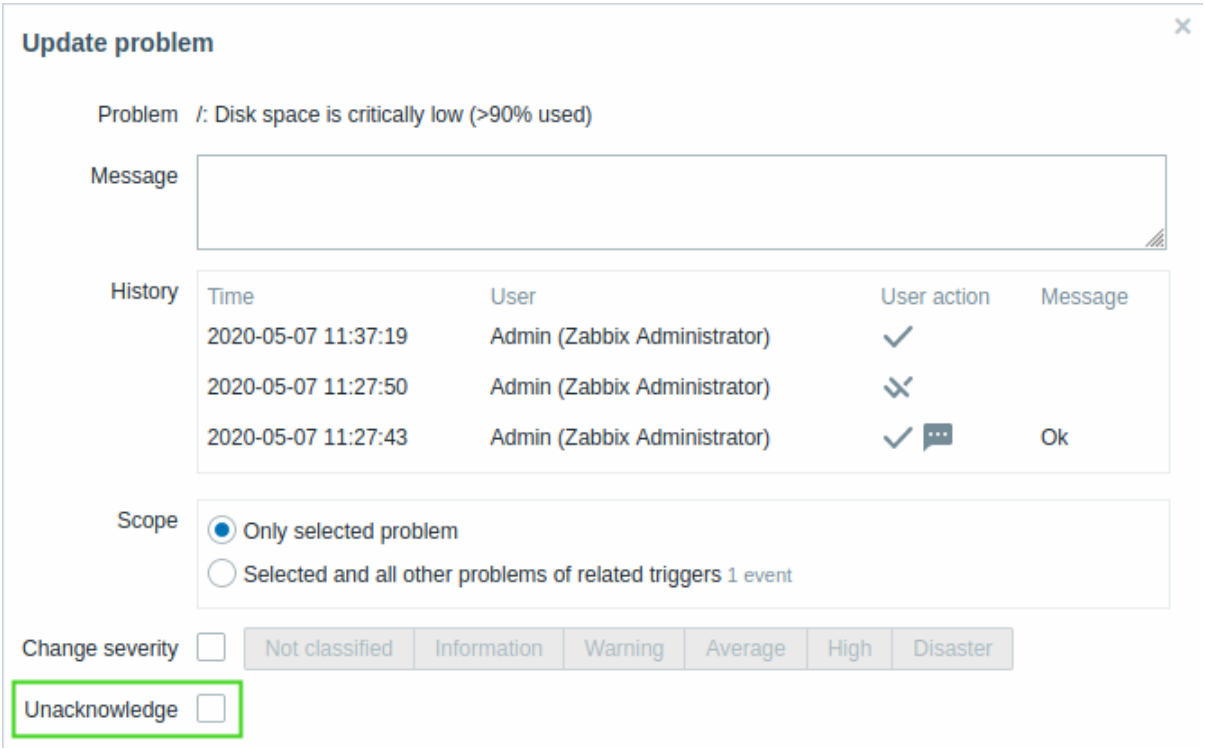
Problem acknowledgment

Several improvements have been made to the problem update screen used for problem acknowledgment and other problem update operations:

- The problem name is displayed (or *N problems selected* if there is more than one problem)
- Size of the problem update message has been increased from 256 to 2048 characters
- It is now possible to unacknowledge problems (see below)

Unacknowledge option

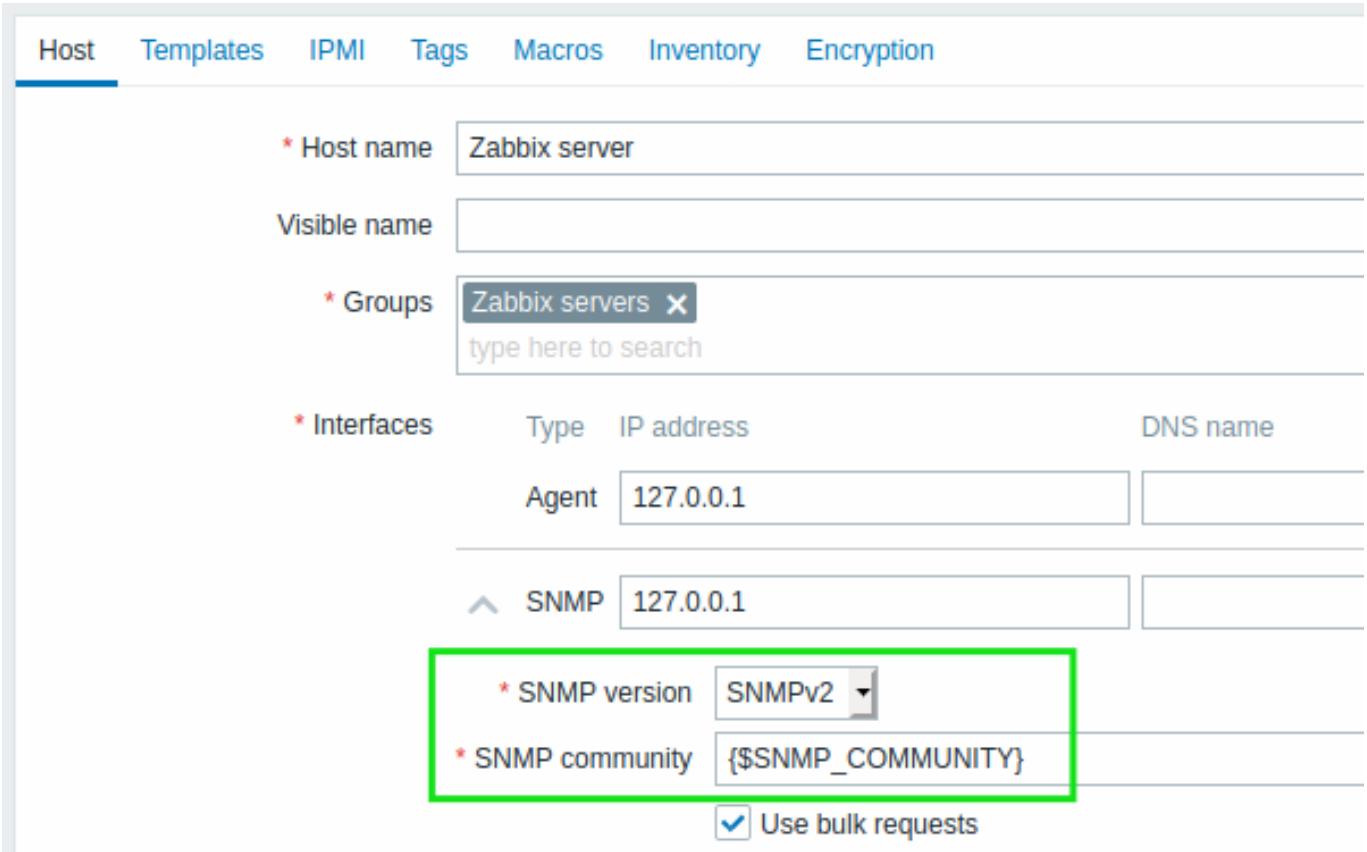
Problems may sometimes be acknowledged by mistake so to remedy this situation it is now also possible to unacknowledge problems. A problem may be unacknowledged in the problem update screen.



In the problem history list unacknowledgment gets a special icon:

SNMP credentials at host interface level

SNMP version and credentials in previous versions were set at an item level. In the new version, all of these can be set at a host interface level:



See also: Configuring SNMP monitoring

When creating an item, the item type dropdown no longer has three entries for SNMP v1, v2 and v3 agent. Instead there is just an *SNMP agent* type and the ability to select the SNMP interface as required.

Manual SNMP cache clearing

Zabbix server and Zabbix proxy now support an `-R snmp_cache_reload` runtime control option, which reloads the SNMP cache

and clears the SNMP properties (engine time, engine boots, engine id, credentials) for all hosts. Net-SNMP version 5.3.0 or higher is required.

Email threading

Email notifications related to the same event are now grouped into one thread.

Elasticsearch 7 support

Elasticsearch version 7.X is now supported. Support of older Elasticsearch versions has been dropped.

SAML authentication

SAML 2.0 authentication is now supported for logging into Zabbix.

Webhook integrations

New integrations are available allowing to use the webhook media type for pushing Zabbix notifications to:

- Jira
- Jira Service Desk
- Microsoft Teams
- Redmine
- ServiceNow
- SIGNL4
- Telegram
- Zammad
- Zendesk

Zabbix agent 2 Zabbix agent 2, first introduced on an experimental basis in Zabbix 4.4, is now officially supported. The functionality of the agent 2 has been extended:

Windows support

Agent 2 can now be compiled from sources on the Windows platform.

Docker monitoring plugin

A Docker plugin for Zabbix agent 2 is now available as part of the out-of-the-box monitoring of Docker containers (see list of supported keys).

Memcached monitoring plugin

A Memcached plugin for Zabbix agent 2 is now available as part of the out-of-the-box monitoring of Memcached instances (see description).

MySQL monitoring plugin

A MySQL plugin for Zabbix agent 2 is now available as part of the out-of-the-box monitoring of MySQL instances (see description).

Agent 2 plugins update

Passing a URI, username and password via plugin configuration parameters is now supported only for named sessions. As such, parameters in a format Plugins.<PluginName>.Uri, Plugins.<PluginName>.User, Plugins.<PluginName>.Password are no longer supported. Named session parameters in a format Plugins.<PluginName>.Sessions.<SessionName>.Uri, Plugins.<PluginName>.Sessions.<SessionName>.Password, Plugins.<PluginName>.Sessions.<SessionName>.User can be used.

Alternatively, a URI, username and password can be provided in the item key parameters directly.

See also:

- Zabbix agent 2
- Plugins
- Building Zabbix agent 2 on Windows

Macros Ability to mask macro content in the frontend

Macro value field now has *Secret text* mode. If enabled, it masks the content of a macro with asterisks to protect sensitive information, such as passwords or shared keys.

Macros supported in host prototypes

User macros can now be defined for host prototypes and LLD macros can be used in the macro value fields (the LLD macro will be resolved when a host is created from the prototype).

User macros supported in IPMI credentials

User macros are now supported in IPMI username and password fields in host configuration.

New macros

The following macros are now supported:

- {EVENT.DURATION} will return the duration of an event.
- {EVENT.TAGSJSON} and {EVENT.RECOVERY.TAGSJSON} macros will resolve to a JSON array containing event tag objects or recovery event tag objects.

For more details, see Macros supported by location.

Updated macros

- {HOST.ID} is now supported in trigger-based notifications and commands, problem update notifications and internal notifications.

Databases Support of IBM DB2 dropped

The IBM DB2 database can no longer be used as a back-end database for Zabbix.

Minimum required versions updated

The minimum required versions for supported databases now are:

- MySQL 5.5.62
- MariaDB 10.0.37
- PostgreSQL 9.2.24
- Oracle 11.2

TimescaleDB native compression support

TimescaleDB native compression is now supported in Zabbix server installations with PostgreSQL version 10.2 or higher and TimescaleDB version 1.5 or higher.

New templates New official templates are available for monitoring:

Elasticsearch

- *Template App Elasticsearch Cluster by HTTP* - native template to monitor Elasticsearch.

ClickHouse

- *Template DB ClickHouse* - collects node metrics from ClickHouse HTTP interface using HTTP agent. (see description).

Memcached

- *Template App Memcached* - Memcached server monitoring via Zabbix agent 2.

MySQL

- *Template DB MySQL by Zabbix agent 2* - DBMS MySQL and its forks monitoring via Zabbix agent 2.

Docker

- *Template App Docker* - Docker monitoring via Zabbix agent 2.

Server

- *Template Server Chassis by IPMI* - server chassis monitoring with BMC over IPMI.

You can get these templates:

- 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).

Items

- The `zabbix[stats,<ip>,<port>]` internal item now also returns version of Zabbix server or Zabbix proxy
- A new `zabbix[version]` internal item has been added returning the version of Zabbix server or Zabbix proxy

Frontend Minimum required PHP version

The minimum required PHP version has been upped from 5.4.0 to 7.2.0.

Support of Internet Explorer 11 dropped

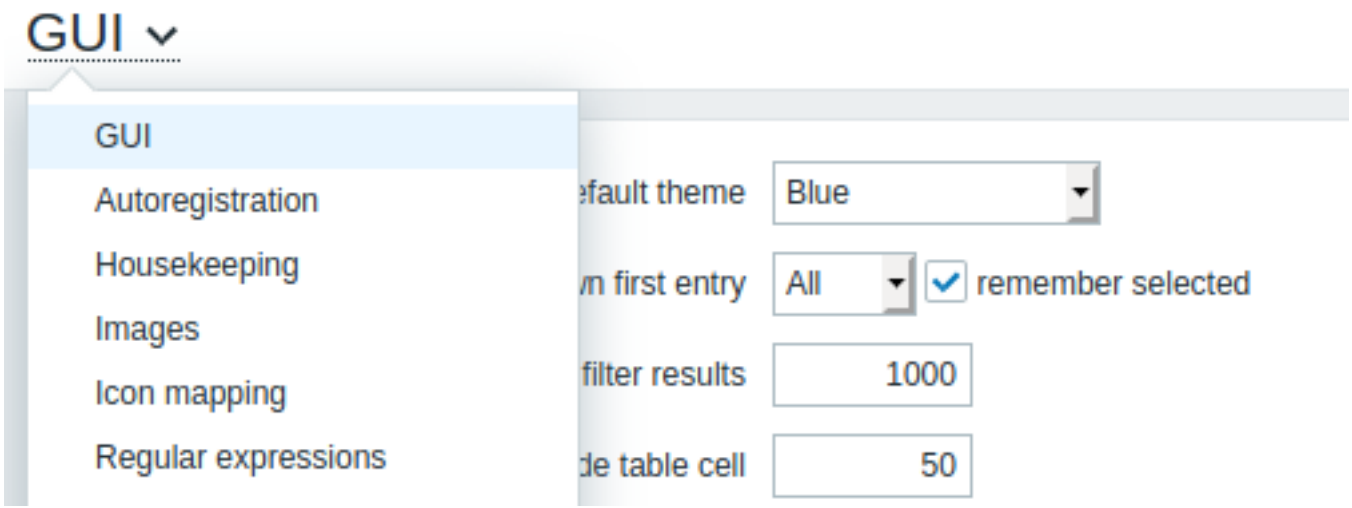
Microsoft Internet Explorer 11 is no longer supported by Zabbix.

Page selection dropdown integrated into headings

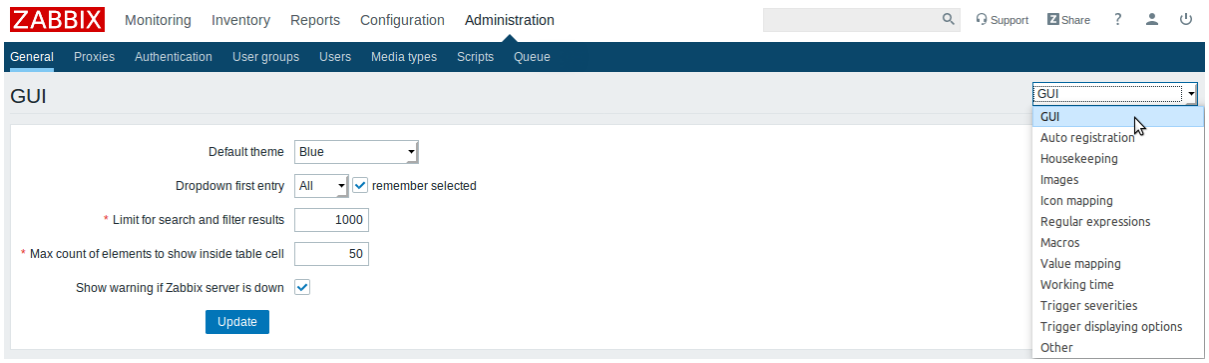
Some frontend sections in Zabbix may display a different page depending on user selection. For example, *Administration → General* may display twelve different pages.

Previously, the page selection was made in a rather small, easy-to-miss dropdown located in the top right corner of the page. Now that selection has been integrated into the headings on the left.

In Zabbix 5.0:



Before Zabbix 5.0:



This change affects the following sections:

- *Monitoring → Overview*
- *Monitoring → Screens*
- *Configuration → Actions*
- *Administration → General*
- *Administration → Queue*

New section for monitoring all hosts

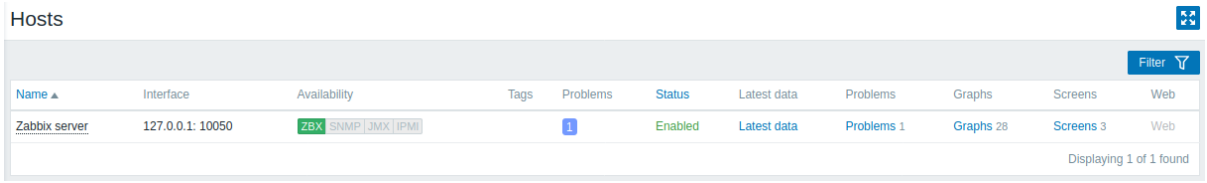
The new frontend section *Monitoring → Hosts* provides detailed view of all monitored devices in a single location. To simplify the navigation *Web* and *Graphs* sections have been removed from the main menu in *Monitoring*. Both sections can now be accessed by clicking on respected links in *Monitoring → Hosts* section.

The following information is available from *Monitoring → Hosts*:

- Hostname
- Main interface
- Availability
- Tags
- Problems (icons indicating currently open problems)

- Status
- Latest data (link to *Latest data* section)
- Problems (number of open problems and a link to *Problems* section)
- Graphs (number of graphs and a link to *Graphs* section)
- Screens (number of screens and a link to *Screens* section)
- Web scenarios (number of web scenarios and a link to *Web* section)

Links in the list above provide a convenient way to view corresponding page with more details about the given host. Users with admin and superadmin rights can also quickly navigate to the host's configuration page from the section. See this page for more details.

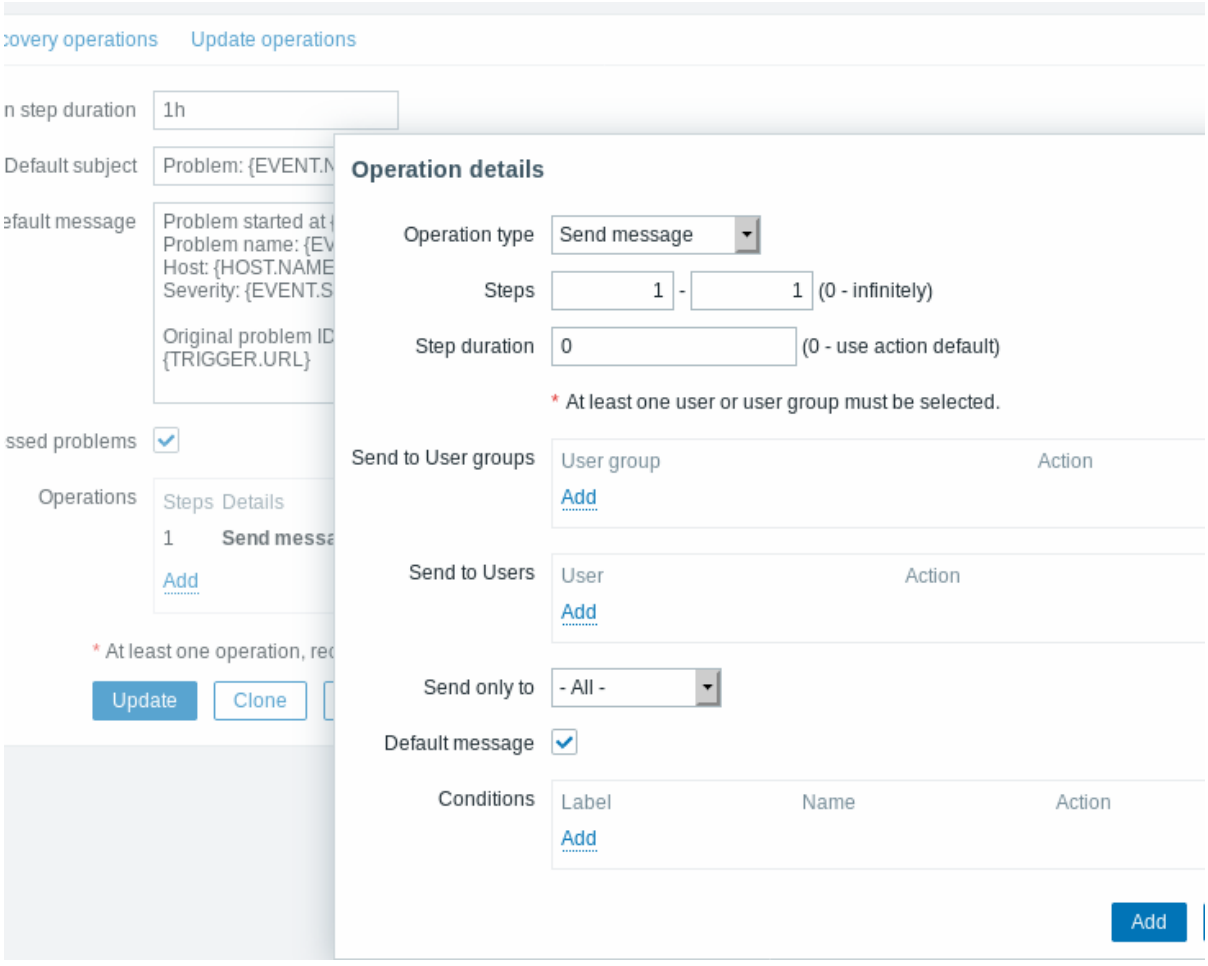


Detail editing as popup window

In several configuration sections of Zabbix frontend detail editing is now opened as a popup window. This is implemented for:

- Action conditions
- Global correlation conditions
- Problem update screen
- Action operation details
  - in the Operations, Recovery operations and Update operations tabs
- Maintenance period details
  - in the Periods tab
- Discovery rule details
  - see discovery check editing

In many cases this change allows to de-clutter the interface of having to configure too many options in one screen. For example, details of an action operation are now opened in a separate popup window.



New filtering options for dashboard widgets

Dashboard widgets Problems by severity and Problem hosts now support filtering problems by tags.

Ability to download graph widgets as images

Screenshots of Graph widget and Graph (classic) widget can now be downloaded as .png files from the widget context menu.

Filtering problems by severities in Monitoring→Problems

Problems displayed in the *Monitoring→ Problems* section can now be filtered by one or several individually selected severities. Previously, there was only filtering by the minimum severity level available.

Webhook media type test usability improved

It is now possible to view log entries during a webhook media type test.

Miscellaneous

- The latest data page no longer displays nothing when opened for the first time.
- The list of web scenario HTTP user agents has been updated.

Daemons Remote command logging on agent

Remote command logging, if enabled on Zabbix agent/agent2 (LogRemoteCommands=1) will no longer create log entries for system.run[] if it is launched locally by HostMetadataItem, HostInterfaceItem or HostnameItem parameters. system.run[] commands will be logged only if executed remotely.

Persistent storage on agent2

Zabbix agent2 is now able to store collected data for active checks in a persistent buffer (disabled by default). The following configuration parameters have been added:

- *EnablePersistentBuffer*
- *PersistentBufferPeriod*
- *PersistentBufferFile*

Crypto libraries

Support of the mbedTLS (PolarSSL) crypto library has been discontinued.

JMX monitoring attributes with tabular data

Support of tabular data objects in JMX Mbean attributes has been added. It is supported for the JMX agent data collection and low-level discovery.

6 What's new in Zabbix 5.0.1

Spiceworks integration

An integration guide with Spiceworks is now available.

OTRS integration

A new webhook integration is available allowing to use webhook media types for pushing Zabbix notifications to OTRS.

Discovery of Windows performance counter instances

It is now possible to discover object instances of Windows performance counters using the new `perf_instance.discovery[]` and `perf_instance_en.discovery[]` items. This is useful for discovering multi-instance performance counters and automated generation of perf_counter and perf_counter_en items (see more information).

PostgreSQL monitoring via agent 2

PostgreSQL plugin and monitoring template are now available for the fast deployment of PostgreSQL monitoring via Zabbix agent 2.

Cache size configuration parameter

The maximum value of the CacheSize configuration parameter for Zabbix server/proxy has been increased from 8GB to 64GB.

More file types by vfs.file.exists[]

The vfs.file.exists[] agent item previously supported only regular files and links. Support for more file types has been added including directories, sockets, block devices, character devices, etc.

Several file types to include can be specified in the second parameter as a quoted, comma-separated list, while file types to exclude similarly can be specified in the third parameter:

```
vfs.file.exists[file,<types_incl,<types_excl>]
```

For more details, see vfs.file.exists[] in agent items.

Logging of global script executions

Information about global script executions is now logged into an *Audit log* with the following details:

- Timestamp
- User that started a script
- IP of the target host or Zabbix server/proxy
- Script after macros substitution (secret macros will be shown as *****)
- Script results

Additionally, there is now an opportunity to pass user information by inserting user-related macros in a script itself. Supported macros:

- {USER.ALIAS} - Zabbix username
- {USER.FULLNAME} - current user's first and last name as specified in Zabbix (username)
- {USER.NAME} - current user's first name as specified in Zabbix
- {USER.SURNAME} - current user's last name as specified in Zabbix

If a script is executed automatically under an action operation, these macros will not be resolved.

Webhooks

The MS Teams webhook now supports custom fields and custom buttons in message cards.

Regular expression support in user macro context

In addition to static strings, regular expressions are now also supported in user macro context, using the following syntax:

```
{$MACRO:regex:"regular expression"}
```

Using regular expressions may significantly reduce the number of user macro contexts you need to define.

See user macros with context for more information.

New templates

Etcd

- *Template App Etcd by HTTP* - collects metrics from Etcd's `/metrics` endpoint with HTTP agent (see description).

Microsoft SQL Server

- *Template DB MSSQL by ODBC* - collects metrics from DBMS Microsoft SQL Server via ODBC (see description).

IIS

- *Template App IIS by Zabbix agent*, *Template App IIS by Zabbix agent active* - collect metrics from Internet Information Services for Windows Server version 2012R2 and newer via Zabbix agent.

You can get these templates:

- In *Configuration* → *Templates* in new installations;
- If you are upgrading from previous versions, you can download these 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.

EnableRemoteCommands deprecated/unsupported by agents

The EnableRemoteCommands agent parameter is now:

- deprecated by Zabbix agent (where it is directly aliased to the corresponding AllowKey/DenyKey parameters)
- unsupported by Zabbix agent2

Use the AllowKey/DenyKey parameters instead.

Templates excluded from host count in System information report

The number of hosts displayed in the *System information* report and widget no longer includes templates. The number of templates is now displayed in a separate row.

Modification time ignored in log, log.count items

File modification time is now ignored in log and log.count items.

Enhanced URL widget security

The URL dashboard widget and the URL screen element now put retrieved URL content into the sandbox. By default, all sandbox restrictions are enabled. It is possible to modify `sandbox` attribute settings in the *defines.inc.php* file, however turning sandboxing off is not recommended for security reasons. To learn more about the sandbox attribute, please see the sandbox section of the iframe HTML element description.

8 What's new in Zabbix 5.0.3

VMware datacenter discovery

A new `vmware.dc.discovery[url]` item returns a JSON containing {#DATACENTER} and {#DATACENTERID} properties.

Host information in VMware event log

The information returned by the `vmware.eventlog[<url>,<mode>]` item now contains information about the source host, if such information is detected in the log.

Service checks on agent 2

Zabbix agent 2 items `net.tcp.service[]` and `net.tcp.service.perf[]` now support HTTPS and LDAP checks on Windows.

List of hosts linked to a template

Configuration → *Templates* section now displays number of linked hosts with a link to the host configuration section filtered by the template. Therefore it is possible to see all hosts connected to the template in a single-click.

New template

Template DB Oracle by ODBC is now available for out-of-the-box monitoring of the Oracle Database. To learn how to configure the template, please see setup instructions.

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.

Latest data

- Expanding/collapsing applications (absent in 5.0.2) has been re-implemented;
- A 'Displaying N to N of N found' (e.g. *Displaying 1 to 50 of 146 found*) string has been added to the page allowing to estimate how many items of the total found are displayed and if items are displayed partially in the page.
- For better page performance, the *Show items without data* filter option is now checked and disabled if no host is selected in the filter.

Monitoring of physical CPU utilization on IBM AIX

Zabbix agent for AIX has been enhanced with capability to monitor physical CPU utilization. A `system.cpu.util[]` item key now has an additional 4th parameter:

- system.cpu.util[<cpu>,<type>,<mode>,**<logical_or_physical>**]

The <logical_or_physical> key parameter allows to specify whether an item should report logical or physical CPU utilization (supported values: `logical`, `physical`). See full item description.

9 What's new in Zabbix 5.0.4

Webhooks New integrations

New integrations are now available allowing to use the webhook media type for pushing Zabbix notifications to:

- iLert
- SolarWinds
- SysAid
- TOPdesk

See the full list of available webhooks.

Support of non-trigger events

In addition to trigger-based notifications, webhooks can now send notifications about discovery, active agent auto-discovery, and internal events.

Retrieving HTTP response headers

It is now possible to retrieve HTTP response headers from the CurlHttpRequest object in webhooks.

Frontend Host menu in web monitoring

The host menu is now available when clicking on the host in the web monitoring page.

Aggregate item helper

The item key selector for aggregate items in the frontend now lists all available aggregate keys (grpavg,grpmax,grpmin,grpsum) and their descriptions.

Oracle monitoring Oracle Database monitoring is now available via the Zabbix agent 2 with a new *Oracle monitoring plugin*. For more information, see:

- Plugin configuration
- Description of supported item keys

A new official template is also available for faster deployment of monitoring:

- *Template DB Oracle by Zabbix agent 2* - Oracle Database monitoring via Zabbix agent 2.

Agent 2 as Windows service Zabbix agent 2 on Windows now can be run as Windows service.

Second precision for age/duration macros Several built-in macros returning age/duration now return the value with down-to-a-second precision:

- {DISCOVERY.DEVICE.UPTIME}
- {DISCOVERY.SERVICE.UPTIME}
- {EVENT.AGE}
- {EVENT.DURATION}
- {ITEM.LOG.AGE}

Previously these macros returned values with down-to-a-minute precision.

10 What's new in Zabbix 5.0.5

DB connection encryption setup

Parameters for encrypting the connection between frontend and the database, available during Zabbix frontend installation, have been modified. The *Configure DB connection* step of Zabbix web interface now features a new checkbox *Verify certificate* with additional options appearing upon marking it. Parameters that are unavailable in the particular configuration will be disabled. For example, the *Database TLS encryption* checkbox cannot be checked if using MySQL and *Database host* is set to localhost. See Secure connection to the database page for the full description.

Ceph monitoring

Ceph monitoring is now available via the Zabbix agent 2 with a new *Ceph plugin*. For more information, see:

- Plugin configuration
- Description of supported item keys

A new official template *Ceph by Zabbix Agent 2* is also available for faster deployment of monitoring.

New templates

The following templates are now available for out-of-the-box monitoring:

- *Template App Ceph by Zabbix agent 2* - see setup instructions for Zabbix agent 2 templates.
- *Template App PHP-FPM by Zabbix agent* - see setup instructions for Zabbix agent templates.
- *Template App PHP-FPM by HTTP* - see setup instructions for HTTP templates.
- *Template App Squid SNMP* - see description.
- *Template Tel Asterisk by HTTP* - see setup instructions for HTTP templates.

You can get these templates:

- 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.

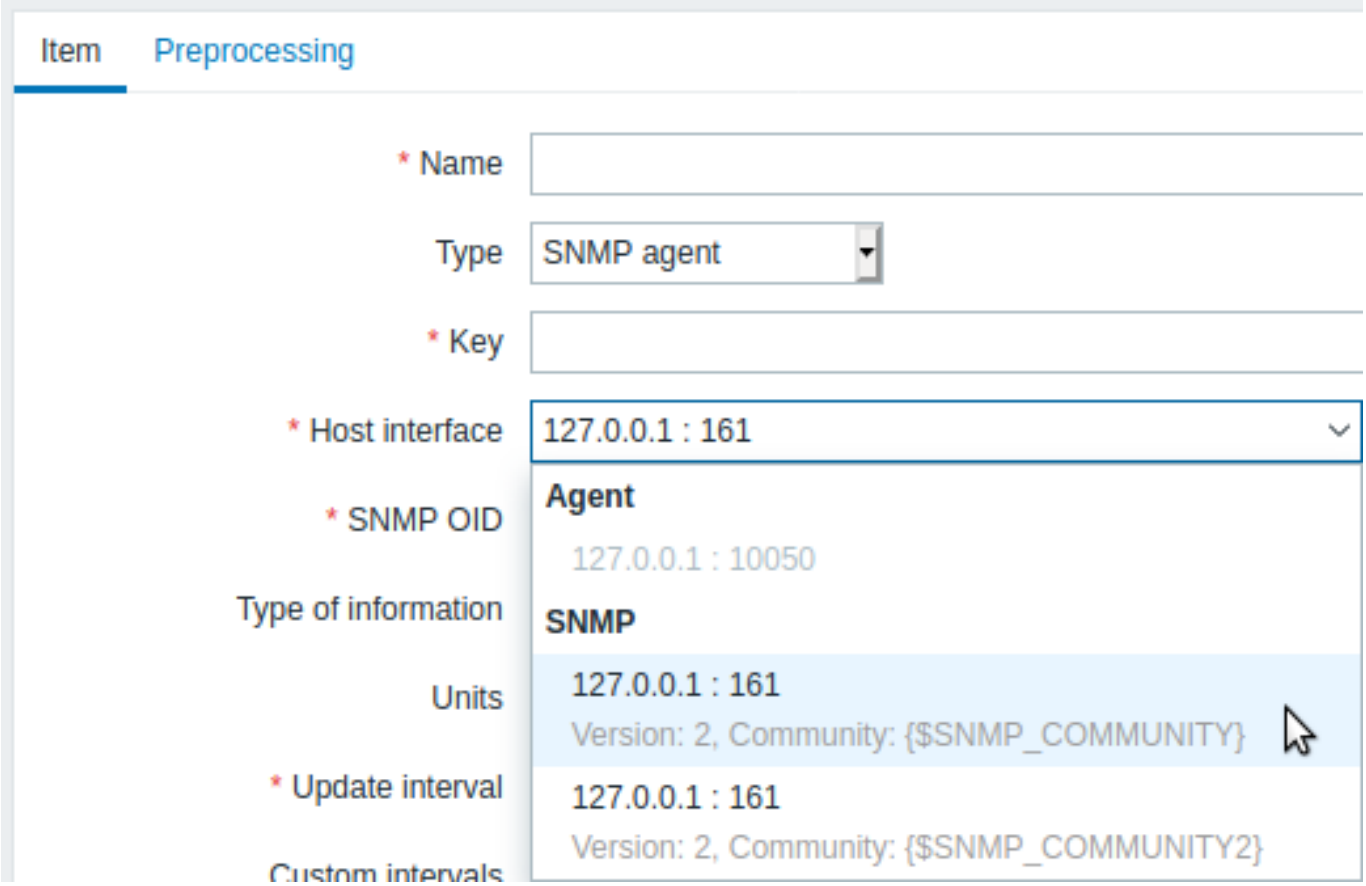
Items

Native support of the **system.swap.size[<device>,<type>]** metric has been added to Zabbix agent 2.

See also: Zabbix agent 2 plugins

SNMP item interface selection

The dropdown for item interface selection has been redesigned visually and it now includes SNMP interface details such as the SNMP version and community string allowing to see a difference between otherwise identical interfaces.



Similarly, such interface details are now also displayed when mass updating items.

Mutex section added to diagnostic information

The 'diaginfo' runtime control option for server/proxy now also returns a section with Zabbix mutexes. You may also run 'diaginfo' to return the mutex section only:

```
zabbix_server -R diaginfo=locks
```

11 What's new in Zabbix 5.0.6

Disabled autocomplete attribute for sensitive fields

To avoid potential exposure of data, the autocomplete attribute is now turned off for many fields containing sensitive information, such as a user's password for logging into Zabbix, pre-shared keys (PSK), usernames and passwords used for data collection by various items and hosts, SNMPv3 authentication and privacy passphrases, passwords for media types; SSL key password and HTTP proxy fields used in web scenarios and HTTP items; usernames, passwords and key passphrases in remote commands. This setting shall prevent most browsers from using autocompletion in the affected fields.

Systemd discovery based on unit state

Zabbix agent 2 item **systemd.unit.discovery** now also returns the current enablement status of unit files under the {#UNIT.UNITFILESTATE} low-level discovery macro. As a use case example, these data may be used in the discovery rule filter to filter out all disabled systemd units and discover only enabled ones.

iTop webhook integration

A new integration is available allowing to use the webhook media type for pushing Zabbix notifications to iTop.

New templates

The following templates are now available for out-of-the-box monitoring:

Apache projects

- *Apache Cassandra by JMX* - see setup instructions for JMX templates;
- *Apache Kafka by JMX* - - see setup instructions for JMX templates;
- *Hadoop by HTTP* - see setup instructions for HTTP templates;
- *ZooKeeper by HTTP* - see setup instructions for HTTP templates.

Morningstar

- *Morningstar ProStar MPPT SNMP* - monitoring of Prostar MPPT solar charge controller via SNMP;
- *Morningstar ProStar PWM SNMP* - monitoring of ProStar pulse width modulation (PWM) solar charge controller via SNMP;
- *Morningstar SunSaver MPPT SNMP* - monitoring of SunSaver MPPT solar charge controller via SNMP;
- *Morningstar SureSine SNMP* - monitoring of SureSine pure sine wave inverter via SNMP;
- *Morningstar TriStar MPPT 600V SNMP* - monitoring of TriStar MPPT 600V solar charge controller via SNMP;
- *Morningstar TriStar MPPT SNMP* - monitoring of TriStar MPPT solar charge controller via SNMP;
- *Morningstar TriStar PWM SNMP* - monitoring of TriStar PWM solar charge controller via SNMP.

These templates are specifically designed for monitoring Morningstar devices; a step-by-step User guide to setting up Zabbix monitoring of the Morningstar products is also available.

You can get new templates:

- 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.

Protection against user enumeration attacks

To make sure an attacker may not guess valid user names because temporary account blocking after consecutive failed login attempts is only applied to existing user names, the account blocking is now also enforced if non-existing user names are used.

To further obscure the possibility of such attacks, a unified generic message is now displayed for all problems related to incorrect login:

`Incorrect user name or password or account is temporarily blocked.`

12 What's new in Zabbix 5.0.7

Items

A new `systemd.unit.get[<unit name>,<interface>]` item allows to retrieve all properties of a systemd unit. This item is supported by Zabbix agent 2 on Linux platforms only.

Custom multiplier preprocessing step

Custom multiplier preprocessing step now accepts strings with macros (a macro must resolve as an integer or a floating-point number).

Java gateway property file

A new `zabbix.propertiesFile` configuration parameter allows to specify a property file, which can be used to set additional properties in such a way that they are not visible on a command line or to overwrite existing ones.

Zabbix agent 2 plugins

Parameter structure for Zabbix agent 2 plugins has been updated:

- For Ceph, Docker, Memcached, MySQL, Oracle, and Redis plugins, a Uri can now only be specified as a named session parameter.
- For PostgreSQL plugin a Uri and Database name can now only be specified as a named session parameter.
- For Uri parameters, specifying a scheme is no longer mandatory.

See also: Zabbix agent 2 (UNIX), Zabbix agent 2 (Windows)

Miscellaneous

- Zabbix agent for Windows now substitutes SIDs with account name and domain name in event log messages.

13 What's new in Zabbix 5.0.8

New templates

The following templates are now available for out-of-the-box monitoring:

- *Apache ActiveMQ by JMX* - see setup instructions for JMX templates;
- *Microsoft Exchange Server 2016 by Zabbix agent*, *Microsoft Exchange Server 2016 by Zabbix agent active* - see setup instructions for Zabbix agent templates;
- *NetApp FAS3220 SNMP* - monitoring of NetApp FAS3220 via SNMP.

You can get these templates:

- 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.

Webhook integrations

- New integration is available allowing to use the webhook media type for pushing Zabbix notifications to Rocket.Chat.
- In the Microsoft Teams webhook, hardcoded teams_endpoint check has been removed to prevent URL syntax error.

PostgreSQL plugin

- PostgreSQL plugin for Zabbix agent 2 now supports custom queries;
- Unix socket support has been fixed.

RHEL/CentOS packages with PHP 7.3

New packages for installing the frontend on RHEL/CentOS 7 with support for PHP 7.3 software collections are now available.

14 What's new in Zabbix 5.0.9

S.M.A.R.T. plugin for agent 2

A plugin for S.M.A.R.T. monitoring has been added out-of-the-box to Zabbix agent 2.

VictorOps webhook integration

A new integration is available allowing to use the webhook media type for pushing Zabbix notifications to VictorOps.

HTTP authentication in webhooks

HTTP authentication is now possible in webhooks using the new `SetHttpAuth(bitmask, username, password)` method in the CurlHttpRequest object of the JavaScript code.

JavaScript functions for MD5 and SHA256 calculation

The functions for MD5 and SHA256 hash calculation have been added to global JavaScript functions.

JavaScript memory limit

The memory limit available for JavaScript scripts has been raised from 10MB to 64MB.

New templates

The following template is now available for out-of-the-box monitoring:

- *Ignite by JMX* - see setup instructions for JMX 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.

15 What's new in Zabbix 5.0.10

MongoDB monitoring

MongoDB monitoring is now available via the Zabbix agent 2 with a new *MongoDB plugin* and monitoring templates. For more information, see:

- Plugin configuration
- Description of supported item keys

brevis.one webhook integration

A new integration is available allowing to use the webhook media type for pushing Zabbix notifications to brevis.one.

New templates

The following templates are now available for out-of-the-box monitoring:

APC UPS

- *APC UPS SNMP* - monitoring of APC UPS Symmetra LX by Zabbix SNMP agent.

Cisco Catalyst 3750 Series

- *Cisco Catalyst 3750V2-24FS SNMP*
- *Cisco Catalyst 3750V2-24PS SNMP*
- *Cisco Catalyst 3750V2-24TS SNMP*
- *Cisco Catalyst 3750V2-48TS SNMP*
- *Cisco Catalyst 3750V2-48TS SNMP*

See also: Standardized templates for network devices.

Huawei Oceanstor

Huawei OceanStor 5300 V5 SNMP - monitoring of SAN Huawei OceanStor 5300 V5 by Zabbix SNMP agent.

NetApp

Template SAN NetApp AFF A700 by HTTP - see setup instructions for HTTP templates.

S.M.A.R.T. monitoring system

- *SMART by Zabbix agent 2*
- *SMART by Zabbix agent 2 active*

See setup instructions for Zabbix agent 2 templates.

MongoDB

- *MongoDB cluster by Zabbix agent 2*
- *MongoDB node by Zabbix agent 2*

See setup instructions for Zabbix agent 2 templates.

You can get these templates:

- 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.

Real-time export

It is now possible to specify entity types to export in real-time export, using the new ExportType server parameter.

Server protected against proxy data overload

Additional protection has been added to Zabbix server against overloading by proxy data, for example, after a server downtime. This is achieved by rotating proxies during the fetching of historical data and updating the history cache.

Timescale DB 2.X

Timescale DB 2.X versions are now officially supported in addition to Timescale DB 1.X. Please note that Timescale DB 2.X works only with PostgreSQL 11 or newer.

15 What's new in Zabbix 5.0.11

Improved Slack webhook

JavaScript in Slack webhook has been updated to ensure the correct work of the webhook. Additionally, proxy support has been added, and the webhook's documentation has been updated and now describes an up-to-date Slack integration setup procedure.

New templates

The following templates are now available for out-of-the-box monitoring:

APC UPS

- *APC UPS Galaxy 3500 SNMP* - monitoring of APC UPS Galaxy 3500 by Zabbix SNMP agent LX by Zabbix SNMP agent;
- *APC Smart-UPS 2200 RM SNMP* - monitoring of APC Smart-UPS 2200 RM by Zabbix SNMP agent;
- *APC Smart-UPS 3000 XLM SNMP* - monitoring of APC Smart-UPS 3000 XLM by Zabbix SNMP agent;
- *APC Smart-UPS RT 1000 RM XL SNMP* - monitoring of APC Smart-UPS RT 1000 RM XL by Zabbix SNMP agent;
- *APC Smart-UPS RT 1000 XL SNMP* - monitoring of APC Smart-UPS RT 1000 XL by Zabbix SNMP agent;
- *APC Smart-UPS SRT 5000 SNMP* - monitoring of APC Smart-UPS SRT 5000 by Zabbix SNMP agent;
- *APC Smart-UPS SRT 8000 SNMP* - monitoring of APC Smart-UPS SRT 8000 by Zabbix SNMP agent;
- *APC UPS SNMP* - monitoring of APC UPS with NMC by Zabbix SNMP agent;
- *APC UPS Symmetra RM SNMP* - monitoring of APC UPS Symmetra RM by Zabbix SNMP agent;
- *APC UPS Symmetra RX SNMP* - monitoring of APC UPS Symmetra RX by Zabbix SNMP agent.

The template *APC UPS SNMP* introduced in Zabbix 5.0.10 for monitoring APC UPS Symmetra LX has been renamed to *APC UPS Symmetra LX SNMP* and updated with new metrics, triggers, a discovery rule, etc., which allow to monitor battery capacity and phase output voltage and load.

You can get these templates:

- 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.

Value cache performance

The value cache performance has been improved by replacing mutexes with read-write locks in the processing logic. The expected benefit of this change is reduced history syncer usage in heavy parallel workloads.

16 What's new in Zabbix 5.0.12

WildFly monitoring

New templates *WildFly Domain by JMX* and *WildFly Server by JMX* are now available allowing to monitor WildFly Domain Controller and WildFly server via JMX. See JMX template operation for setup instructions.

You can get these templates:

- 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.

Webhook integrations

New integrations are available allowing to use the webhook media type for pushing Zabbix notifications to:

- Express.ms messenger
- ManageEngine ServiceDesk

Real-time export

Severity is now exported for trigger events in real-time export.

17 What's new in Zabbix 5.0.13

New templates

The following templates are now available for out-of-the-box monitoring:

Cisco

- *Cisco UCS Manager SNMP* - monitoring of Cisco UCS Manager via SNMP.

DELL PowerEdge

- *DELL PowerEdge R720 by HTTP* - monitoring of DELL PowerEdge R720 servers with iDRAC 8/9 firmware 4.32 and later with Redfish API enabled via HTTP (see setup instructions);
- // DELL PowerEdge R720 SNMP// - monitoring of DELL PowerEdge R720 servers with iDRAC version 7 and later via SNMP;
- *DELL PowerEdge R740 by HTTP* - monitoring of DELL PowerEdge R740 servers with iDRAC 8/9 firmware 4.32 and later with Redfish API enabled via HTTP (see setup instructions);
- *DELL PowerEdge R740 SNMP* - monitoring of DELL PowerEdge R740 servers with iDRAC version 7 and later via SNMP;
- *DELL PowerEdge R820 by HTTP* - monitoring of DELL PowerEdge R820 servers with iDRAC 8/9 firmware 4.32 and later with Redfish API enabled via HTTP (see setup instructions);
- *DELL PowerEdge R820 SNMP* - monitoring of DELL PowerEdge R820 servers with iDRAC version 7 and later via SNMP;
- *DELL PowerEdge R840 by HTTP* - monitoring of DELL PowerEdge R840 servers with iDRAC 8/9 firmware 4.32 and later with Redfish API enabled via HTTP (see setup instructions);
- *DELL PowerEdge R840 SNMP* - monitoring of DELL PowerEdge R840 servers with iDRAC version 7 and later via SNMP.

HP ProLiant

- *HPE ProLiant BL460 SNMP* - monitoring of HPE ProLiant BL460 servers with HP iLO version 4 and later via SNMP;
- *HPE ProLiant BL920 SNMP* - monitoring of HPE ProLiant BL920 servers with HP iLO version 4 and later via SNMP;
- *HPE ProLiant DL360 SNMP* - monitoring of HPE ProLiant DL360 servers with HP iLO version 4 and later via SNMP;
- *HPE ProLiant DL380 SNMP* - monitoring of HPE ProLiant DL380 servers with HP iLO version 4 and later via SNMP.

Nginx Plus

- *NGINX Plus by HTTP* - see setup instructions for HTTP templates.

Zyxel

- *AAM1212-51 IES-612 SNMP* - monitoring of Zyxel AAM1212-51 / IES-612 via SNMP;
- *ES3500-8PD SNMP* - monitoring of Zyxel ES3500-8PD via SNMP;
- *GS-4012F SNMP* - monitoring of Zyxel GS-4012F via SNMP;
- *IES-500x SNMP* - monitoring of Zyxel IES-500x via SNMP;
- *IES1248-51 SNMP* - monitoring of Zyxel IES1248-51 via SNMP;
- *MES-3528 SNMP* - monitoring of Zyxel MES-3528 via SNMP;
- *MES3500-10 SNMP* - monitoring of Zyxel MES3500-10 via SNMP;
- *MES3500-24 SNMP* - monitoring of Zyxel MES3500-24 via SNMP;
- *MGS-3712 SNMP* - monitoring of Zyxel MGS-3712 via SNMP;
- *MGS-3712F SNMP* - monitoring of Zyxel MGS-3712F via SNMP;
- *MES3500-24S SNMP* - monitoring of Zyxel MES3500-24S via SNMP;
- *MGS3520-28x SNMP* - monitoring of Zyxel MGS3520-28x via SNMP;
- *XGS-4728F SNMP* - monitoring of Zyxel XGS-4728F via SNMP.

Updated templates

Zabbix server and *Remote Zabbix server* templates have been updated according to the latest template guidelines.

You can get these templates:

- 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.

Diagnostic information

Diagnostic information about preprocessing has been improved to include the oldest values in queue and the totals to include done, queued, processing and pending.

Default dashboards

The *Zabbix server health* default dashboard widgets now display problems for the selected items only.

Miscellaneous

- In media types, the maximum number of attempts to send an alert has been increased from 10 to 100; the maximum attempt interval has been increased from 60 seconds to 1 hour.
- In web scenario variables and web scenario step variables and post fields, the maximum field size has been increased from 255 to 2000 characters.

18 What's new in Zabbix 5.0.14

Templates

New templates are available:

- *Big-IP SNMP* - monitoring of BIG-IP application services;
- *GridGain by JMX* - see setup instructions for JMX templates;
- *Systemd by Zabbix agent 2* - see setup instructions for Zabbix agent 2 templates.

You can get these templates:

- 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.

Templates *Oracle by ODBC* and *Asterisk by HTTP* have been removed and will no longer be available in Zabbix 5.0 due to compatibility issues. Note that these templates are still available in Zabbix 5.2 and newer.

19 What's new in Zabbix 5.0.15

Plugins

- A new Zabbix agent 2 plugin *WebCertificate* is now available allowing to monitor validity and expiration dates of TLS/SSL website certificates.
- Plugins *MySQL* and *PostgreSQL* now support encrypted TLS connection between the agent and monitored databases. TLS encryption parameters can be provided in the agent configuration file.

See also:

- Plugins
- Zabbix agent 2 configuration parameters (UNIX)
- Zabbix agent 2 configuration parameters (Windows)

New templates

New templates are now available for out-of-the-box monitoring:

- *Cisco ASAv SNMP* - monitoring of Cisco Adaptive Security Virtual Appliance (ASAv) via SNMP;
- *Website certificate by Zabbix agent 2* - native monitoring of TLS/SSL website certificates by Zabbix agent 2.

You can get these templates:

- 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.

Configurable TCP queue maximum size

A new configuration parameter *ListenBacklog* has been added to Zabbix server, Zabbix proxy, and Zabbix agent (Unix/Windows) configuration. This optional parameter can be used to specify the maximum number of pending connections in the TCP queue.

20 What's new in Zabbix 5.0.16

Databases

Support for MariaDB 10.6.X has been added.

Handling large proxy configuration

Protocol has been improved to support Zabbix proxy configuration of size up to 16 GB. Additionally performance and memory usage have been improved by freeing uncompressed data as fast as possible and compressing before connection.

See also protocol header information.

Items

- Zabbix agent 2 items **proc.num**, **proc.cpu.utilization**, **proc.mem** have been updated to use the latest functions introduced in Go 1.16 and thus provide better performance. Go version 1.16 or newer is now required for compiling Zabbix agent 2 to ensure correct work of these items and avoid an issue observed when compiling with older Go versions.
- **net.if.in[]**, **net.if.out[]** and **net.if.total[]** items on Windows now support the network interface GUID as the first parameter, if included in braces.
- **net.if.discovery** on Windows now returns the network interface GUID in the {#IFGUID} macro.
- **system.swap.size[]** now behaves differently if no swap is configured. Now in this case it returns '0' as the value with the *total*, *used*, *free*, and *pused* parameters; it returns '100.0' with *pfree*. Previously in this case the item would return a not supported error with message "Cannot be calculated because swap file size is 0.".

Frontend

- If Zabbix web interface is opened in one of the languages available on the Zabbix website, clicking the Support link will open the Support page in the appropriate language. For all other languages, including English, the Support page will be opened in English.

21 What's new in Zabbix 5.0.17

Items

The **vmware.eventlog[]** item for VMware monitoring, when used with the 'skip' option, e. g. **vmware.eventlog[<url>,skip]** now behaves differently after being recreated (i.e. previous item removed, new one created with a different internal ID) - now the internal events cache is reset and only new events are read. Previously, the skip option would not be enforced in this scenario.

22 What's new in Zabbix 5.0.18

Items

Agent variant check

The new **agent.variant** item returns the variant of Zabbix agent - '1' for Zabbix agent and '2' for Zabbix agent 2.

VMware hypervisor maintenance monitoring

The new **vmware.hv.maintenance[]** item returns '0' when the hypervisor is not in maintenance and '1' when the hypervisor is in maintenance.

VMware system health monitoring

The new **vmware.hv.sensors.get[]** item returns a JSON with various VMware hardware system health data.

Docker container statistics

The **docker.container_stats []** item now also returns CPU usage in percentage as part of container resource usage statistics.

Hostname

The **system.hostname[]** item can now return only shorthost (part of the hostname before the first dot) and, optionally, transform the hostname into lowercase. See Zabbix agent items for details.

New parameter for log*[] and logrt*[] items

The new optional parameter **persistent_dir** specifies a directory for storing a file with a state of **log[]**, **log.count[]**, **logrt[]** or **logrt.count[]** item.

The state includes the log file name, size, position how far the log file was analyzed, MD5 sums for identification of file and a few more attributes.

Its purpose is restoring of more detailed state of the log*[] item in Zabbix agent memory when the agent is started than available from Zabbix server.

It was developed for using in Unix environments with mirrored disks or filesystems supporting snapshots where Zabbix agent can be stopped and later started from a split-off disk copy or a filesystem snapshot.

If Zabbix agent is often stopped/started the new parameter can be also useful on simple filesystems for better protection against analyzing the same records twice or skipping records.

Read the persistent files notes for more information before using it.

Web monitoring

The ability to handle compressed content has been added to Zabbix web monitoring. All encoding formats supported by **libcurl** are supported.

23 What's new in Zabbix 5.0.19

Items

VMware hypervisor discovery

The **vmware.hv.discovery** item now also returns a {#HV.NETNAME} macro.

Security Vulnerability to CVE-2021-42550 has been fixed. As an additional security measure it is recommended to check permissions to the `/etc/zabbix/zabbix_java_gateway_logback.xml` file and set it read-only, if write permissions are available for the "zabbix" user.

24 What's new in Zabbix 5.0.20

Sleep method for JavaScript engine

A new `Zabbix.sleep()` method of Zabbix object has been implemented into JavaScript engine. It will allow to pause JavaScript execution if required. For example, it may be useful in webhooks when working with two separate services, which have a delay between data flows. The method accepts milliseconds as argument, e. g. to delay for 15 seconds:

```
Zabbix.sleep(15000);
```

See also: Additional JavaScript objects

pfSense monitoring template

A new template *pfSense SNMP* is now available for out-of-the-box monitoring.

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.

Item changes

Native support for the items **system.hw.chassis**, **system.hw.devices**, **vfs.dir.count** and **vfs.dir.size** has been added to Zabbix agent 2. These items, used with Zabbix agent 2, now support concurrent check processing.

25 What's new in Zabbix 5.0.21

Zabbix agent 2 items

- Zabbix agent 2 items *smart.disk.discovery* and *smart.attribute.discovery* items, supported for S.M.A.R.T. plugin, have been updated and now return {#DISKTYPE} macro value in the lower case.
- Native support for the items **net.dns** and **net.dns.record** has been added to Zabbix agent 2. These items, used with Zabbix agent 2, now support concurrent check processing. On Windows, custom DNS IP addresses are allowed in the ip parameter, timeout and count parameters are no longer ignored.

SourceIP support in LDAP simple checks SourceIP support has been added to LDAP simple checks. *Note* that with OpenLDAP, version 2.6.1 or above is required.

26 What's new in Zabbix 5.0.22

PostgreSQL metrics

A new item has been added to PostgreSQL plugin for Zabbix agent 2. The metric **pgsql.queries** is used for monitoring query execution time.

Templates The following templates have been updated:

- The template *Generic Java JMX* now contains discovery rules for low-level discovery of memory pools and garbage collectors.
- The templates *Template Module Windows services by Zabbix agent* and *Template Module Windows services by Zabbix agent active* now contain a new macro {$SERVICE.NAME.NOT_MATCHES.EXTENDED}, which is used to filter out additional services.
- The template *PostgreSQL by Zabbix agent 2* now will check the number of slow queries and generate a problem if the amount exceeds a threshold.

You can get these templates:

- 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).

Keyboard navigation Keyboard control has been implemented for info icons in the frontend. Thus it is now possible to focus on info icons, and open the hints, using the keyboard.

27 What's new in Zabbix 5.0.23

S.M.A.R.T. monitoring *Smart plugin*, supported for Zabbix agent 2, now provides more efficient disk discovery and allows returning information about a specific disk, instead of all discovered disks. Zabbix agent 2 items **smart.disk.discovery** and **smart.disk.get** have been updated. The templates *SMART by Zabbix agent 2* and *SMART by Zabbix agent 2 (active)* have also been modified to incorporate the new functionality.

Templates New macros allowing to define warning and critical thresholds of the filesystem utilization for virtual file system monitoring have been added to the templates *Template App PFSense SNMP*, *Template Module HOST-RESOURCES-MIB storage SNMP*, *Template Module Linux filesystems SNMP*, *Template Module Linux filesystems by Zabbix agent active*, *Template Module Linux filesystems by Zabbix agent*, *Template Module Windows filesystems by Zabbix agent active*, *Template Module Windows filesystems by Zabbix agent*, *Template Net Mellanox SNMP*, *Template OS Linux by Prom*. Filesystem utilization triggers have been updated to use these macros.

You can get these templates:

- 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.

28 What's new in Zabbix 5.0.24

Frontend languages German, Greek, Romanian, Spanish and Vietnamese languages are now enabled in the frontend.

29 What's new in Zabbix 5.0.25

ExpressMS messenger webhook API changed API version changed to v4 in ExpressMS messenger webhook.

30 What's new in Zabbix 5.0.26

TimescaleDB 2.6 and 2.7 support TimescaleDB 2.6 and 2.7 are now supported. The maximum supported version for TimescaleDB is now 2.7.

Updated templates PostgreSQL Agent 2 template updated.

A trigger for detecting checksum failures has been added to the Dbstat item of the PostgreSQL Agent 2 template.

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.

31 What's new in Zabbix 5.0.27

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.

New templates

A new template is now available for out-of-the-box monitoring:

- *Template App Ceph by Zabbix agent 2* - see setup instructions for Zabbix agent 2 templates.
- *Template App PHP-FPM by Zabbix agent* - see setup instructions for Zabbix agent templates.
- *Template App PHP-FPM by HTTP* - see setup instructions for HTTP templates.
- *Template App Squid SNMP* - see description.
- *Template Tel Asterisk by HTTP* - see setup instructions for HTTP templates.
- *Template App OPNsense SNMP*.

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.

32 What's new in Zabbix 5.0.28

OpenSSL 3.0 support OpenSSL 3.0.x is now supported. Note that this change does not affect frontend encryption (which uses its own openssl-php package) and Java gateway JMX encrypted connections to monitoring targets (which uses its own Java encrypted libraries).

33 What's new in Zabbix 5.0.29

TimescaleDB 2.8 support The maximum supported version for TimescaleDB is now 2.8.

Frontend Miscellaneous

- Warnings about incorrect housekeeping configuration for TimescaleDB are now displayed if history or trend tables contain compressed chunks, but *Override item history period* or *Override item trend period* options are disabled. For more information, see TimescaleDB setup.

34 What's new in Zabbix 5.0.30

Reporting file systems with zero inodes `vfs.fs.get` agent items are now capable of reporting file systems with the inode count equal to zero, which can be the case for file systems with dynamic inodes (e.g. btrfs).

Additionally `vfs.fs.inode` items now will not become unsupported in such cases with mode set to 'pfree' or 'pused'. Instead the pfree/pused values for such file systems will be reported as "100" and "0" respectively.

35 What's new in Zabbix 5.0.31

TimescaleDB 2.9 support The maximum supported version for TimescaleDB is now 2.9.

Improved performance of history syncers The performance of history syncers has been improved by introducing a new read-write lock. This reduces locking between history syncers, trappers and proxy pollers by using a shared read lock while accessing the configuration cache. The new lock can be write locked only by the configuration syncer performing a configuration cache reload.

Patch for API queries optimization Optional patches are now available for optimizing API database queries, used when searching through names in *hosts* and *items* tables. To apply the patches, run the file *index_host_and_item_name_upper_field.sql* manually. Note that database settings must allow creation of deterministic triggers for the time of patch application. See Known issues for instructions how to apply the patch.

Updated templates *Template DB Oracle by Zabbix agent 2* has been updated (multiple static items removed; multiple item prototypes added) according to the changes made to multiple Zabbix agent 2 items.

For more information about the updates, see Template changes.

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.

Query separate tablespaces in Oracle databases with Zabbix agent 2 The following Zabbix agent 2 items, supported for the Oracle plugin, now have additional optional parameters:

- oracle.diskgroups.stats[<existingParameters>,**<diskgroup>**]
- oracle.archive.info[<existingParameters>,**<destination>**]
- oracle.cdb.info[<existingParameters>,**<database>**]
- oracle.pdb.info[<existingParameters>,**<database>**]
- oracle.ts.stats[<existingParameters>,**<tablespace>**,**<type>**]

These parameters allow to query separate instances of data instead of all data, thus improving performance.

Retrieving additional information with `docker.container_info[]` The `docker.container_info[]` Zabbix agent 2 item now supports the option to retrieve either partial (short) or full low-level information about a Docker container.

36 What's new in Zabbix 5.0.32

Limits for JavaScript objects in preprocessing The following limits for JavaScript objects in preprocessing have been introduced:

- The total size of all messages that can be logged with the `Log()` method has been limited to 8 MB per script execution.
- The initialization of multiple `CurlHttpRequest` objects has been limited to 10 per script execution.
- The total length of header fields that can be added to a single `CurlHttpRequest` object with the `AddHeader()` method has been limited to 128 Kbytes (special characters and header names included).

37 What's new in Zabbix 5.0.33

TimescaleDB 2.10 support The maximum supported version for TimescaleDB is now 2.10.

Connection options for Oracle plugin Oracle plugin, supported for Zabbix agent 2, now allows to specify `as sysdba`, `as sysoper`, or `as sysasm` login option. The option can be appended either to the user item key parameter or to the plugin configuration parameter Plugins.Oracle.Sessions.<SessionName>.User in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).

38 What's new in Zabbix 5.0.34

Mixing item key and session parameters in Zabbix agent 2 plugins Zabbix agent 2 now allows to override named session parameters by specifying new values in the item key parameters. Previously, users had to select if they prefer to provide connection string values in a named session or in an item key. If a named session has been used, related item key parameters had to be empty. Now, if using named sessions, only the first parameter (usually, a URI) has to be specified in the named session, whereas other parameters can be defined either in the named session or in the item key.

39 What's new in Zabbix 5.0.35

Default values for Zabbix agent 2 Zabbix agent 2 plugins now allow to define default values for connecting to monitoring targets in the configuration file. If no value is specified in an item key or a named session, the plugin will use the value defined in the corresponding default parameter. New parameters have the structure `Plugins.<PluginName>.Default.<Parameter>` - for example, `Plugins.MongoDB.Default.Uri=tcp://localhost:27017`. See for more info:

- Configuring plugins
- Plugin configuration file parameters

40 What's new in Zabbix 5.0.36

TimescaleDB 2.11 support Support for TimescaleDB version 2.11 is now available.

41 What's new in Zabbix 5.0.37

This minor version does not have any functional changes.

42 What's new in Zabbix 5.0.38

This minor version does not have any functional changes.

43 What's new in Zabbix 5.0.39

This minor version does not have any functional changes.

44 What's new in Zabbix 5.0.40

TimescaleDB 2.12 support

Support for TimescaleDB version 2.12 is now available.

45 What's new in Zabbix 5.0.41

TimescaleDB 2.13 support

Support for TimescaleDB version 2.13 is now available.

MySQL 8.2 support

The maximum supported version for MySQL is now 8.2.X.

46 What's new in Zabbix 5.0.42

MySQL 8.3 support

The maximum supported version for MySQL is now 8.3.X.

MariaDB 11.2 support

The maximum supported version for MariaDB is now 11.2.X.

TimescaleDB 2.14 support

The maximum supported version for TimescaleDB is now 2.14.X.

Zabbix agent 2 support on Windows

To prevent critical security vulnerabilities, the minimum Windows version for Zabbix agent 2 has been raised to Windows 10/Server 2016. See note under Supported platforms for more information.

47 What's new in Zabbix 5.0.43

MySQL 9.0 support

The maximum supported version for MySQL is now 9.0.X.

MariaDB 11.4 support

The maximum supported version for MariaDB is now 11.4.X.

TimescaleDB 2.15 support

The maximum supported version for TimescaleDB is now 2.15.X.

Frontend languages

Catalan language is now enabled in the frontend.

GSM modem validation for SMS media type

In SMS media type configuration, the GSM modem path is now validated to be a modem device or symlink to such.

48 What's new in Zabbix 5.0.44

Databases TimescaleDB 2.16 support

The maximum supported version for TimescaleDB is now 2.16.X.

MariaDB 11.5 support

The maximum supported version for MariaDB is now 11.5.X.

49 What's new in Zabbix 5.0.45

TimescaleDB 2.17 support

The maximum supported version for TimescaleDB is now 2.17.X.

PostgreSQL 17 support

PostgreSQL 17 is now supported.

50 What's new in Zabbix 5.0.46

See breaking changes for this version.

51 What's new in Zabbix 5.0.47

This minor version does not have any functional changes.

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; it may contain hosts and templates. Hosts and templates within a host group are not in any way linked to each other. 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 from 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, screens, applications, 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.

application

- *a grouping of items in a logical group.*

web scenario

- *one or several HTTP requests to check the availability of a web site.*

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 from 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.*

> **Attention:**
> 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, application 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> systemctl start zabbix-server
```

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> systemctl stop zabbix-server
shell> systemctl restart zabbix-server
shell> systemctl status zabbix-server
```

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.co
-f --foreground                 run Zabbix server in foreground
-R --runtime-control <option>   perform administrative functions
-h --help                       give this help
-V --version                    display version number
```

Note:
Runtime control is not supported on OpenBSD and NetBSD.

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:

Option	Description	Target
config_cache_reload	Reload configuration cache. Ignored if cache is being currently loaded.	
diaginfo[=<**target**>]	Gather diagnostic information in the server log file. Available since Zabbix 5.0.4.	**historycache** - history cache statistics **valuecache** - value cache statistics **preprocessing** - preprocessing manager statistics **alerting** - alert manager statistics **lld** - LLD manager statistics **locks** (since 5.0.5) - list of mutexes
snmp_cache_reload	Reload SNMP cache, clear the SNMP properties (engine time, engine boots, engine id, credentials) for all hosts.	
housekeeper_execute	Start the housekeeping procedure. Ignored if the housekeeping procedure is currently in progress.	
log_level_increase[=<**target**>]	Increase log level, affects all processes if target is not specified.	**process type** - All processes of specified type (e.g., poller) See all server process types. **process type,N** - Process type and number (e.g., poller,3) **pid** - Process identifier (1 to 65535). For larger values specify target as 'process type,N'.
log_level_decrease[=<**target**>]	Decrease log level, affects all processes if target is not specified.	

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 gather diagnostic information:

```
Gather all available diagnostic information in the server log file:
shell> zabbix_server -R diaginfo

Gather history cache statistics in the server log file:
shell> zabbix_server -R diaginfo=historycache
```

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
```

Examples of using runtime control to change log level:

```
Increase log level of all processes:
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"
```

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
- `configuration syncer` - process for managing in-memory cache of configuration data
- `discoverer` - process for discovery of devices
- `escalator` - process for escalation of actions
- `history syncer` - history DB writer
- `housekeeper` - process for removal of old historical data
- `http poller` - web monitoring poller
- `icmp pinger` - poller for icmpping 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
- `poller` - normal poller for passive checks
- `preprocessing manager` - manager of preprocessing tasks
- `preprocessing worker` - process for data preprocessing
- `proxy poller` - poller for passive proxies
- `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

Note:
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.

2 Agent

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 on the following platforms:

- Windows (all desktop and server versions since XP)
- Linux (also available in distribution packages)
- macOS
- IBM AIX
- FreeBSD
- OpenBSD
- Solaris

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.

Attention:
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> systemctl start zabbix-agent
```

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> systemctl stop zabbix-agent
shell> systemctl restart zabbix-agent
shell> systemctl status zabbix-agent
```

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:

Parameter	Description
UNIX and Windows agent	
-c --config <config-file>	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
-p --print	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).
-t --test <item key>	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).
-h --help	Display help information
-V --version	Display version number
UNIX agent only	
-R --runtime-control <option>	Perform administrative functions. See runtime control.
Windows agent only	
-m --multiple-agents	Use multiple agent instances (with -i,-d,-s,-x functions). To distinguish service names of instances, each service name will include the Hostname value from the specified configuration file.
Windows agent only (functions)	
-i --install	Install Zabbix Windows agent as service

Parameter	Description
-d --uninstall	Uninstall Zabbix Windows agent service
-s --start	Start Zabbix Windows agent service
-x --stop	Stop Zabbix Windows agent service

Specific **examples** of using command line parameters:

- printing all built-in agent items with values
- testing a user parameter with "mysql.ping" key defined in the specified configuration file
- installing a "Zabbix Agent" service for Windows using the default path to configuration file c:\zabbix_agentd.conf
- 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.

Option	Description	Target
log_level_increase[=<target>]	Increase log level. If target is not specified, all processes are affected.	Target can be specified as: **process type** - all processes of specified type (e.g., listener) See all agent process types. **process type,N** - process type and number (e.g., listener,3) **pid** - process identifier (1 to 65535). For larger values specify target as 'process-type,N'.
log_level_decrease[=<target>]	Decrease log level. If target is not specified, all processes are affected.	

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"
```

Note:
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.

Agent 2 does not have built-in daemonization support on Linux; since Zabbix 5.0.4 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.

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.<Plugin name>.Capacity=N` setting in the *Plugins* configuration parameter.

See also: Plugin development guidelines.

Supported platforms

Zabbix agent 2 is supported on the following platforms:

- Windows (all desktop and server versions since Windows 10/Server 2016) - available as a pre-compiled binary or in Zabbix sources
- Linux - available in distribution packages or in Zabbix sources

Installation

To install Zabbix agent 2, the following options are available:

Windows:

- from a pre-compiled binary - download the binary and follow the instructions on the Windows agent installation from MSI page
- from sources - see Building Zabbix agent 2 on Windows

Linux:

- from distribution packages - follow the instructions on the Zabbix packages page, available by choosing your distribution and the Agent 2 component
- from sources - see Installation from sources; note that you must configure the sources by specifying the `--enable-agent2` configuration option

Options

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

Parameter	Description
-c --config <config-file>	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*
-f --foreground	Run Zabbix agent in foreground (default: true).
-p --print	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).
-t --test <item key>	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).
-h --help	Print help information and exit.
-v --verbose	Print debugging information. Use this option with -p and -t options.
-V --version	Print agent version number and exit.
-R --runtime-control <option>	Perform administrative functions. See runtime control.

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.

Option	Description
log_level_increase	Increase log level. In Zabbix versions 5.0.0-5.0.3 use "loglevel increase" instead (quoted).
log_level_decrease	Decrease log level. In Zabbix versions 5.0.0-5.0.3 use "loglevel decrease" instead (quoted).
metrics	List available metrics.
version	Display agent version.
help	Display help information on runtime control.

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.

New parameters	Description
ControlSocket	The runtime control socket path. Agent 2 uses a control socket for runtime commands.
EnablePersistentBuffer, *PersistentBufferFile*, *PersistentBufferPeriod*	These parameters are used to configure persistent storage on agent 2 for active items.
Plugins	Plugins may have their own parameters, in the format `Plugins.<Plugin name>.<Parameter>=<value>`. A common plugin parameter is *Capacity*, setting the limit of checks that can be executed at the same time.

New parameters	Description
StatusPort	The port agent 2 will be listening on for HTTP status request and display of a list of configured plugins and some internal parameters
Dropped parameters	**Description**
AllowRoot, *User*	Not supported because daemonization is not supported.
LoadModule, *LoadModulePath*	Loadable modules are not supported.
StartAgents	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.
HostInterface, *HostInterfaceItem*	Not yet supported.

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.

Attention:
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> systemctl start zabbix-proxy
```

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

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
```

Note:
Runtime control is not supported on OpenBSD and NetBSD.

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:

Option	Description	Target
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.	
diaginfo[=<**target**>]	Gather diagnostic information in the proxy log file. Available since Zabbix 5.0.4.	**historycache** - history cache statistics **preprocessing** - preprocessing manager statistics **locks** (since 5.0.5) - list of mutexes
snmp_cache_reload	Reload SNMP cache, clear the SNMP properties (engine time, engine boots, engine id, credentials) for all hosts.	
housekeeper_execute	Start the housekeeping procedure. Ignored if the housekeeping procedure is currently in progress.	
log_level_increase[=<**target**>]	Increase log level, affects all processes if target is not specified.	**process type** - All processes of specified type (e.g., poller) See all proxy process types. **process type,N** - Process type and number (e.g., poller,3) **pid** - Process identifier (1 to 65535). For larger values specify target as 'process type,N'.
log_level_decrease[=<**target**>]	Decrease log level, affects all processes if target is not specified.	

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

- `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 icmpping checks
- `ipmi manager` - IPMI poller manager
- `ipmi poller` - poller for IPMI checks
- `java poller` - poller for Java 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 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.

If you are trying to monitor Wildfly-based Java applications with Zabbix Java gateway, please install the latest jboss-client.jar available on the Wildfly download page.

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.

Installation from	Instructions	Instructions
Sources	Installation	Setup
RHEL/CentOS packages	Installation	Setup
Debian/Ubuntu packages	Installation	Setup

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-1.5.16.jar
lib/logback-classic-1.5.16.jar
lib/slf4j-api-2.0.16.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.

> **Warning:**
> 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:

```
# systemctl restart 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:

```
# systemctl restart zabbix-java-gateway
```

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.

See also zabbix_utils - a Python library that has built-in functionality to act like Zabbix sender.

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

Attention:
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 polling time still is 1 second.

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

Running Zabbix sender with low-level discovery

An example of running Zabbix sender for sending a JSON-formatted value for low-level discovery:

```
./zabbix_sender -z 192.168.1.113 -s "Zabbix server" -k trapper.discovery.item -o '[{"{#FSNAME}":"/","{#FST
```

For this to work, the low-level discovery rule must have a Zabbix trapper item type (in this example, with `trapper.discovery.item` key).

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.

See also zabbix_utils - a Python library that has built-in functionality to act like Zabbix get.

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.-]+)\
```

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.
  -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:

```
  -s, --script script-file          Specify the file name of the script to execute. If '-' is specified as
  -i, --input input-file            Specify the file name of the input parameter. If '-' is specified as f
  -p, --param input-param           Specify the input parameter.
  -l, --loglevel log-level          Specify the log level.
  -t, --timeout timeout             Specify the timeout in seconds. Valid range: 1-60 seconds (default: 10
  -h, --help                        Display help information.
  -V, --version                     Display the version number.
```

Example:

```
zabbix_js -s script-file.js -p example
```

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:

```
sudo apt-get update
sudo apt-get install git
```

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. 128 MB of physical memory and 256 MB of free disk space could be a good starting point. However, 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. Amount of memory allocated for the connection depends on configuration of the database engine.

Note:
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 hardware configuration

The table provides several examples of hardware configurations:

Name	Platform	CPU/Memory	Database	Monitored hosts
Small	CentOS	Virtual Appliance	MySQL InnoDB	100
Medium	CentOS	2 CPU cores/2GB	MySQL InnoDB	500
Large	RedHat Enterprise Linux	4 CPU cores/8GB	RAID10 MySQL InnoDB or PostgreSQL	>1000
Very large	RedHat Enterprise Linux	8 CPU cores/16GB	Fast RAID10 MySQL InnoDB or PostgreSQL	>10000

Note:
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 server 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:

Platform	Server	Agent	Agent2
Linux	x	x	x
IBM AIX	x	x	-
FreeBSD	x	x	-

Platform	Server	Agent	Agent2
NetBSD	x	x	-
OpenBSD	x	x	-
HP-UX	x	x	-
Mac OS X	x	x	-
Solaris	x	x	-
Windows	-	x	x

Note:
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.

To prevent critical security vulnerabilities in Zabbix agent 2, it is compiled only with supported Go releases. As of Go 1.21, the minimum required Windows versions are raised; therefore, since Zabbix 5.0.42, the minimum Windows version for Zabbix agent 2 is Windows 10/Server 2016.

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

Required software

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

Database management system

Software	Version	Comments
MySQL	5.7.X-9.0.X	Required if MySQL is used as Zabbix backend database. InnoDB engine is required. MariaDB (10.0.37-11.5.X) also works with Zabbix. For MySQL we recommend using the C API (libmysqlclient) library for building server/proxy. For MariaDB we recommend using the MariaDB Connector/C library for building server/proxy. For MySQL versions 8.0.0-8.0.28 and MariaDB, see also: Possible deadlocks.
Oracle	11.2 or later	Required if Oracle is used as Zabbix backend database.
PostgreSQL	9.2.24-17.X	Required if PostgreSQL is used as Zabbix backend database. Depending on the installation size, it might be required to increase PostgreSQL *work_mem* configuration property (4MB being the default value), so that the amount of memory used by the database for particular operation is sufficient and query execution does not take too much time. Added support for PostgreSQL versions: - 17.X since Zabbix 5.0.45.

Software	Version	Comments
TimescaleDB	For Zabbix 5.0.0-5.0.9: 1.X, OSS (free, under Apache license) version	Required if TimescaleDB is used as a PostgreSQL database extension. Make sure to install TimescaleDB Community Edition, which supports compression.
	Since Zabbix 5.0.10: 1.X, 2.0.1-2.17.X	Note that TimescaleDB 2.0.1-2.9 works only with PostgreSQL 11-14. PostgreSQL 15 is supported since TimescaleDB 2.10. You may also refer to Timescale documentation for details regarding PostgreSQL and TimescaleDB version compatibility. Added support for TimescaleDB versions: - 2.6 and 2.7 since Zabbix 5.0.26; - 2.8 since Zabbix 5.0.29; - 2.9 since Zabbix 5.0.31; - 2.10 since Zabbix 5.0.33; - 2.11 since Zabbix 5.0.36; - 2.12 since Zabbix 5.0.40; - 2.13 since Zabbix 5.0.41; - 2.14 since Zabbix 5.0.42; - 2.15 since Zabbix 5.0.43; - 2.16 since Zabbix 5.0.44; - 2.17 since Zabbix 5.0.45.
SQLite	3.3.5 or later	SQLite is only supported with Zabbix proxies. Required if SQLite is used as Zabbix proxy database.

Frontend

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

Software	Version	Comments
Apache	1.3.12 or later	
PHP	7.2.0 or later	PHP 8.0 is not supported.
PHP extensions:		
gd	2.0.28 or later	PHP GD extension must support PNG images (*--with-png-dir*), JPEG (*--with-jpeg-dir*) images and FreeType 2 (*--with-freetype-dir*). Version 2.3.0 or later might be required to avoid possible text overlapping in graphs for some frontend languages.
bcmath		php-bcmath (*--enable-bcmath*)
ctype		php-ctype (*--enable-ctype*)
libXML	2.6.15 or later	php-xml, if provided as a separate package by the distributor.
xmlreader		php-xmlreader, if provided as a separate package by the distributor.
xmlwriter		php-xmlwriter, if provided as a separate package by the distributor.
session		php-session, if provided as a separate package by the distributor.
sockets		php-net-socket (*--enable-sockets*). Required for user script support.
mbstring		php-mbstring (*--enable-mbstring*)
gettext		php-gettext (*--with-gettext*). Required for translations to work.
ldap		php-ldap. Required only if LDAP authentication is used in the frontend.
openssl		php-openssl. Required only if SAML authentication is used in the frontend.

Software	Version	Comments
mysqli		Required if MySQL is used as Zabbix backend database.
oci8		Required if Oracle is used as Zabbix backend database.
pgsql		Required if PostgreSQL is used as Zabbix backend database.

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

Attention:
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.

Web browser on client side

Cookies and Java Script must be enabled.

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

Warning:
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 screens or dashboards on `http://secure-zabbix.com/zabbix/`, will have full JS access to Zabbix.

Server

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

Requirement	Status	Description
libpcre	Mandatory	PCRE 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'. Note that you need exactly PCRE (v8.x); PCRE2 (v10.x) library is not used.
libevent		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.
libpthread		Required for mutex and read-write lock support.
zlib		Required for compression support.
OpenIPMI	Optional	Required for IPMI support.
libssh2 or *libssh*		Required for SSH checks. Version 1.0 or higher (libssh2); 0.6.0 or higher (libssh). libssh is supported since Zabbix 4.4.6.
fping		Required for ICMP ping items.

Requirement	Status	Description
libcurl		Required for web monitoring, VMware monitoring, SMTP authentication, `web.page.*` Zabbix agent items, 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
libxml2		Required for VMware monitoring and XML XPath preprocessing.
net-snmp		Required for SNMP support. Version 5.3.0 or higher.
GnuTLS, *OpenSSL* or *LibreSSL*		Required when using encryption.

Agent

Requirement	Status	Description
libpcre	Mandatory	PCRE 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'. Note that you need exactly PCRE (v8.x); PCRE2 (v10.x) library is not used.
GnuTLS, *OpenSSL* or *LibreSSL*	Optional	Required when using encryption. On Microsoft Windows systems OpenSSL 1.1.1 or later is required.

Note:
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

Requirement	Status	Description
libpcre	Mandatory	PCRE 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'. Note that you need exactly PCRE (v8.x); PCRE2 (v10.x) library is not used.
OpenSSL	Optional	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.

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:

Library	License	Website
logback-core-1.5.16.jar	EPL 1.0, LGPL 2.1	http://logback.qos.ch/
logback-classic-1.5.16.jar	EPL 1.0, LGPL 2.1	http://logback.qos.ch/
slf4j-api-2.0.16.jar	MIT License	http://www.slf4j.org/
android-json-4.3_r3.1.jar	Apache License 2.0	https://android.googlesource.com/platform/libcore/+/master/json. See src/zabbix_java/lib/README for instructions on creating a JAR file.

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:

Distribution	OpenJDK version
RHEL/CentOS 8	11.0.25
RHEL/CentOS 7	1.8.0
SLES 15	11.0.4
SLES 12	1.8.0
Debian 10	11.0.8
Debian 9	1.8.0
Debian 8	1.7.0
Ubuntu 20.04	11.0.8
Ubuntu 18.04	11.0.8
Ubuntu 16.04	1.8.0
Ubuntu 14.04	1.6.0

Default port numbers

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

Zabbix component	Port number	Protocol	Type of connection
Zabbix agent	10050	TCP	on demand
Zabbix agent 2	10050	TCP	on demand
Zabbix server	10051	TCP	on demand
Zabbix proxy	10051	TCP	on demand
Zabbix Java gateway	10052	TCP	on demand

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 refresh rate of 60 seconds, the number of values per second is calculated as 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, total number of values will be around (**30***24*3600)* **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 * 90 bytes = **10.9GB** of disk space.

Note:
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 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*24*365* **90** = **2.2GB** per year, or **11GB** 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 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 * 100 bytes of disk space.

It means that if we want to keep 3 years of events, this would require 3*365*24*3600* (250+80+#tags*100) = **~30GB**+#tags*100B disk space[2].

Note:
[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:

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

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 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.

> **Attention:**
> 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 an "Admin" user type has less privileges than "Super Admin" user type, it has administrative permissions that allow managing configuration and execute custom scripts.

> **Note:**
> 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 (Windows)

If you have compiled Zabbix agent on Windows, with OpenSSL located in an unprotected directory (e.g., `c:\openssl-64bit`, `C:\OpenSSL-Win64-111-static`, or `C:\dev\openssl`), make sure to revoke write access from non-administrator users to this directory. Otherwise, the agent loads SSL settings from a path that can be modified by unprivileged users, resulting in a potential security vulnerability.

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 -
```

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.

```
Country Name (2 letter code) [XX]:
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:example.com
```

```
Email Address []:
```

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:

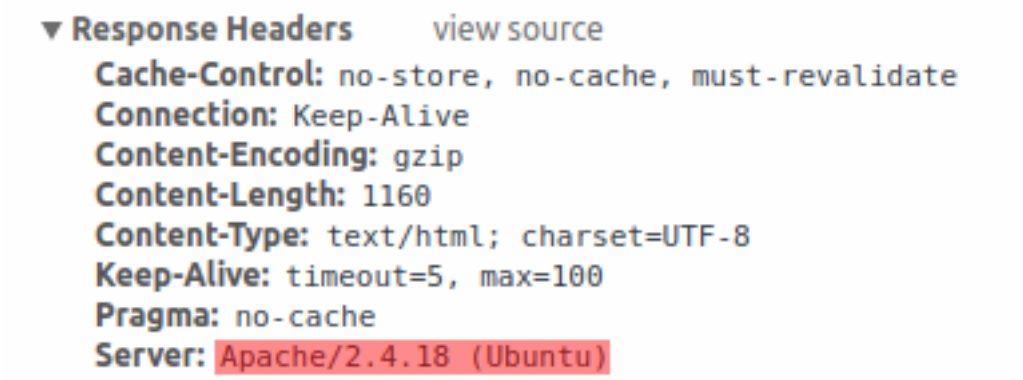
```
<VirtualHost *:443>
   Header set Strict-Transport-Security "max-age=31536000"
</VirtualHost>
```

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:

```
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):

```
expose_php = Off
```

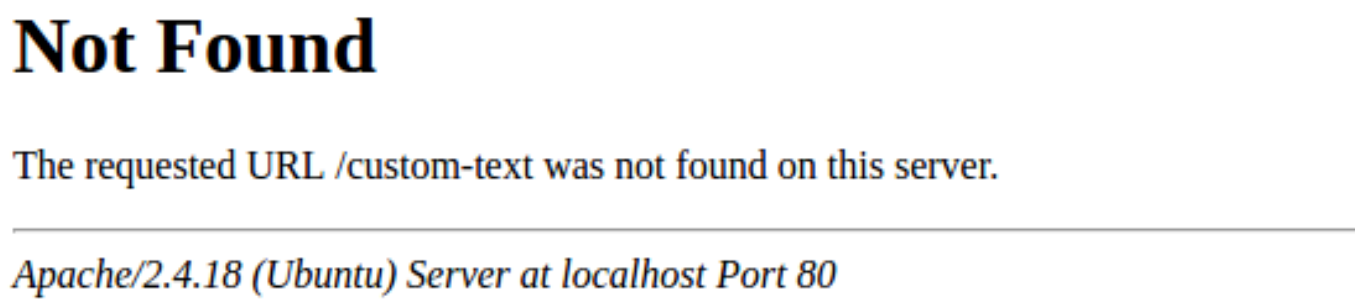
Web server restart is required for configuration file changes to be applied.

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:

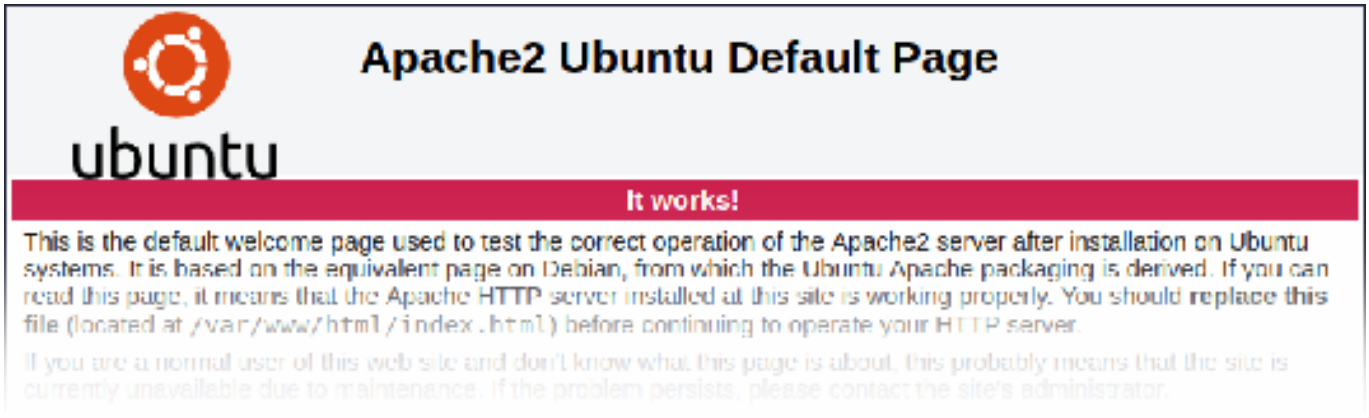


Default error pages should be replaced/removed as part of the web server hardening process. The "ErrorDocument" directive can be used to define a custom error page/text for Apache web server (used as an example).

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>
```

Displaying URL content in the sandbox

Since version 5.0.2, some Zabbix frontend elements (for example, the URL dashboard widget) are preconfigured to sandbox content retrieved from the URL. It is recommended to keep all sandboxing restrictions enabled to ensure protection against XSS attacks.

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.

Windows installer paths

When using Windows installers, it is recommended to use default paths provided by the installer as using custom paths without proper permissions could compromise the security of the installation.

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-5.0.0.tar.gz
```

> **Note:**
> 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
```

Attention:
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.

Attention:
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 --wit
```

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, Zabbix agent 2:

```
./configure --enable-agent2
```

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.
- A configured Go environment with a currently supported Go version is required for building Zabbix agent 2. See go.dev for installation instructions.

Attention:
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

Note:
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.

Note:
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**

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.

- edit the Zabbix server configuration file **/usr/local/etc/zabbix_server.conf**

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.

- 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).

Note:
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
```

Note:
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.

Run zabbix_agentd on all the monitored machines.

```
shell> zabbix_agentd
```

Note:
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.

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-rt_v6-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 the PCRE (not PCRE2) library (mandatory library since Zabbix 4.0) from http://pcre.org/ 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" -DCM
```

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 -O
```

4. Next, run:

```
mingw32-make install
```

Installing OpenSSL development libraries

1. Download 32 and 64 bit builds from 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.

Make sure to revoke write access from non-administrator users to the OpenSSL install directory (`C:\dev\openssl32` or `C:\dev\openssl`). Otherwise, Zabbix agent 2 will load SSL settings from a path that can be modified by unprivileged users, resulting in a potential security vulnerability.

Compiling Zabbix agent 2

32 bit

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

Run:

```
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:

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

Note:
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). 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/. To install it, open terminal and run the following command:

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

Then install Automake and pkg-config:

```
$ 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 PCRE:

```
$ brew install pcre
```

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

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

Download Zabbix source:

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

Build agent without TLS:

```
$ cd zabbix/
$ git checkout 5.0.1 -b 5.0.1 # replace 5.0.1 with the latest release available
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6
$ make
$ make install
```

Build agent with OpenSSL:

```
$ cd zabbix/
$ git checkout 5.0.1 -b 5.0.1 # replace 5.0.1 with the latest release available
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-openssl=/usr/local/op
$ make
$ make install
```

Build agent with GnuTLS:

```
$ cd zabbix/
$ git checkout 5.0.1 -b 5.0.1 # replace 5.0.1 with the latest release available
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-gnutls=/usr/local/opt
$ 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 PCRE 8.42.

```
$ PCRE_PREFIX="$HOME/static-libs/pcre-8.42"
```

Download and build PCRE with Unicode properties support:

```
$ mkdir static-libs-source
$ cd static-libs-source
$ curl --remote-name https://ftp.pcre.org/pub/pcre/pcre-8.42.tar.gz
$ tar xf pcre-8.42.tar.gz
$ cd pcre-8.42
$ ./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/
$ git checkout 5.0.1 -b 5.0.1 # replace 5.0.1 with the latest release available
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre="$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 PCRE 8.42 and OpenSSL 1.1.1a.

```
$ PCRE_PREFIX="$HOME/static-libs/pcre-8.42"
$ 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://ftp.pcre.org/pub/pcre/pcre-8.42.tar.gz
$ tar xf pcre-8.42.tar.gz
$ cd pcre-8.42
$ ./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 m
$ 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/
$ git checkout 5.0.1 -b 5.0.1 # replace 5.0.1 with the latest release available
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre="$PCRE_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 8.42, Nettle 3.4.1 and GnuTLS 3.6.5.

```
$ PCRE_PREFIX="$HOME/static-libs/pcre-8.42"
$ 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
$ 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 --d
$ make
$ make check
$ make install
$ cd ..
```

Download Zabbix source and build agent:

```
$ git clone https://git.zabbix.com/scm/zbx/zabbix.git
$ cd zabbix/
$ git checkout 5.0.1 -b 5.0.1 # replace 5.0.1 with the latest release available
$ ./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-libpcre="$PCRE_PREFIX
$ make
$ make install
```

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 commandline 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`
    - Make sure to revoke write access from non-administrator users to the OpenSSL install directory (`C:\OpenSSL-Win64-111-stati`. Otherwise, Zabbix agent will load SSL settings from a path that can be modified by unprivileged users, resulting in a potential security vulnerability.
    - 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 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 the PCRE (not PCRE2) library (mandatory library since Zabbix 4.0) from http://pcre.org/.
2. Extract to directory *E:\pcre-8.41*.
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:\pcre-8.41\build*.
5. Open a commandline 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:\pcre-8.41* and *E:\pcre-8.41\build* for the source and build directories, respectively.
7. Hit the "Configure" button.
8. When specifying the generator for this project select "NMake Makefiles".
9. Create a new, empty install directory. For example, *E:\pcre-8.41-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:\pcre-8.41-install*
11. Hit "Configure" again. The adjacent "Generate" button should now be active.
12. Hit "Generate".
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:\pcre-8.41\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/` `$ git checkout 5.0.1 -b 5.0.1 # replace 5.0.1 with the latest release available` `$ ./bootstrap.sh` `$ ./configure --enable-agent --enable-ipv6 --prefix=`pwd`` `$ make dbschema` `$ make dist`
2. Copy and unpack the archive, e.g. zabbix-5.0.0.tar.gz, on a Windows machine.
3. Let's assume that sources are in e:\zabbix-5.0.0. Open a commandline window e.g. the x64 Native Tools Command Prompt for VS 2017 RC. Go to E:\zabbix-5.0.0\build\win32\project.
4. Compile zabbix_get, zabbix_sender and zabbix_agent.
    - without TLS: `E:\zabbix-5.0.0\build\win32\project> nmake /K PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\pcre-8.41-install\lib`
    - with TLS: `E:\zabbix-5.0.0\build\win32\project> nmake /K -f Makefile_get TLS=openssl TLSINCDIR=C:\Ope TLSLIBDIR=C:\OpenSSL-Win64-111-static\lib PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\p E:\zabbix-5.0.0\build\win32\project> nmake /K -f Makefile_sender TLS=openssl TLSINCDIR="C:\OpenSS TLSLIBDIR="C:\OpenSSL-Win64-111-static\lib" PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\pcre-8.41-install\lib E:\zabbix-5.0.0\build\win32\project> nmake /K -f Makefile_agent TLS=openssl TLSINCDIR=C:\OpenSSL-Win64-111-static\include TLSLIBDIR=C:\OpenSSL-Win PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\pcre-8.41-install\lib`
5. New binaries are located in e:\zabbix-5.0.0\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) /DHAVE_OPENSSL /DHAVE_OPENSSL_WITH_PSK</code>and replace it with CFLAGS = $(CFLAGS) /DHAVE_OPENSSL

```
  * In ''Makefile_common.inc'' add ''/NODEFAULTLIB:LIBCMT'' i.e. find <code>
```

/MANIFESTUAC:"level='asInvoker' uiAccess='false'" /DYNAMICBASE:NO /PDB:$(TARGETDIR)\$(TARGETNAME).pdb</code>and replace it with `/MANIFESTUAC:"level='asInvoker' uiAccess='false'" /DYNAMICBASE:NO /PDB:$(TARGETDIR)\$(TARGETNAME) /NODEFAULTLIB:LIBCMT`

4 Installation from packages

Using 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 third 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.

1 Red Hat Enterprise Linux/CentOS

Overview

Official Zabbix 5.0 LTS packages for Red Hat Enterprise Linux, CentOS, and Oracle Linux are available on Zabbix website.

Attention:
Zabbix packages for Red Hat Enterprise Linux systems are intended only for RHEL systems. Alternative environments, such as Red Hat Universal Base Image, may lack the necessary dependencies and repository access requirements for successful installation. To address such issues, verify compatibility with the target environment and ensure access to required repositories and dependencies before proceeding with Zabbix installation from packages. For more information, see Known issues.

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

Zabbix agent packages and utilities *Zabbix get* and *Zabbix sender* are available on Zabbix Official Repository for RHEL 9, RHEL 8, RHEL 7, RHEL 6, and RHEL 5.

Zabbix Official Repository provides *fping*, *iksemel* and *libssh2* packages as well. These packages are located in the non-supported directory.

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

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.

Importing data with Timescale DB

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

```
# zcat /usr/share/doc/zabbix-server-pgsql*/timescaledb.sql.gz | sudo -u zabbix psql zabbix
```

Warning:
TimescaleDB is supported with Zabbix server only.

Frontend installation prerequisites

Zabbix frontend requires additional packages not available in basic installation. You need to enable repository of optional rpms in the system you will run Zabbix frontend on:

RHEL 7:

```
# yum-config-manager --enable rhel-7-server-optional-rpms
```

Note:
Note that Nginx for RHEL is available in Red Hat Software Collections and in EPEL. If Red Hat Software Collections are used, simply install zabbix-nginx-conf-scl package.

PHP 7.2

Zabbix frontend requires PHP version **7.2 or newer** starting with Zabbix 5.0.

Note that RHEL/CentOS 7 only provide PHP 5.4. See instructions for installing Zabbix frontend on Red Hat Enterprise Linux/CentOS 7.

SELinux configuration

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:

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

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

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

RHEL prior to 7:

```
# 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:

```
# systemctl restart httpd
```

Proxy installation

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

```
# yum install zabbix-proxy-mysql
```

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

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:

```
# zcat /usr/share/doc/zabbix-proxy-mysql*/schema.sql.gz | mysql -uzabbix -p zabbix
```

For proxy with PostgreSQL (or SQLite):

```
# zcat /usr/share/doc/zabbix-proxy-pgsql*/schema.sql.gz | sudo -u zabbix psql zabbix
# zcat /usr/share/doc/zabbix-proxy-sqlite3*/schema.sql.gz | 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. See SELinux configuration for instructions.

Starting Zabbix proxy process

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

```
# systemctl start 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:

```
# yum install zabbix-java-gateway
```

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

Installing debuginfo packages

> **Note:**
> Debuginfo packages are currently available for RHEL/CentOS versions 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 - $basearch
baseurl=http://repo.zabbix.com/zabbix/5.0/rhel/7/$basearch/debuginfo/
enabled=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 5.0 LTS packages for Debian, Ubuntu, and Raspberry Pi OS (Raspbian) are available on Zabbix website.

Packages are available with either MySQL/PostgreSQL database and Apache/Nginx web server 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.

Debian-based distributions usually provide their own Zabbix packages in their repositories. These are not supported by Zabbix. Only the ones from Zabbix Official Repository are.

Importing data with Timescale DB

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

```
# zcat /usr/share/doc/zabbix-server-pgsql*/timescaledb.sql.gz | sudo -u zabbix psql zabbix
```

> **Warning:**
> TimescaleDB is supported with Zabbix server only.

PHP 7.2

Zabbix frontend requires PHP version **7.2 or newer** starting with Zabbix 5.0.

See instructions for installing Zabbix frontend on distributions with PHP versions below 7.2.

SELinux configuration

See SELinux configuration for RHEL/CentOS.

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

```
# systemctl restart apache2
```

Proxy installation

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

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

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

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:

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

For proxy with PostgreSQL (or SQLite):

```
# zcat /usr/share/doc/zabbix-proxy-pgsql/schema.sql.gz | sudo -u zabbix psql zabbix
# zcat /usr/share/doc/zabbix-proxy-sqlite3/schema.sql.gz | 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 5.0 LTS packages for SUSE Linux Enterprise Server are available on Zabbix website.

Zabbix agent packages and utilities *Zabbix get* and *Zabbix sender* are available on Zabbix Official Repository for SLES 15 (SP4 and newer) and SLES 12 (SP4 and newer).

Note:
Using SLES 15 with SP3 or older is not recommended and may contain limitations. Also, please note that the *Verify CA* encryption mode does not 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/5.0/sles/15/x86_64/zabbix-release-latest.el15.noar
# zypper --gpg-auto-import-keys refresh 'Zabbix Official Repository'
```

SLES 12:

```
# rpm -Uvh --nosignature https://repo.zabbix.com/zabbix/5.0/sles/12/x86_64/zabbix-release-latest.el12.noar
# zypper --gpg-auto-import-keys refresh 'Zabbix Official Repository'
```

Server/frontend/agent installation

To be able to install Zabbix frontend `Web and Scripting Module` must be activated. It contains the necessary PHP dependencies.

SLES 15:

```
# SUSEConnect -p sle-module-web-scripting/15/x86_64
```

SLES 12:

```
# SUSEConnect -p sle-module-web-scripting/12/x86_64
```

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).

To install Zabbix proxy with MySQL support:

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

Substitute 'mysql' in the commands with 'pgsql' to use PostgreSQL.

Creating database

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

Warning:
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-server-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-server-pgsql*/create.sql.gz | sudo -u <username> psql zabbix
```

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

```
# zcat /usr/share/doc/packages/zabbix-server-pgsql*/timescaledb.sql.gz | sudo -u <username> psql zabbix
```

Warning:
TimescaleDB is supported with Zabbix server only.

For **proxy**, import initial schema:

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

For proxy with PostgreSQL:

```
# zcat /usr/share/doc/packages/zabbix-proxy-pgsql*/schema.sql.gz | sudo -u <username> 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/5.0/sles/15/x86_64/debuginfo/
gpgcheck=1
gpgkey=http://repo.zabbix.com/zabbix/5.0/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.

The minimum requirement for Zabbix agent 2 MSI installation is Windows 10 32-bit/Server 2016.

The Zabbix get and sender utilities can also be installed, either together with Zabbix agent/agent 2 or separately.

A 32-bit package cannot be installed on a 64-bit Windows.

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

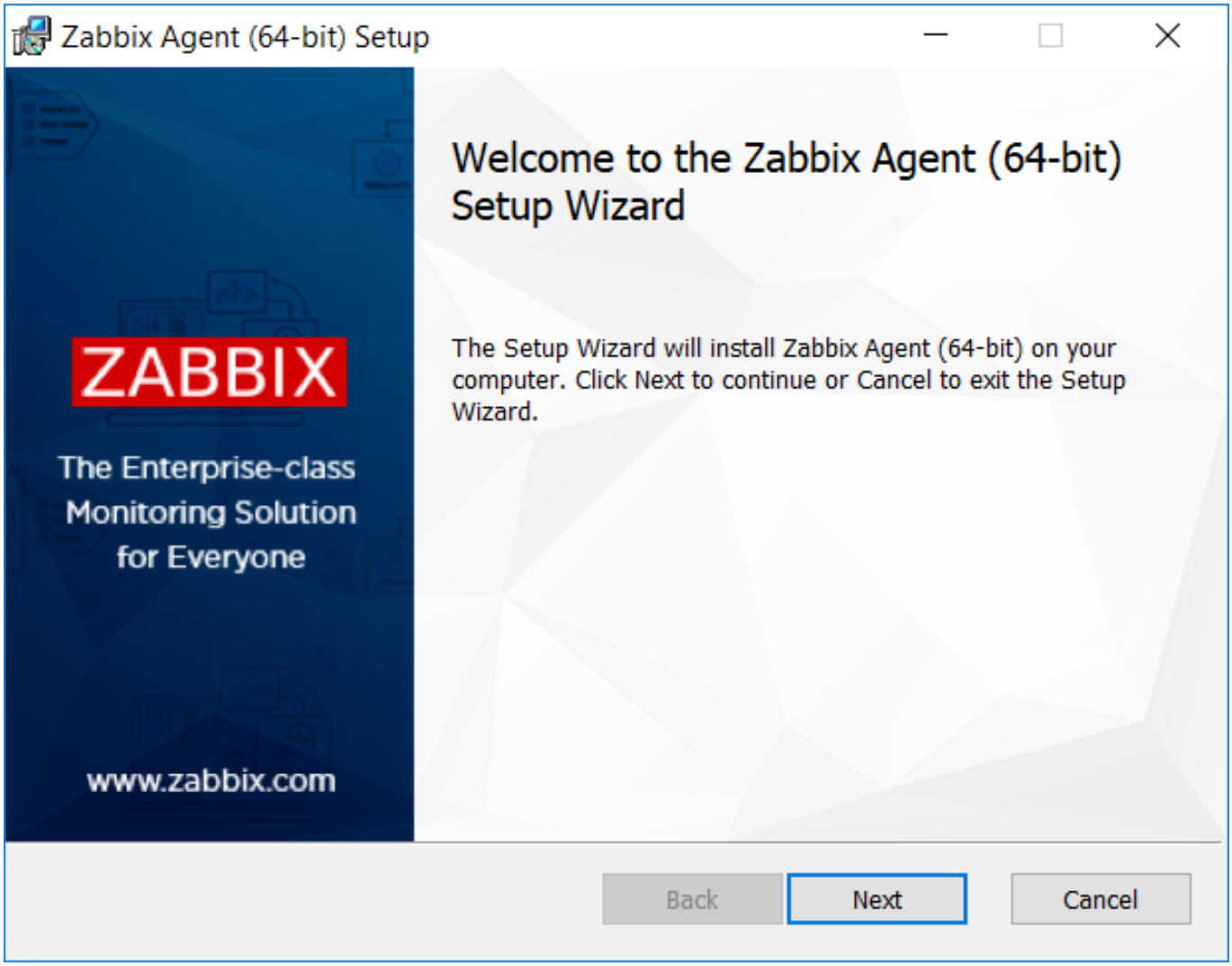
Both UI and command-line based installation is supported.

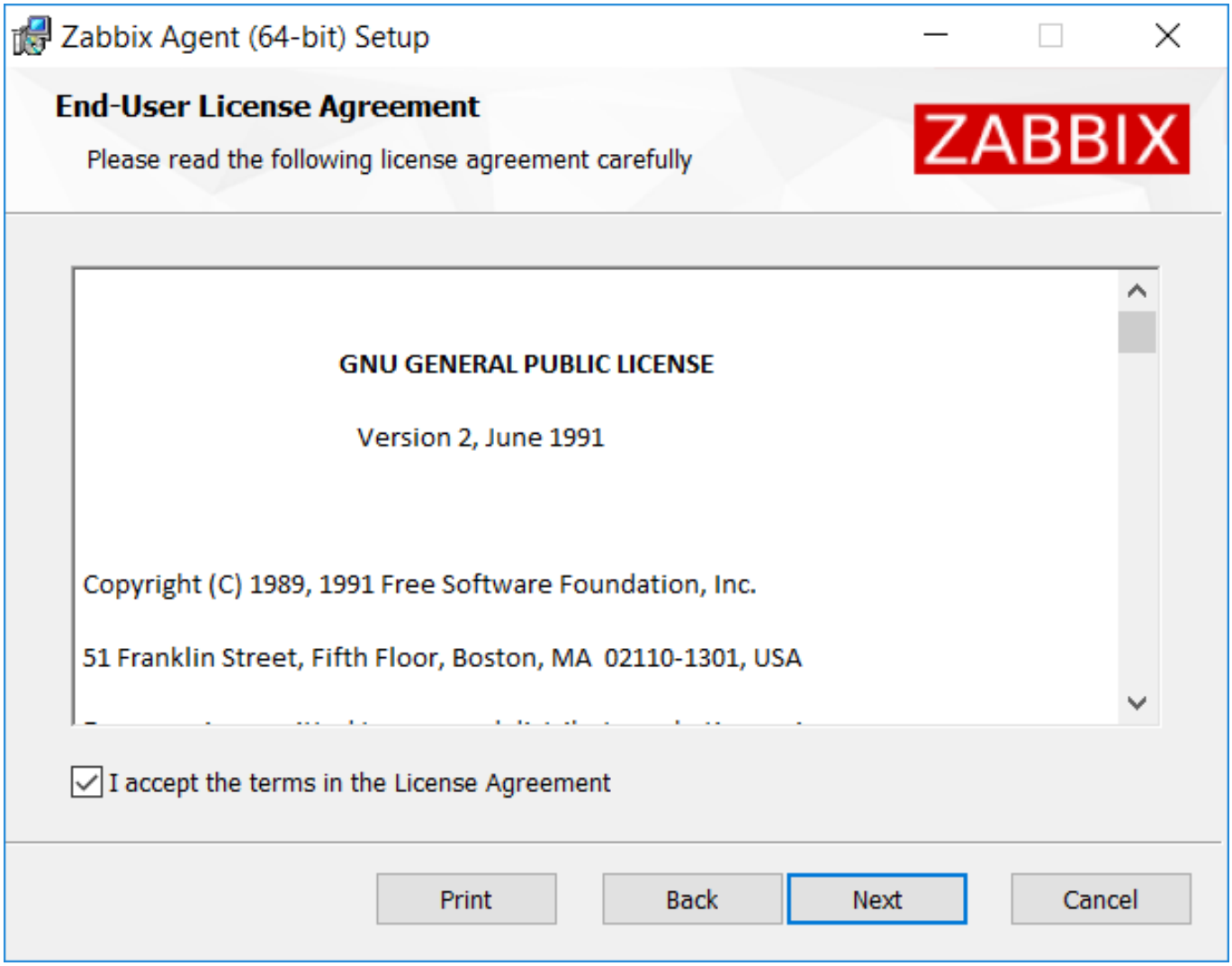
Note:
Although Zabbix installation from MSI installer packages is fully supported, it is recommended to install at least *Microsoft .NET Framework 2* for proper error handling. See Microsoft Download .NET Framework.

Attention:
It is recommended to use default paths provided by the installer as using custom paths without proper permissions could compromise the security of the installation.

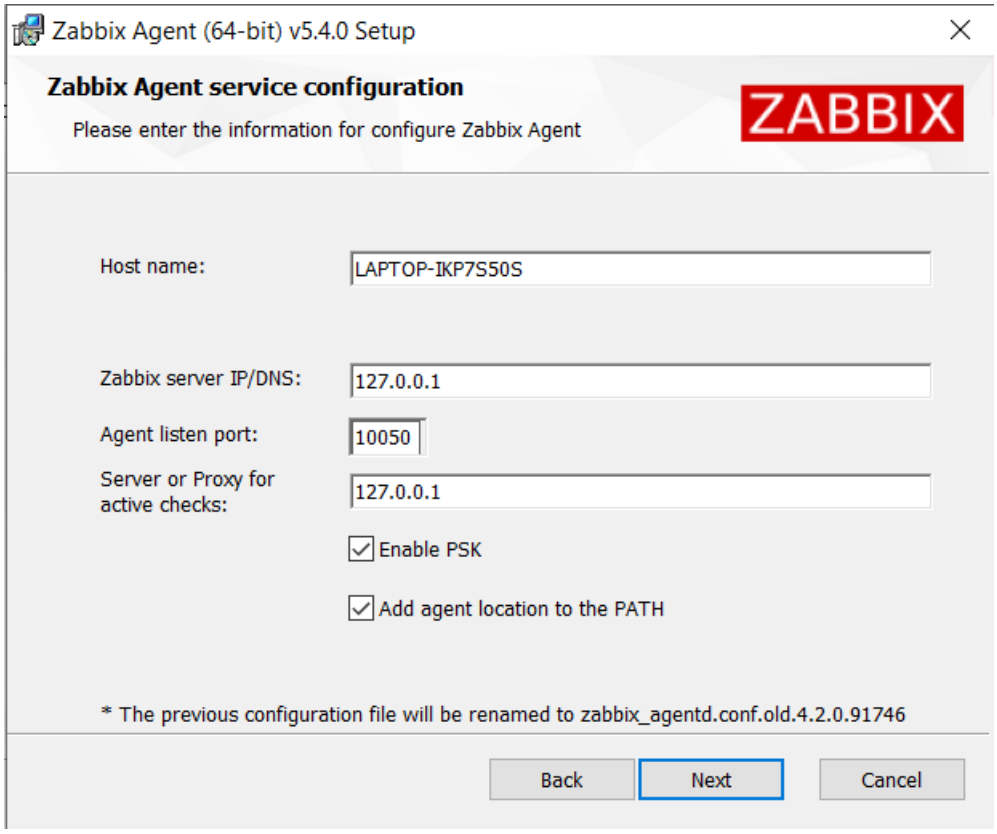
Installation steps

To install, double-click the downloaded MSI file.


Zabbix Agent (64-bit) Setup
ZABBIX
The Enterprise-class Monitoring Solution for Everyone
www.zabbix.com
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.
Back
Next
Cancel

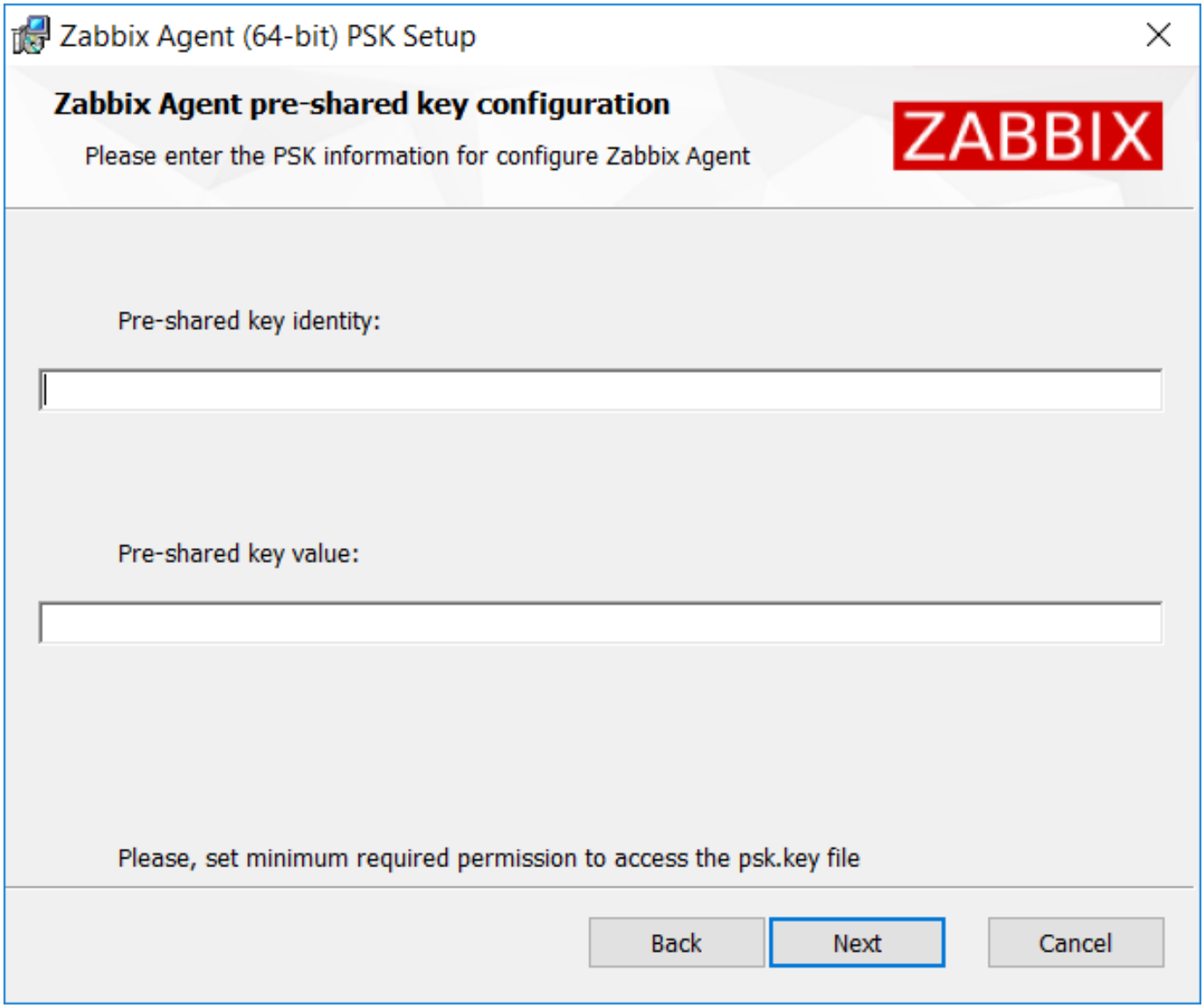


Accept the license to proceed to the next step.

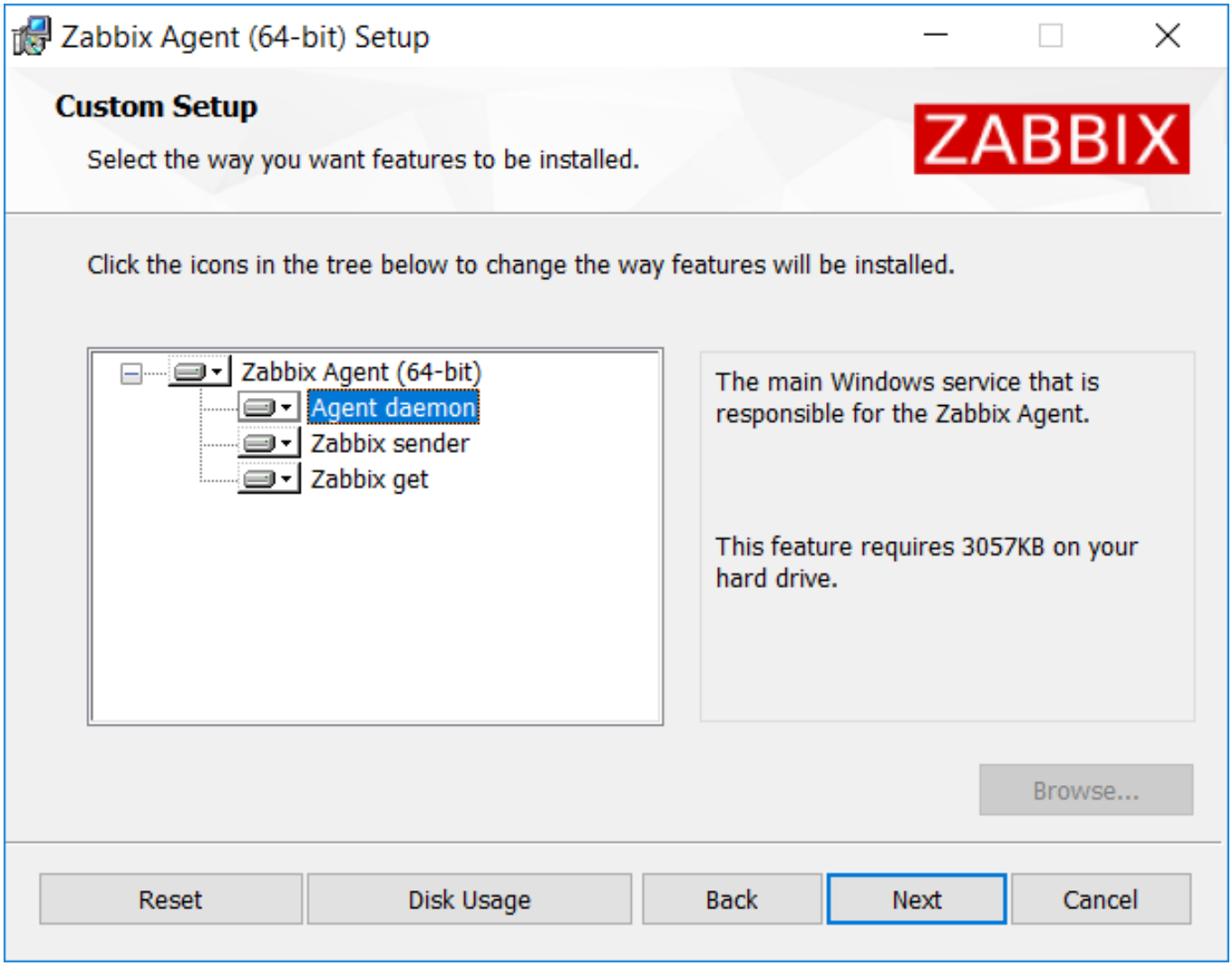


Specify the following parameters.

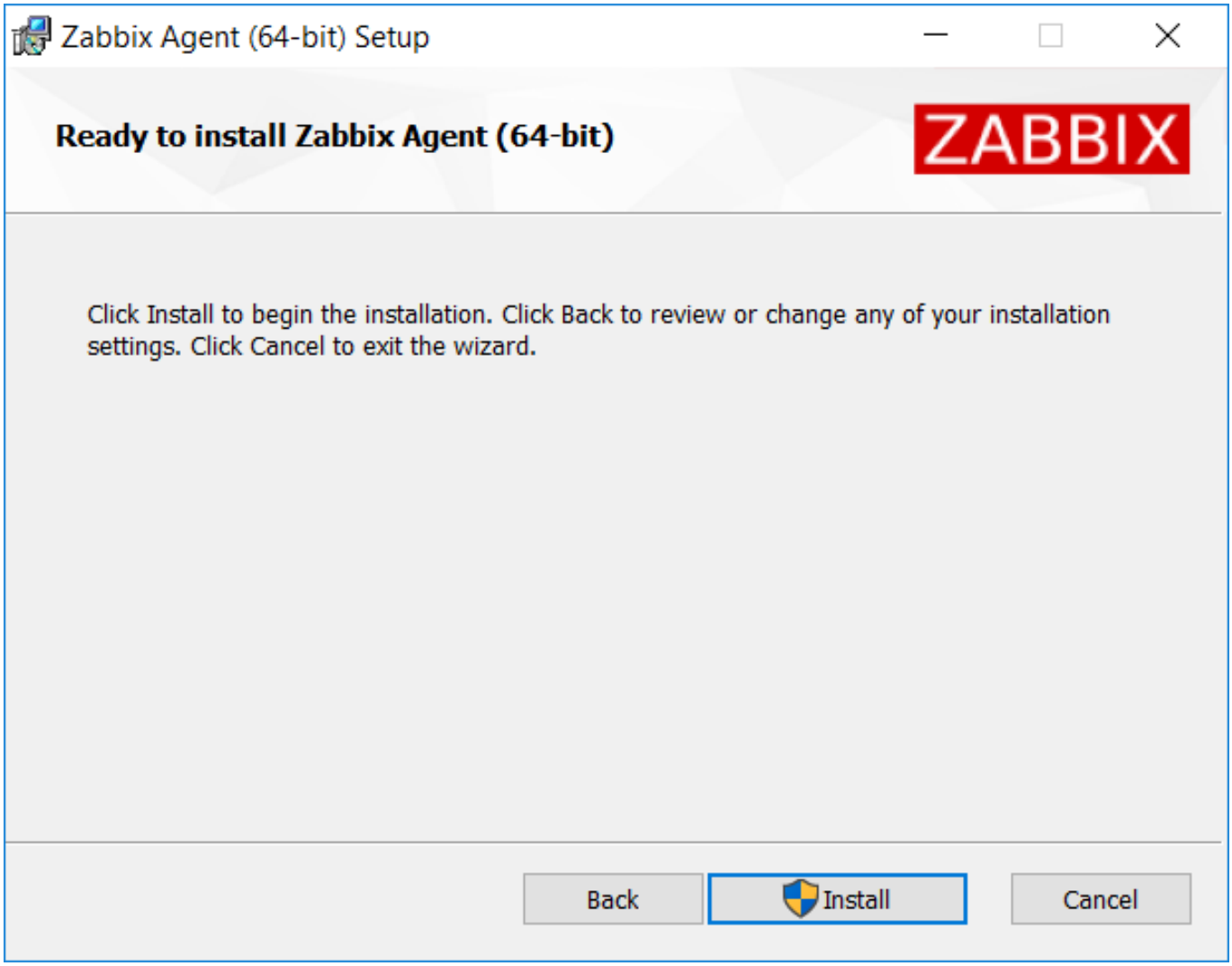
Parameter	Description
Host name	Specify host name.
Zabbix server IP/DNS	Specify IP/DNS of Zabbix server.
Agent listen port	Specify agent listen port (10050 by default).
Server or Proxy for active checks	Specify IP/DNS of Zabbix server/proxy for active agent checks.
Enable PSK	Mark the checkbox to enable TLS support via pre-shared keys.
Add agent location to the PATH	Add agent location to the PATH variable.



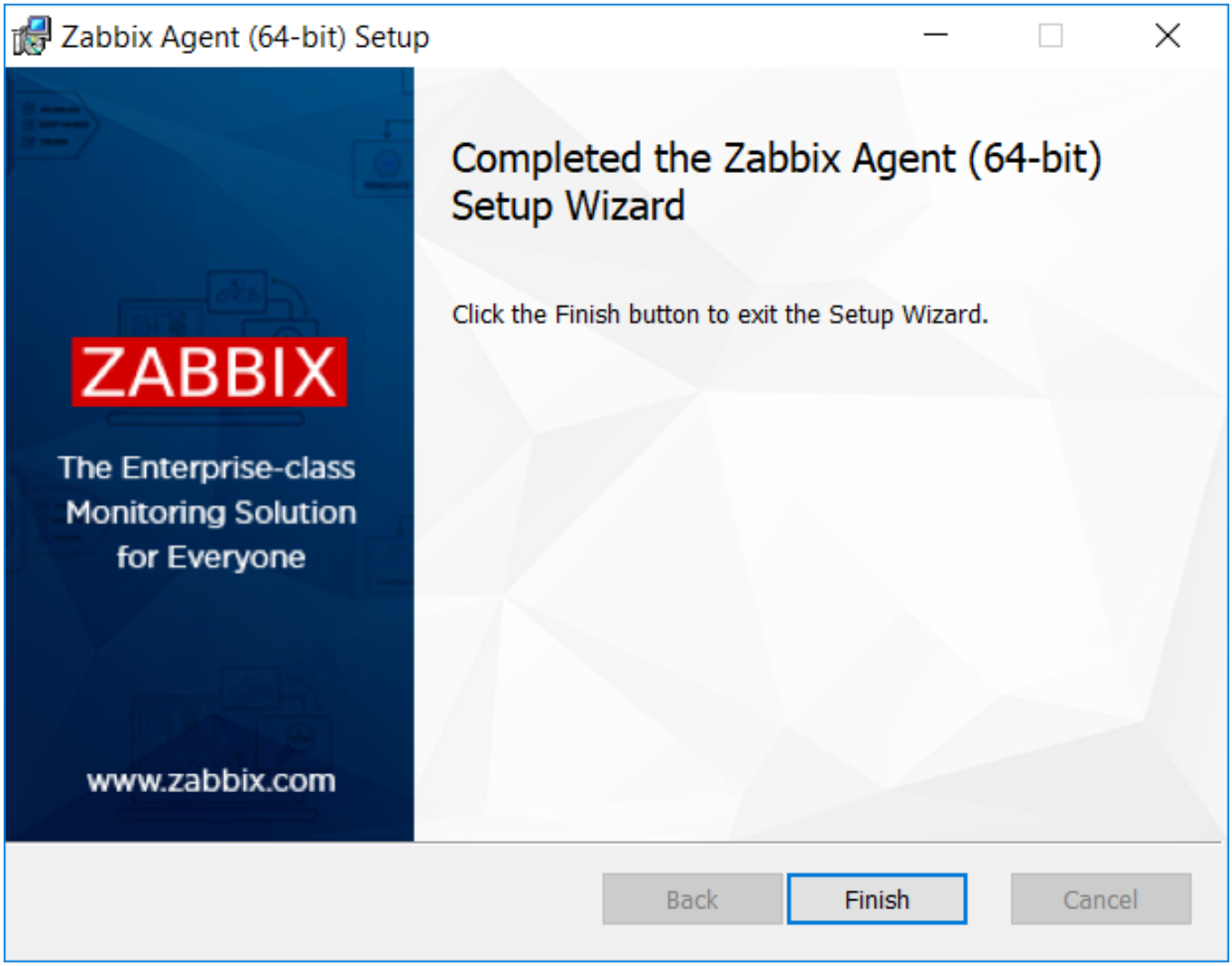
Enter pre-shared key identity and value. This step is only available if you checked *Enable PSK* in the previous step.



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:

Number	Parameter	Description
1	LOGTYPE	
2	LOGFILE	
3	SERVER	
4	LISTENPORT	
5	SERVERACTIVE	
6	HOSTNAME	
7	TIMEOUT	
8	TLSCONNECT	
9	TLSACCEPT	
10	TLSPSKIDENTITY	
11	TLSPSKFILE	
12	TLSPSKVALUE	
13	TLSCAFILE	
14	TLSCRLFILE	
15	TLSSERVERCERTISSUER	
16	TLSSERVERCERTSUBJECT	
17	TLSCERTFILE	
18	TLSKEYFILE	
19	LISTENIP	
20	HOSTINTERFACE	
21	HOSTMETADATA	
22	HOSTMETADATAITEM	Supported since Zabbix 5.0.19.

Number	Parameter	Description
23	STATUSPORT	Zabbix agent 2 only.
24	ENABLEPERSISTENTBUFFER	Zabbix agent 2 only.
25	PERSISTENTBUFFERPERIOD	Zabbix agent 2 only.
26	PERSISTENTBUFFERFILE	Zabbix agent 2 only.
27	INSTALLFOLDER	
28	ENABLEPATH	
29	SKIP	`SKIP=fw` - do not install firewall exception rule
30	INCLUDE	Sequence of includes separated by `;`. Supported since Zabbix 5.0.19.
31	ALLOWDENYKEY	Sequence of "AllowKey" and "DenyKey" parameters separated by `;`. Use `\\;` to escape the delimiter. Example: `ALLOWDENYKEY="AllowKey=system.run[type c:\windows\system32\drivers\etc\hosts];DenyKey=s` Supported since Zabbix 5.0.19.
32	ADDPROGRAM	A comma-delimited list of programs to install. Possible values: `AgentProgram`, `GetProgram`, `SenderProgram` E.g., ADDPROGRAM=AgentProgram,GetProgram
33	ADDLOCAL	A comma-delimited list of programs to install. Possible values: `AgentProgram`, `GetProgram`, `SenderProgram` E.g., ADDLOCAL=AgentProgram,SenderProgram
34	CONF	Specify path to custom configuration file, e.g., `CONF=c:\full\path\to\user.conf`

To install you may run, for example:

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

msiexec /l*v log.txt /i zabbix_agent-5.0.20-x86.msi /qn^
 LOGTYPE=file^
 LOGFILE="%INSTALLFOLDER%\za.log"^
 SERVER=192.168.6.76^
 LISTENPORT=12345^
 SERVERACTIVE=::1^
 HOSTNAME=myHost^
 TLSCONNECT=psk^
 TLSACCEPT=psk^
 TLSPSKIDENTITY=MyPSKID^
 TLSPSKFILE="%INSTALLFOLDER%\mykey.psk"^
 TLSCAFILE="c:\temp\f.txt1"^
 TLSCRLFILE="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

```
msiexec /l*v log.txt /i zabbix_agent-5.0.20-x86.msi /qn^
 SERVER=192.168.6.76^
 TLSCONNECT=psk^
 TLSACCEPT=psk^
 TLSPSKIDENTITY=MyPSKID^
 TLSPSKVALUE=1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952
```

If both TLSPSKFILE and TLSPSKVALUE are passed, then TLSPSKVALUE will be written to TLSPSKFILE.

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:

```
sudo installer -pkg zabbix_agent-5.0.30-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:

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

To stop the agent manually:

```
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:

```
ps aux | grep zabbix_agentd
```

See if Zabbix agent has been installed from packages:

```
$ 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):

```
$ 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`:

```
sudo launchctl unload /Library/LaunchDaemons/com.zabbix.zabbix_agentd.plist
```

Remove files (including configuration and logs) that were installed with installer package:

```
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:

```
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:

Image	Version
alpine	3.14
ubuntu	20.04 (focal)
centos	8

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-5.0-latest` - latest minor version of a Zabbix 5.0 component based on Alpine Linux image
- `ubuntu-5.0-latest` - latest minor version of a Zabbix 5.0 component based on Ubuntu image
- `alpine-5.0.*` - different minor versions of a Zabbix 5.0 component based on Alpine Linux image, where * is the minor version of Zabbix component
- `ubuntu-5.0.*` - different minor versions of a Zabbix 5.0 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.

Attention:
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:

Variable	Components	Description
DB_SERVER_HOST	Server Proxy Web interface	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
DB_SERVER_PORT	Server Proxy Web interface	This variable is port of MySQL or PostgreSQL server. By default, value is '3306' or '5432' respectively.
MYSQL_USER	Server Proxy Web-interface	MySQL database user. By default, value is 'zabbix'.
MYSQL_PASSWORD	Server Proxy Web interface	MySQL database password. By default, value is 'zabbix'.
MYSQL_DATABASE	Server Proxy Web interface	Zabbix database name. By default, value is 'zabbix' for Zabbix server and 'zabbix_proxy' for Zabbix proxy.
POSTGRES_USER	Server Web interface	PostgreSQL database user. By default, value is 'zabbix'.
POSTGRES_PASSWORD	Server Web interface	PostgreSQL database password. By default, value is 'zabbix'.
POSTGRES_DB	Server Web interface	Zabbix database name. By default, value is 'zabbix' for Zabbix server and 'zabbix_proxy' for Zabbix proxy.
PHP_TZ	Web-interface	Timezone in PHP format. Full list of supported timezones are available on php.net. By default, value is 'Europe/Riga'.
ZBX_SERVER_NAME	Web interface	Visible Zabbix installation name in right top corner of the web interface. By default, value is 'Zabbix Docker'
ZBX_JAVAGATEWAY_ENABLE	Server Proxy	Enables communication with Zabbix Java gateway to collect Java related checks. By default, value is "false"
ZBX_ENABLE_SNMP_TRAPS	Server Proxy	Enables SNMP trap feature. It requires **zabbix-snmptraps** instance and shared volume */var/lib/zabbix/snmptraps* to Zabbix server or Zabbix proxy.

Volumes

The images allow to use some mount points. These mount points are different and depend on Zabbix component type:

Volume	Description
Zabbix agent	
/etc/zabbix/zabbix_agentd.d	The volume allows to include **.conf* files and extend Zabbix agent using the UserParameter feature

/var/lib/zabbix/modules	The volume allows to load additional modules and extend Zabbix agent using the LoadModule feature
/var/lib/zabbix/enc	The volume is used to store TLS-related files. These file names are specified using ZBX_TLSCAFILE, ZBX_TLSCRLFILE, ZBX_TLSKEY_FILE and ZBX_TLSPSKFILE environment variables
Zabbix server	
/usr/lib/zabbix/alertscripts	The volume is used for custom alert scripts. It is the AlertScriptsPath parameter in zabbix_server.conf
/usr/lib/zabbix/externalscripts	The volume is used by external checks. It is the ExternalScripts parameter in zabbix_server.conf
/var/lib/zabbix/modules	The volume allows to load additional modules and extend Zabbix server using the LoadModule feature
/var/lib/zabbix/enc	The volume is used to store TLS related files. These file names are specified using ZBX_TLSCAFILE, ZBX_TLSCRLFILE, ZBX_TLSKEY_FILE and ZBX_TLSPSKFILE environment variables
/var/lib/zabbix/ssl/certs	The volume is used as location of SSL client certificate files for client authentication. It is the SSLCertLocation parameter in zabbix_server.conf
/var/lib/zabbix/ssl/keys	The volume is used as location of SSL private key files for client authentication. It is the SSLKeyLocation parameter in zabbix_server.conf
/var/lib/zabbix/ssl/ssl_ca	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
/var/lib/zabbix/snmptraps	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'
/var/lib/zabbix/mibs	The volume allows to add new MIB files. It does not support subdirectories, all MIBs must be placed in /var/lib/zabbix/mibs
Zabbix proxy	
/usr/lib/zabbix/externalscripts	The volume is used by external checks. It is the ExternalScripts parameter in zabbix_proxy.conf
/var/lib/zabbix/db_data/	The volume allows to store database files on external devices. Supported only for Zabbix proxy with SQLite3
/var/lib/zabbix/modules	The volume allows to load additional modules and extend Zabbix server using the LoadModule feature
/var/lib/zabbix/enc	The volume is used to store TLS related files. These file names are specified using ZBX_TLSCAFILE, ZBX_TLSCRLFILE, ZBX_TLSKEY_FILE and ZBX_TLSPSKFILE environment variables
/var/lib/zabbix/ssl/certs	The volume is used as location of SSL client certificate files for client authentication. It is the SSLCertLocation parameter in zabbix_proxy.conf
/var/lib/zabbix/ssl/keys	The volume is used as location of SSL private key files for client authentication. It is the SSLKeyLocation parameter in zabbix_proxy.conf
/var/lib/zabbix/ssl/ssl_ca	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
/var/lib/zabbix/snmptraps	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'

/var/lib/zabbix/mibs	The volume allows to add new MIB files. It does not support subdirectories, all MIBs must be placed in `/var/lib/zabbix/mibs`
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	
/var/lib/zabbix/snmptraps	The volume contains the `snmptraps.log` log file named with received SNMP traps
/var/lib/zabbix/mibs	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:

```
# docker network create --subnet 172.20.0.0/16 --ip-range 172.20.240.0/20 zabbix-net
```

2. Start empty MySQL server instance

```
# docker run --name mysql-server -t \
      -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 \
      --character-set-server=utf8 --collation-server=utf8_bin \
      --default-authentication-plugin=mysql_native_password
```

3. Start Zabbix Java gateway instance

```
# docker run --name zabbix-java-gateway -t \
      --network=zabbix-net \
      --restart unless-stopped \
      -d zabbix/zabbix-java-gateway:alpine-5.0-latest
```

4. Start Zabbix server instance and link the instance with created MySQL server instance

```
# 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-5.0-latest
```

Note:
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

```
# docker run --name zabbix-web-nginx-mysql -t \
      -e ZBX_SERVER_HOST="zabbix-server-mysql" \
      -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-5.0-latest
```

Note:
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:

```
# docker network create --subnet 172.20.0.0/16 --ip-range 172.20.240.0/20 zabbix-net
```

2. Start empty PostgreSQL server instance

```
# 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

```
# 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-5.0-latest
```

Note:
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

```
# 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-5.0-latest
```

Note:
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

```
# docker run --name zabbix-web-nginx-pgsql -t \
      -e ZBX_SERVER_HOST="zabbix-server-pgsql" \
      -e DB_SERVER_HOST="postgres-server" \
      -e POSTGRES_USER="zabbix" \
      -e POSTGRES_PASSWORD="zabbix_pwd" \
      -e POSTGRES_DB="zabbix" \
      --network=zabbix-net \
      -p 443:8443 \
      -p 80:8080 \
      -v /etc/ssl/nginx:/etc/ssl/nginx:ro \
      --restart unless-stopped \
      -d zabbix/zabbix-web-nginx-pgsql:alpine-5.0-latest
```

Note:
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):

```
podman pod create --name zabbix -p 80:8080 -p 10051:10051
```

2. (optional) Start Zabbix agent container in `zabbix` pod location:

```
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:

```
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 ./mysql/:/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
```

4. Start Zabbix server container:

```
podman run --name zabbix-server-mysql -t \
                  -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" \
                  -e ZBX_JAVAGATEWAY="127.0.0.1" \
                  --restart=always \
                  --pod=zabbix \
                  -d registry.connect.redhat.com/zabbix/zabbix-server-mysql-50
```

5. 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
```

6. 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
```

Note:
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:

File name	Description
`docker-compose_v3_alpine_mysql_latest.yaml`	The compose file runs the latest version of Zabbix 5.0 components on Alpine Linux with MySQL database support.
`docker-compose_v3_alpine_mysql_local.yaml`	The compose file locally builds the latest version of Zabbix 5.0 and runs Zabbix components on Alpine Linux with MySQL database support.
`docker-compose_v3_alpine_pgsql_latest.yaml`	The compose file runs the latest version of Zabbix 5.0 components on Alpine Linux with PostgreSQL database support.
`docker-compose_v3_alpine_pgsql_local.yaml`	The compose file locally builds the latest version of Zabbix 5.0 and runs Zabbix components on Alpine Linux with PostgreSQL database support.
`docker-compose_v3_centos_mysql_latest.yaml`	The compose file runs the latest version of Zabbix 5.0 components on CentOS 8 with MySQL database support.
`docker-compose_v3_centos_mysql_local.yaml`	The compose file locally builds the latest version of Zabbix 5.0 and runs Zabbix components on CentOS 8 with MySQL database support.
`docker-compose_v3_centos_pgsql_latest.yaml`	The compose file runs the latest version of Zabbix 5.0 components on CentOS 8 with PostgreSQL database support.
`docker-compose_v3_centos_pgsql_local.yaml`	The compose file locally builds the latest version of Zabbix 5.0 and runs Zabbix components on CentOS 8 with PostgreSQL database support.
`docker-compose_v3_ubuntu_mysql_latest.yaml`	The compose file runs the latest version of Zabbix 5.0 components on Ubuntu 20.04 with MySQL database support.
`docker-compose_v3_ubuntu_mysql_local.yaml`	The compose file locally builds the latest version of Zabbix 5.0 and runs Zabbix components on Ubuntu 20.04 with MySQL database support.
`docker-compose_v3_ubuntu_pgsql_latest.yaml`	The compose file runs the latest version of Zabbix 5.0 components on Ubuntu 20.04 with PostgreSQL database support.
`docker-compose_v3_ubuntu_pgsql_local.yaml`	The compose file locally builds the latest version of Zabbix 5.0 and runs Zabbix components on Ubuntu 20.04 with PostgreSQL database support.

Attention:
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 **

```
# git checkout 5.0
# docker-compose -f ./docker-compose_v3_alpine_mysql_latest.yaml up -d
```

The command will download latest Zabbix 5.0 images for each Zabbix component and run them in detach mode.

Attention:
Do not forget to download `.env_<type of component>` files from github.com official Zabbix repository with compose files.

** Example 2 **

```
# git checkout 5.0
# 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 5.0 components locally and run them in detach mode.

Installation with OpenShift

Overview

Zabbix helps you to do a real-time monitoring of millions of metrics collected from tens of thousands of servers, virtual machines and network devices. The Zabbix Operator allows users to easily deploy, manage, and maintain Zabbix deployments on OpenShift. By installing this integration you will be able to deploy Zabbix server/proxies and other components with a single command.

Supported features

Zabbix Operator comes with a few possible installation options:

- **Zabbix server** - a simple Zabbix installation with included Zabbix server, Zabbix web interface and Zabbix Java gateway with MySQL database support. The feature does not provide MySQL service and requires an external MySQL database.
- **Zabbix server (full)** - a Zabbix installation with included Zabbix server, Zabbix web interface, Zabbix Java gateway and MySQL server instance.
- **Zabbix proxy (SQLite3)** - a very simple way to gain power of Zabbix proxy. The feature has SQLite3 support for Zabbix proxies and allows to specify the amount of proxies.
- **Zabbix proxy (MySQL)** - another option of Zabbix proxy. This option supports and delivers a MySQL database. It is possible to use a built-in MySQL database instance or an external one.
- **Zabbix agent** - a Zabbix agent can be deployed on each available node for stability and performance monitoring on remote nodes. It allows to gather metrics with full automation!
- **Zabbix appliance** - a Zabbix appliance is a very simple way to test and check Zabbix features. This option provides all the core components in one solution. It includes Zabbix server, Zabbix Java gateway, Zabbix web interface and MySQL server in deployment. It is very useful for testing Zabbix features!

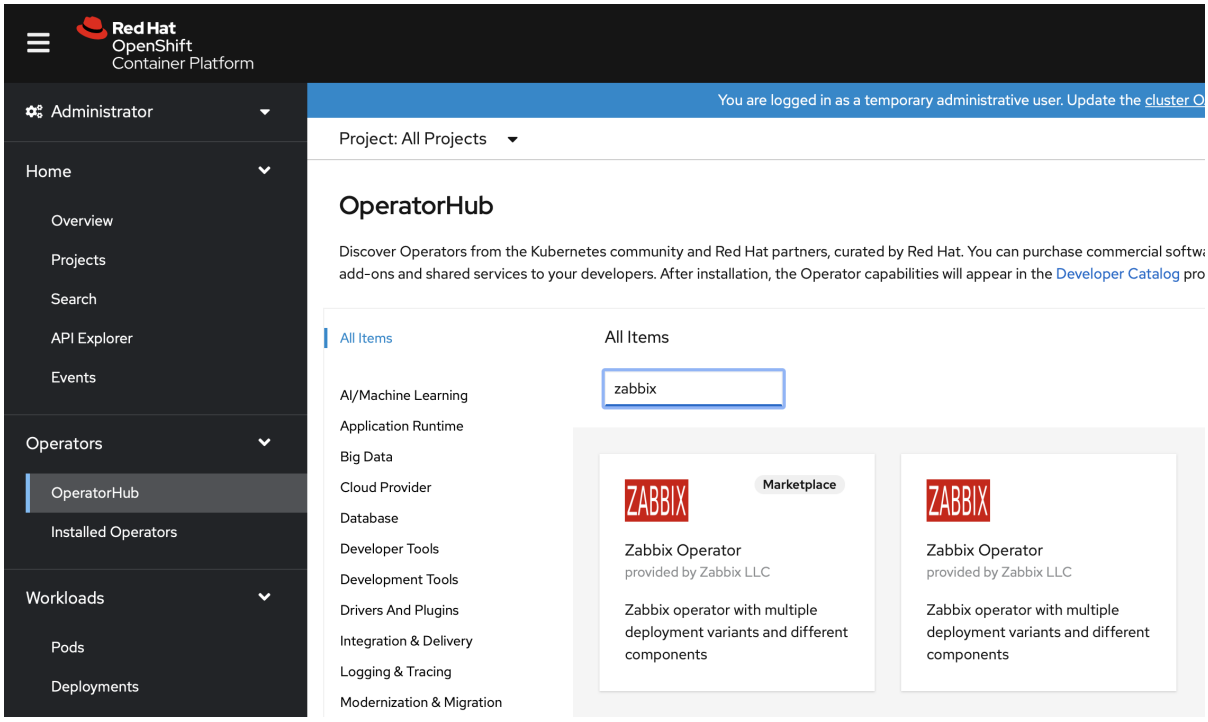
Currently Zabbix Operator is based on the Zabbix 5.0 LTS version and supports OpenShift 4.0, 4.1, 4.2, 4.3, 4.4 and 4.5.

Installing Zabbix Operator

Using RedHat Marketplace

Attention:
The installation of Zabbix Operator using Red Hat Marketplace requires the OpenShift cluster to be registered in the Marketplace Portal, including the roll out of the PullSecret in your cluster. Failure to do so will result in an image pull authentication failure with the Red Hat registry.

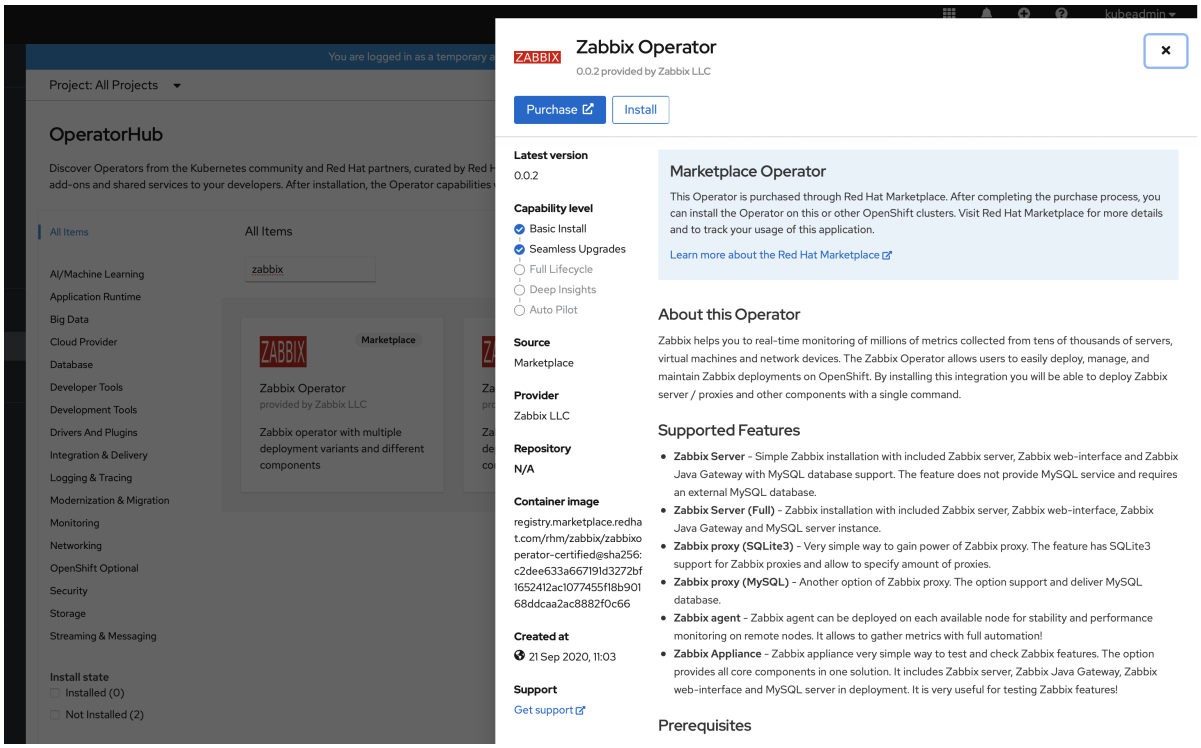
1. Select the OperatorHub from the Operators submenu and search for Zabbix.



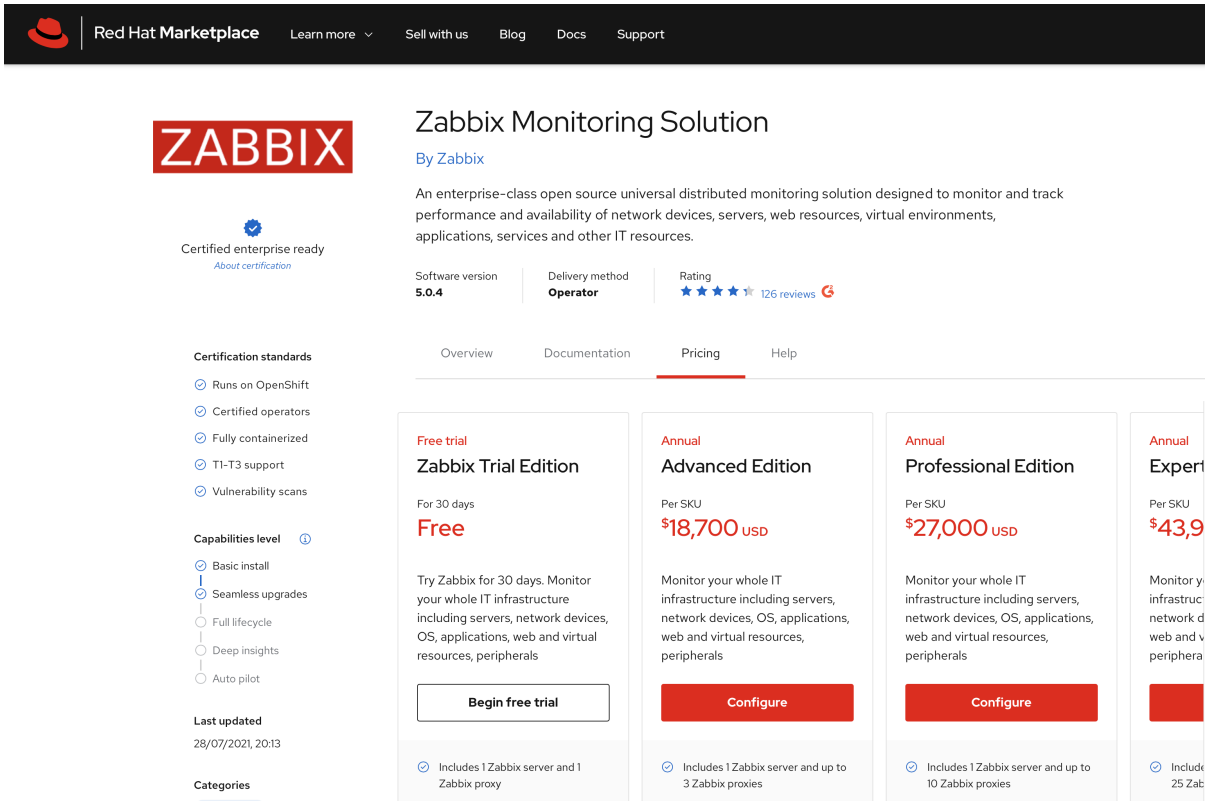
Choose the RedHat Marketplace option.

2. Select ”Zabbix Operator” and click on *Purchase*.

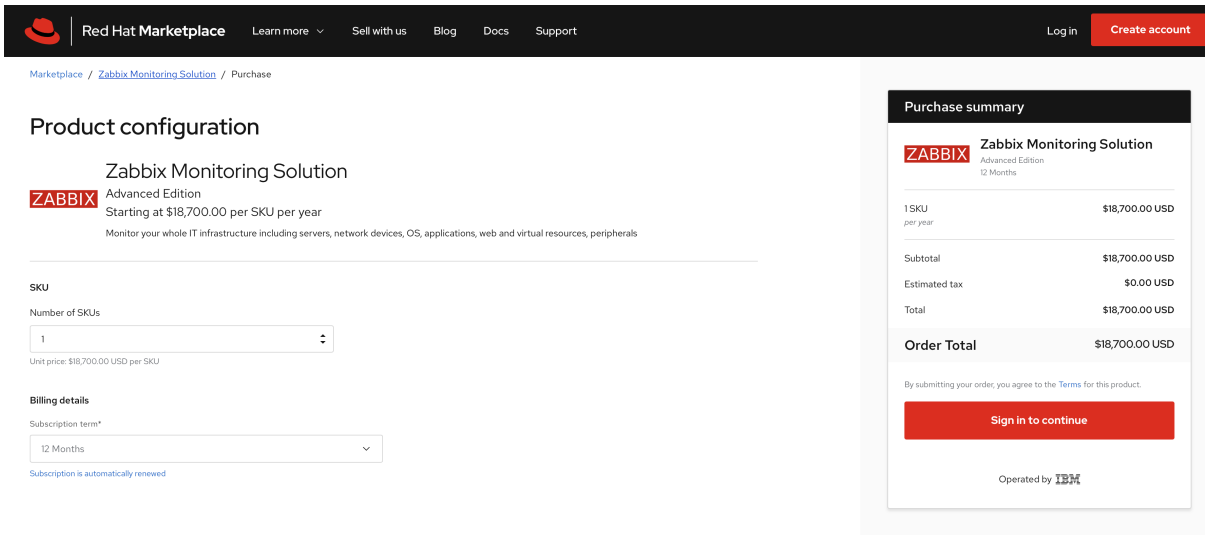
Attention:
Openshift needs to be registered with the Red Hat Marketplace portal.



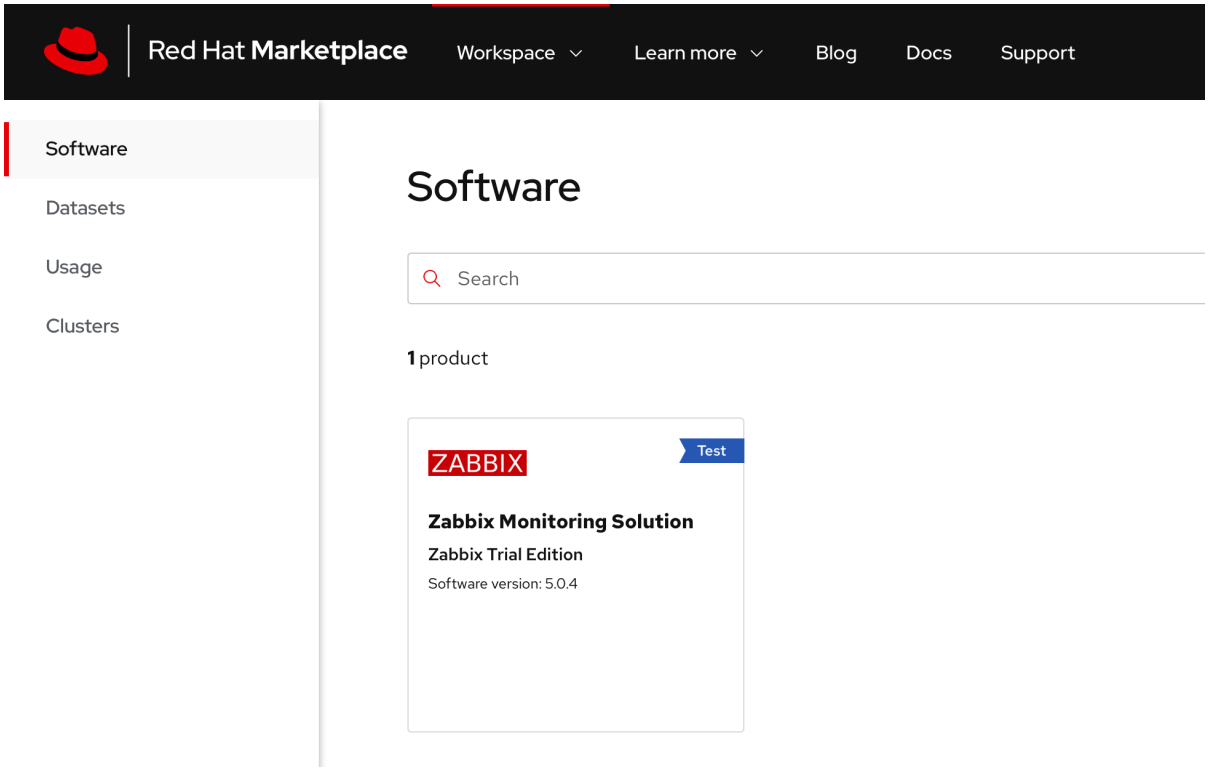
3. Select the most suitable install option.



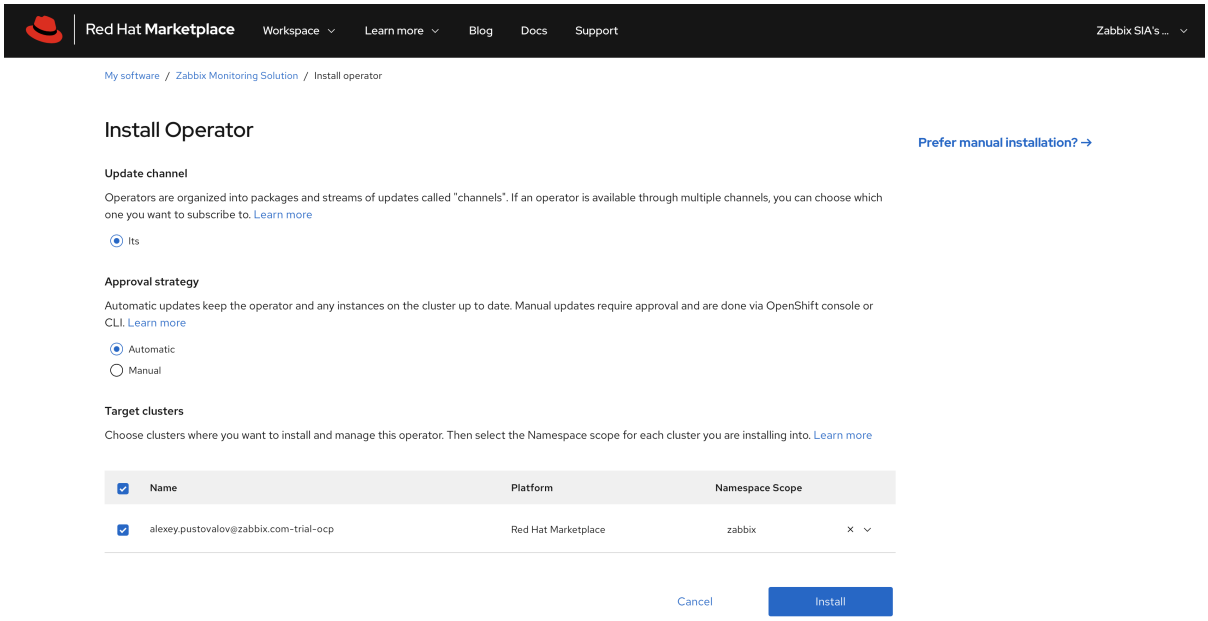
4. Specify the product configuration to fit your needs.



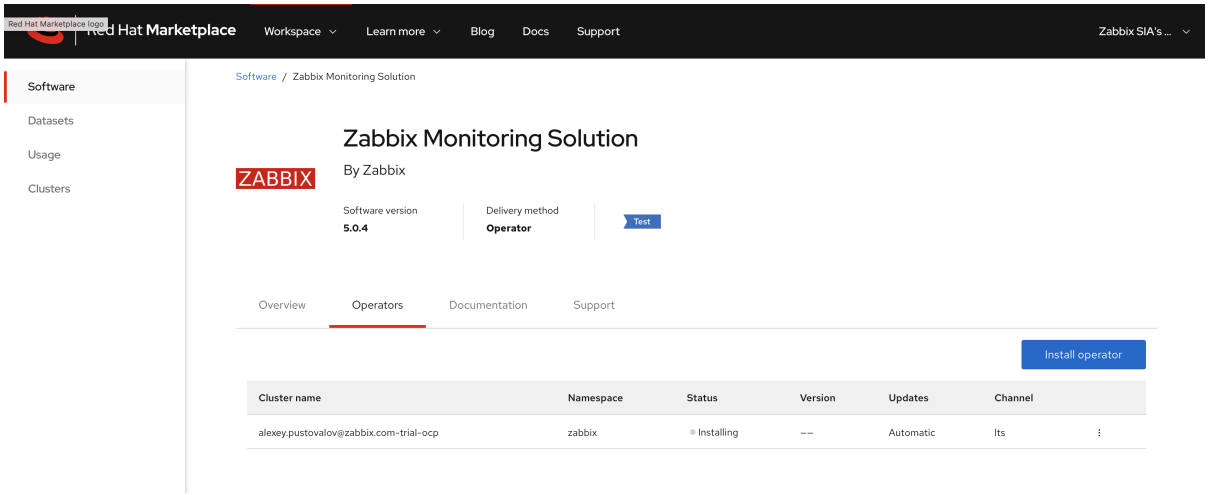
5. Navigate to your software within Red Hat Marketplace and install the Zabbix Operator software as specified in the image.



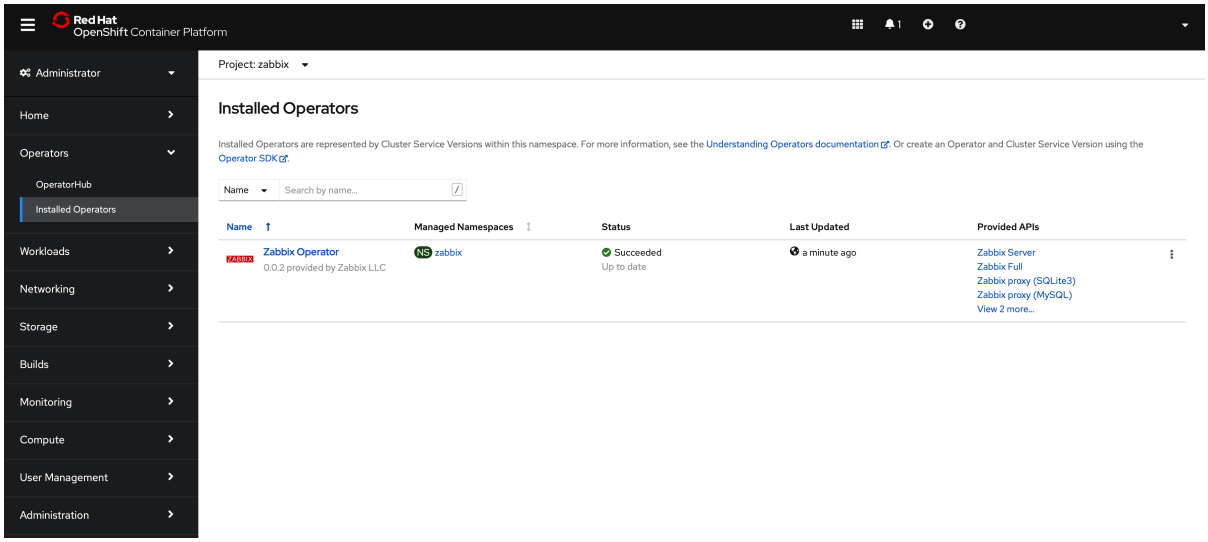
6. Install the Operator. Set the update approval strategy to *Automatic* to ensure that you always have the latest version of Zabbix components installed.



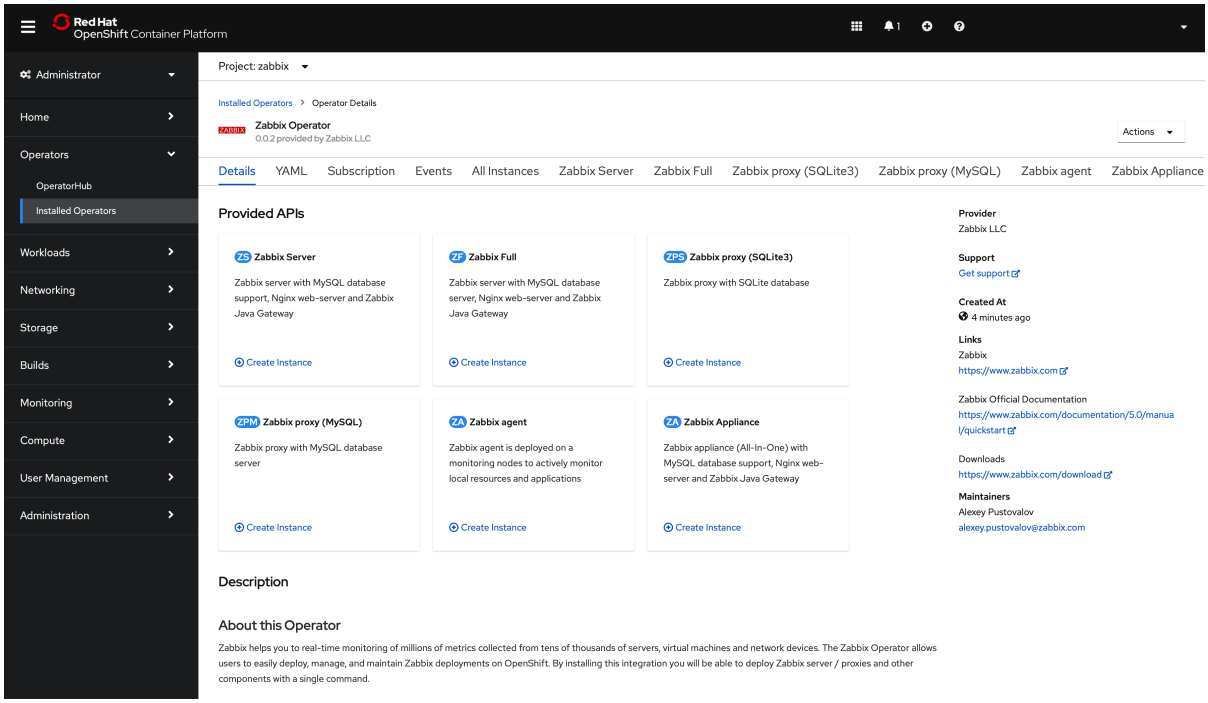
7. The Zabbix Operator is now installed into your specified cluster.



8. Go to Operators → Installed Operators.



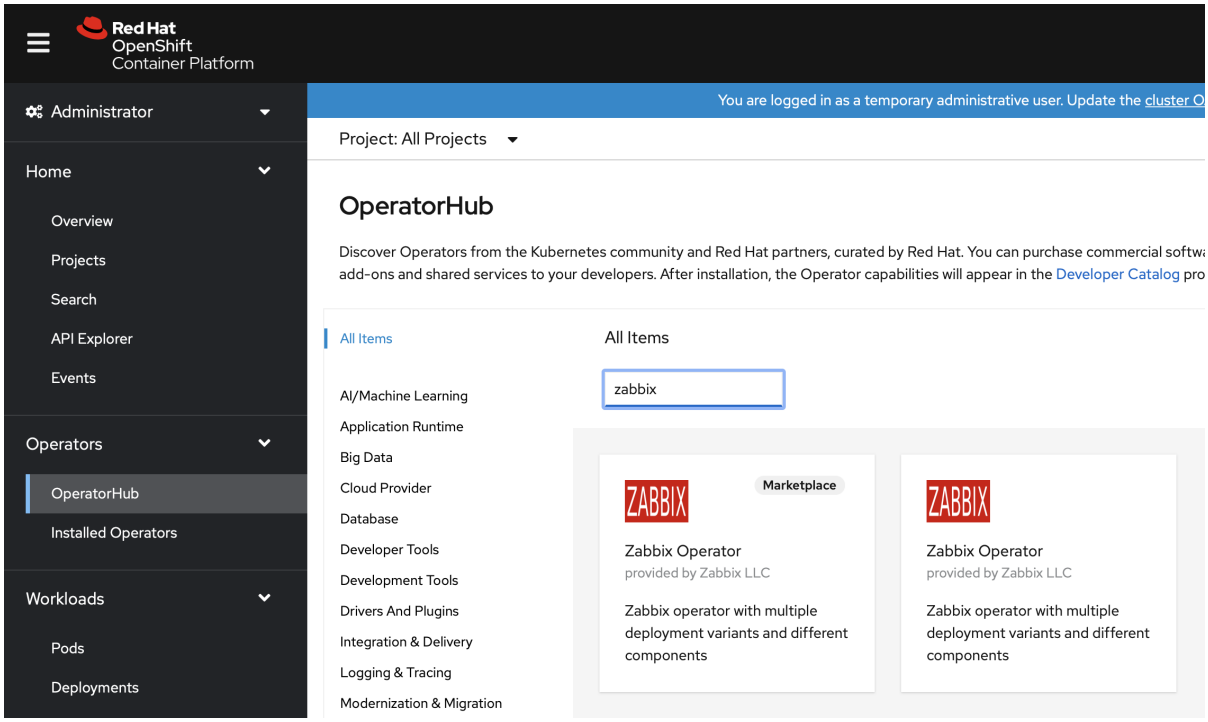
9. Open the ”Zabbix Operator” configuration page.



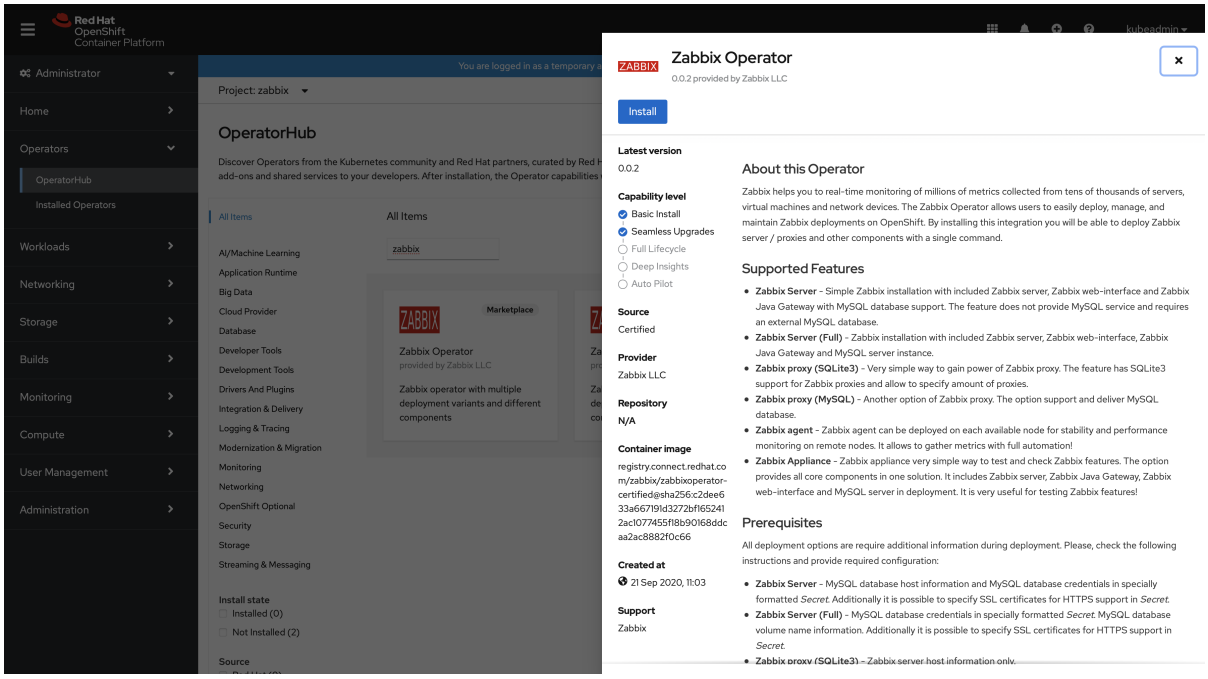
Using OperatorHub

Note:
If you have installed OpenShift in AWS ensure that the requisite ports are opened for the worker nodes' security group.

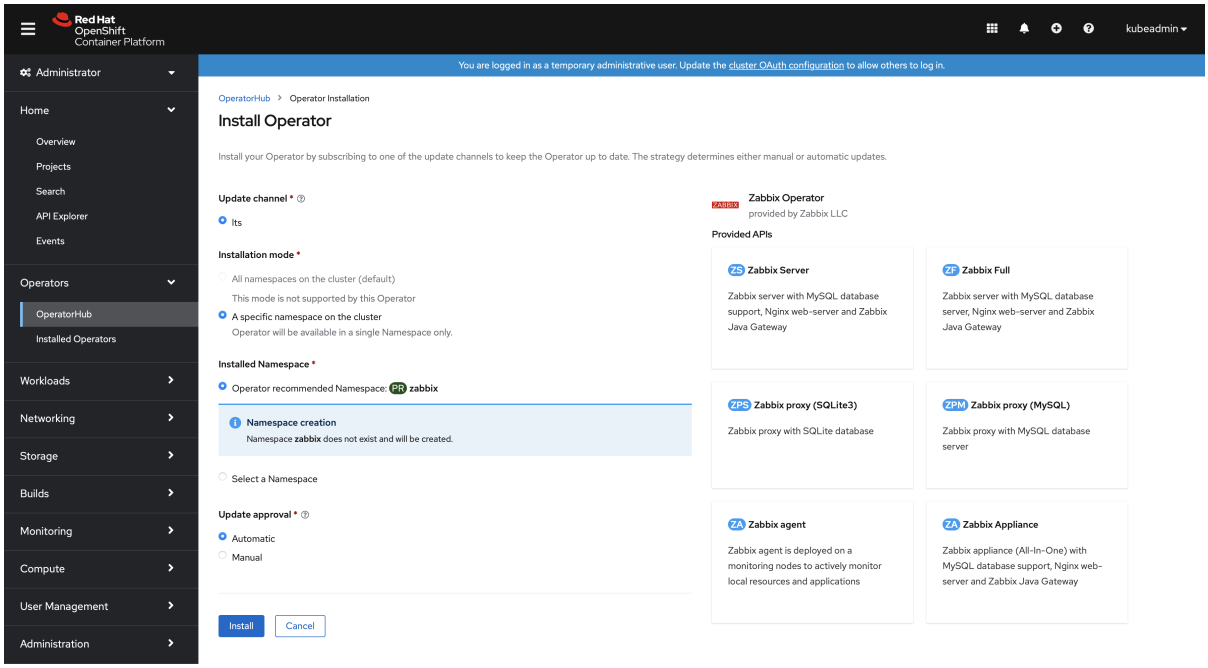
1. Select OperatorHub from the Operators submenu and search for Zabbix.



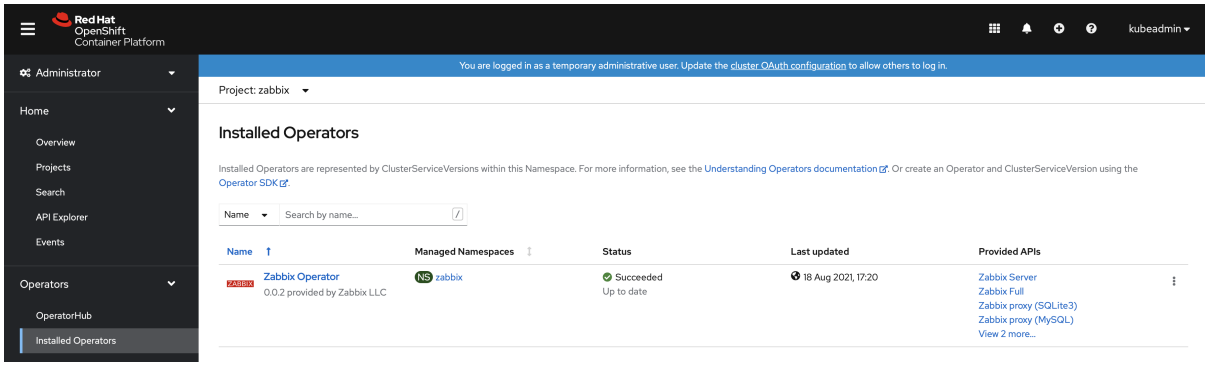
2. Select Zabbix Operator and click on Install.



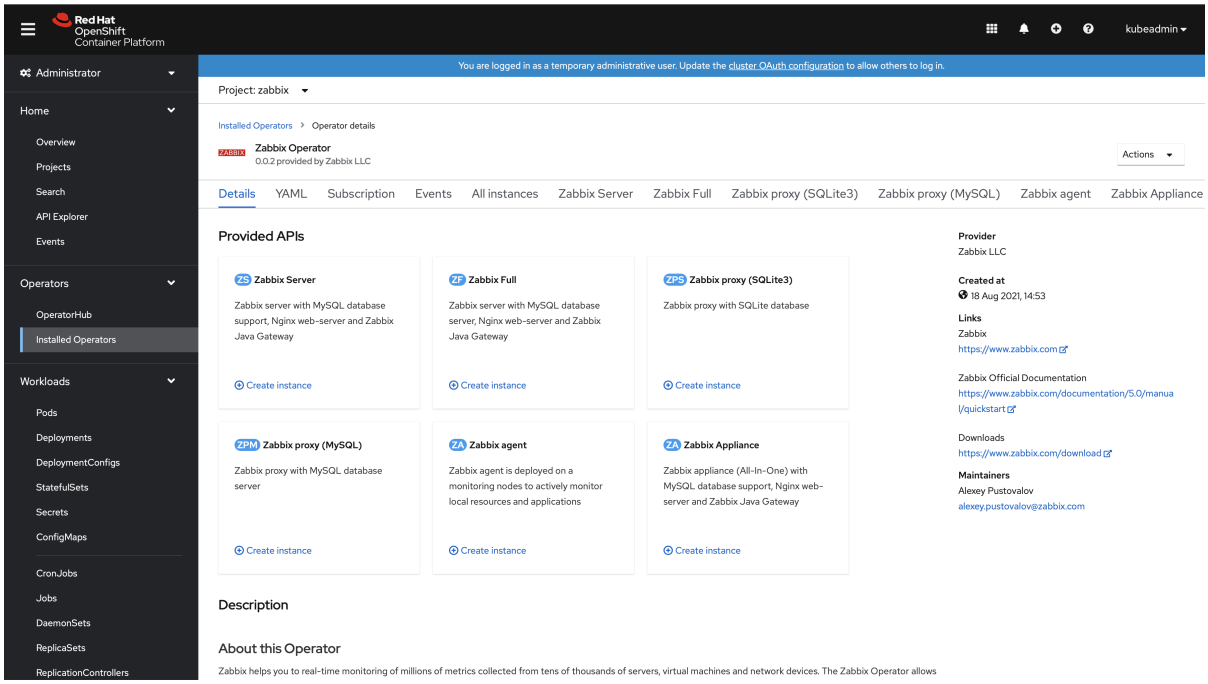
3. Select the installation options.



4. Go to Operators → Installed Operators.

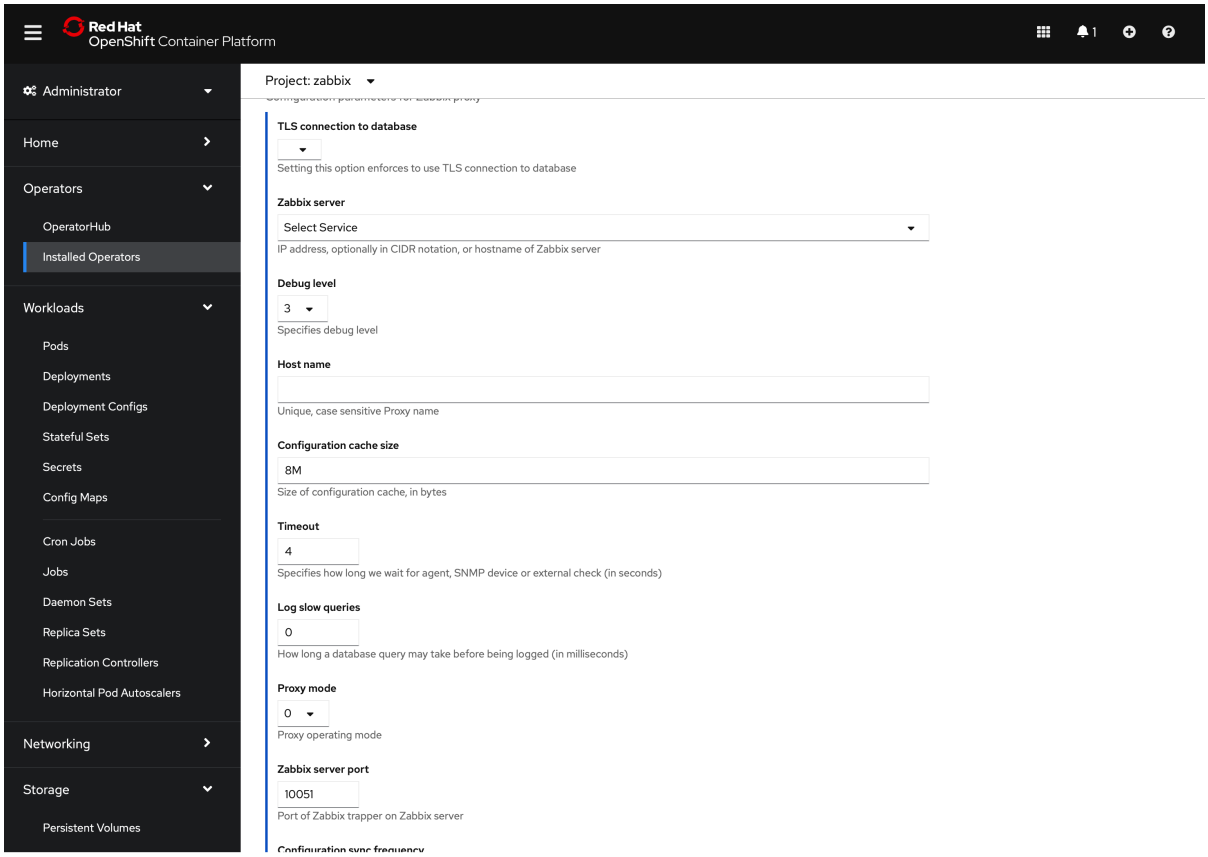


5. Open the ”Zabbix Operator” configuration page.

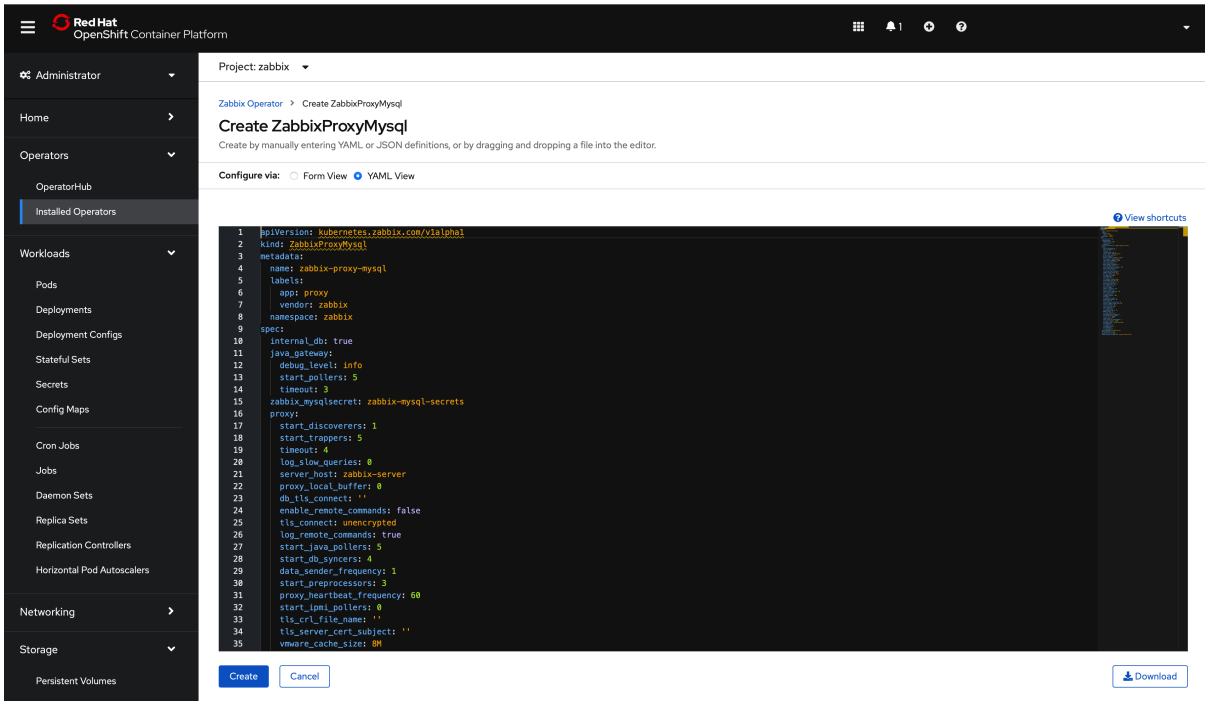


Configuration

Some of the operands (installation options) require additional resources to be created before. The following section describes these prerequisites. All possible configuration options are available during operand deployment. For example, **Zabbix proxy (MySQL)**:



The YAML section provides all available options with default values:



Zabbix server

This operand has a few prerequisites:

1. An existing MySQL database entry point - a MySQL database/cluster must be created before running the ”Zabbix Server” operand. For example, a standalone MySQL server with persistent volume:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
spec:
  accessModes:
```

```
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - image: mysql:8.0
        name: mysql
        args:
          - mysqld
          - '--character-set-server=utf8'
          - '--collation-server=utf8_bin'
          - '--default-authentication-plugin=mysql_native_password'
        env:
          # Use secret in real usage
        - name: MYSQL_ROOT_PASSWORD
          value: Welcome1!
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim
---
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  ports:
  - port: 3306
  selector:
    app: mysql
  clusterIP: None
```

Please, note that Zabbix does not support a utf8_mb4 charset and default caching_sha2_password authentication plugin.

2. MySQL credentials using secret - must be secret with mysql_root_password, mysql_zabbix_username and mysql_zabbix_password data. For example:

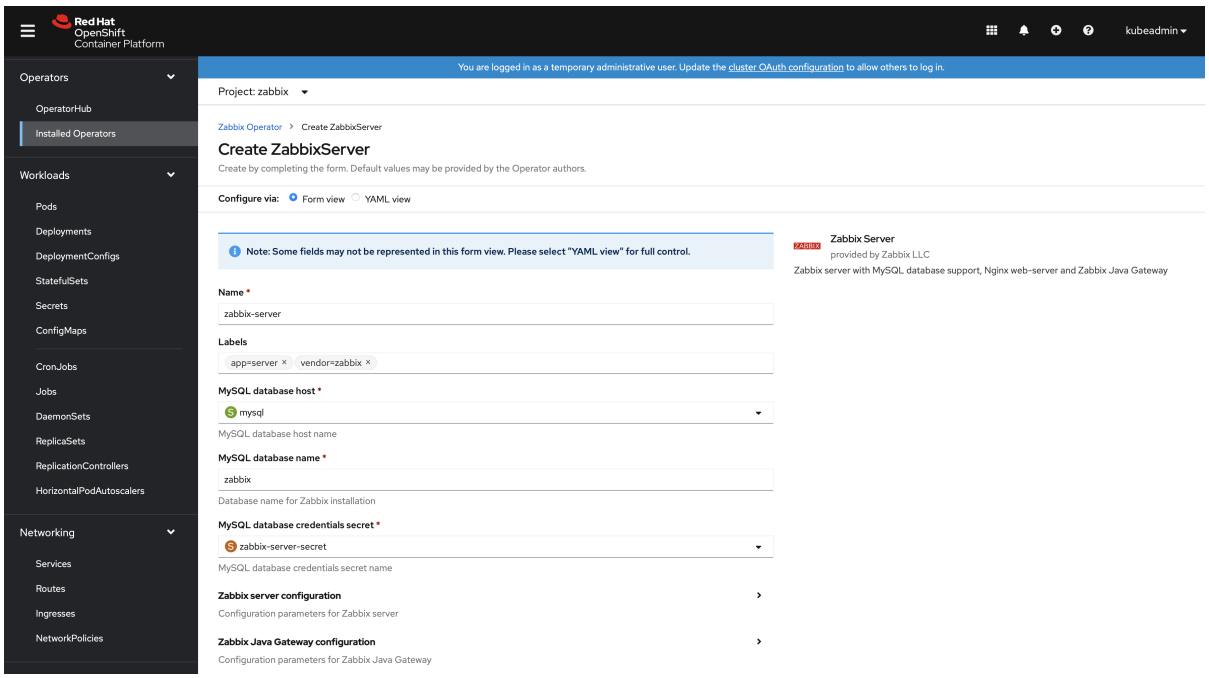
```
kind: Secret
apiVersion: v1
metadata:
  name: zabbix-server-secrets
data:
```

```
  mysql_root_password: V2VsY29tZTEh
  mysql_zabbix_password: emFiYml4X3N1cGVyIQ==
  mysql_zabbix_username: emFiYml4
type: Opaque
```

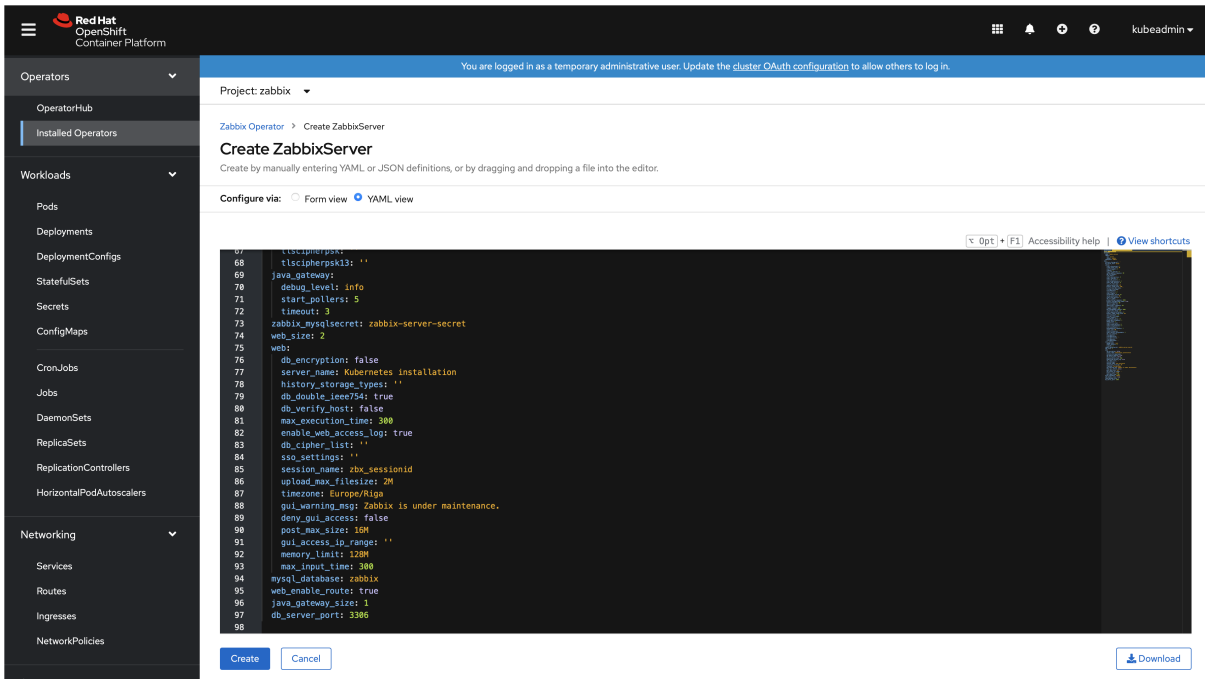
where all fields are encoded using base64. For example:

```
# echo -n "zabbix" | base64
emFiYml4Cg
```

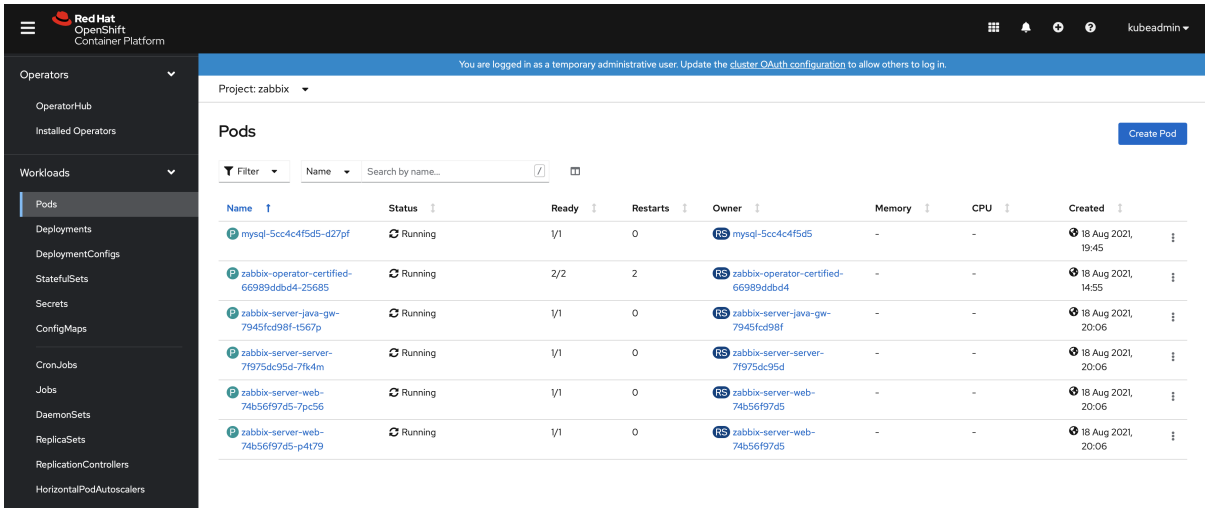
An example of ”Zabbix Server” operand configuration:



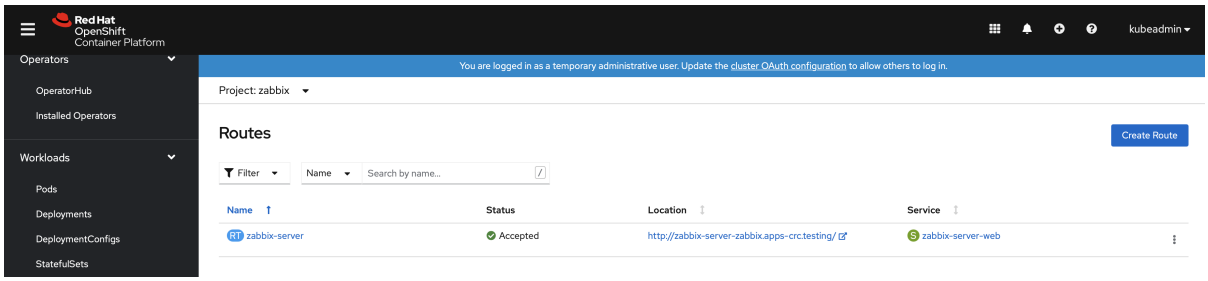
All configuration options are available using the form view, but it is possible to use the YAML view as well. For example:



Finally, the operand will create multiple pods. It is possible to examine them in the Workloads → Pods section:



The route for Zabbix web interface is located under Networking → Routes. The URL provides access to the Zabbix web interface. In the following example it is `http://zabbix-server-zabbix.apps-crc.testing/`:



Zabbix full

This operand has a few prerequisites:

1. MySQL volume claim - must be persistent volume claim. For example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: zabbix-database
  namespace: zabbix
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 50Gi
```

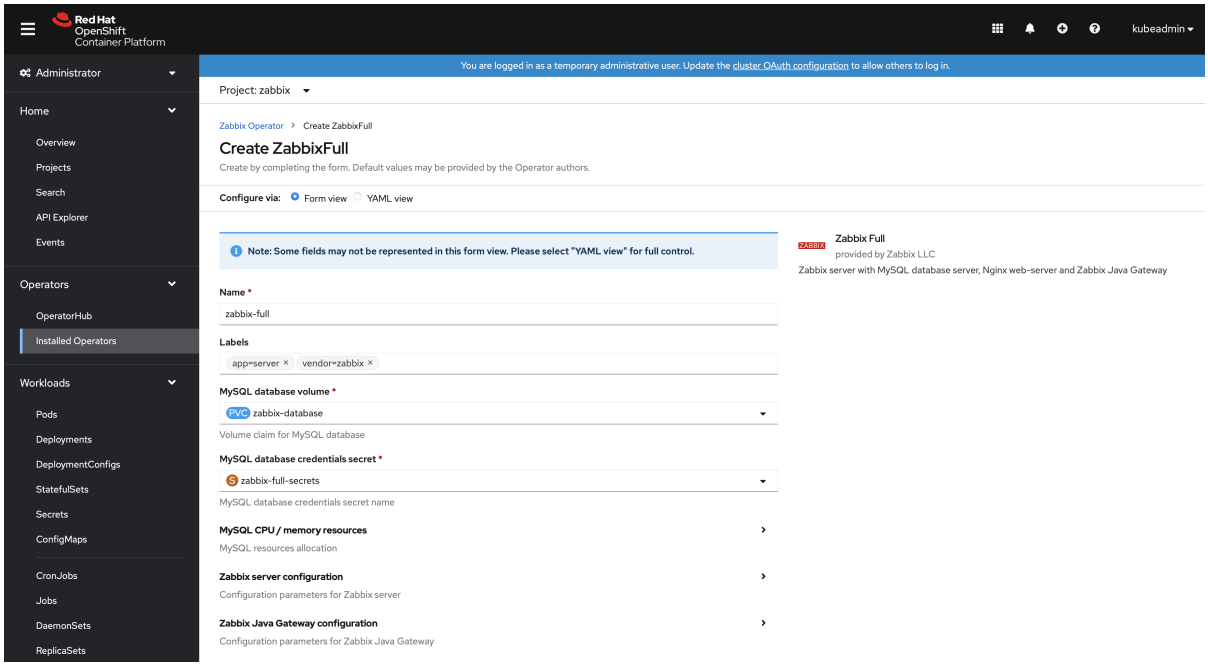
2. MySQL credentials using secret - must be secret with mysql_root_password, mysql_zabbix_username and mysql_zabbix_password data. For example:

```
kind: Secret
apiVersion: v1
metadata:
  name: zabbix-full-secrets
data:
  mysql_root_password: V2VsY29tZTEh
  mysql_zabbix_password: emFiYml4X3N1cGVyIQ==
  mysql_zabbix_username: emFiYml4
type: Opaque
```

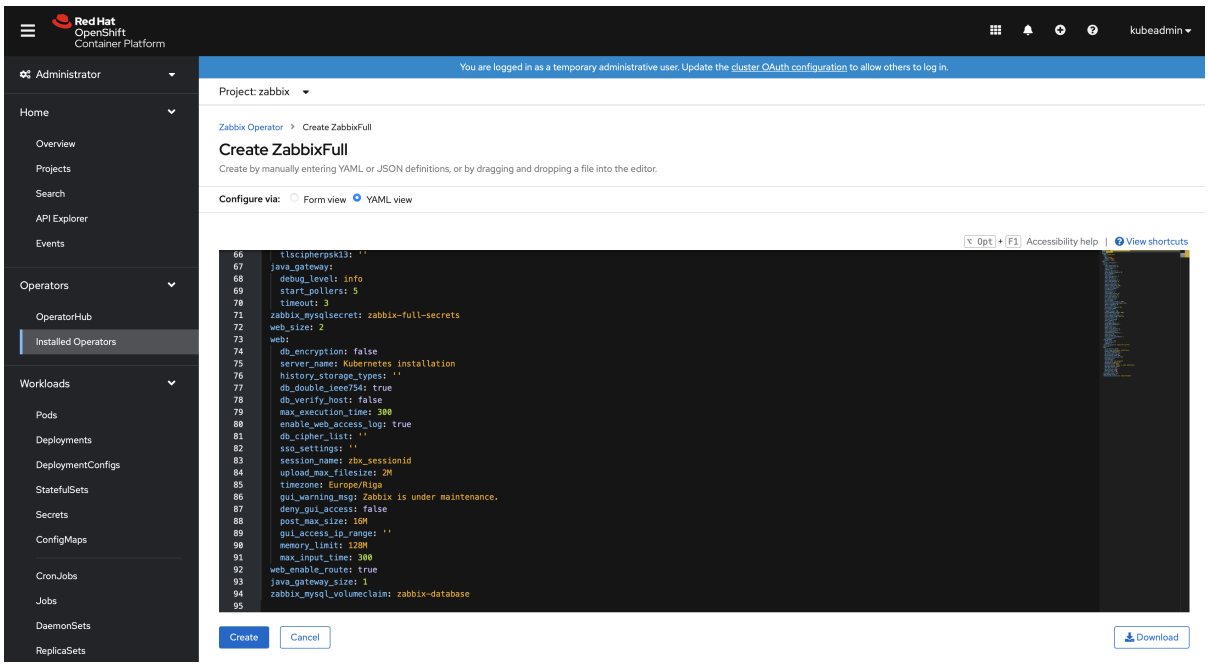
where all fields are encoded using base64. For example:

```
# echo -n "zabbix" | base64
emFiYml4Cg
```

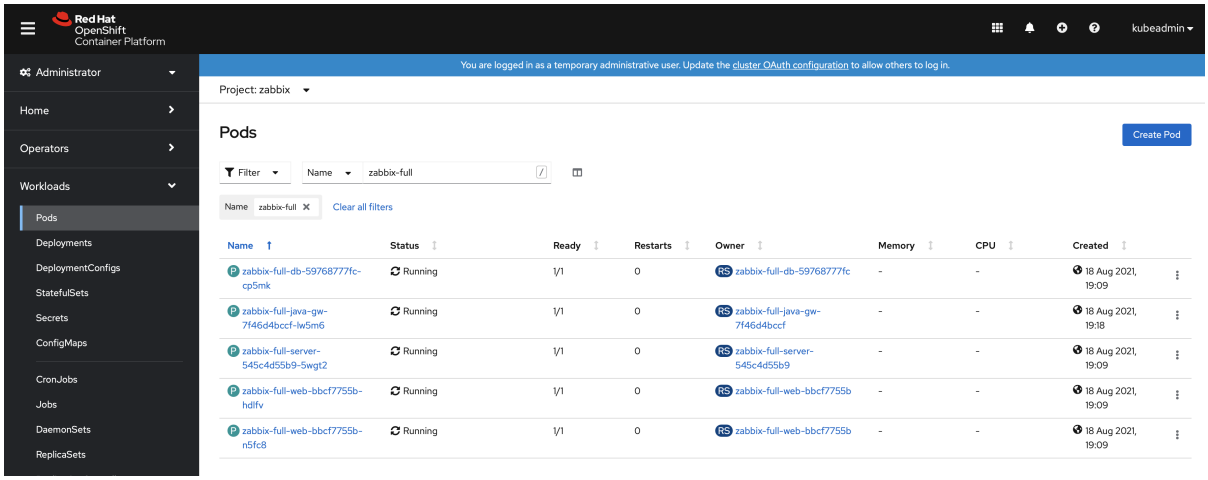
An example of ”Zabbix Full” operand configuration:



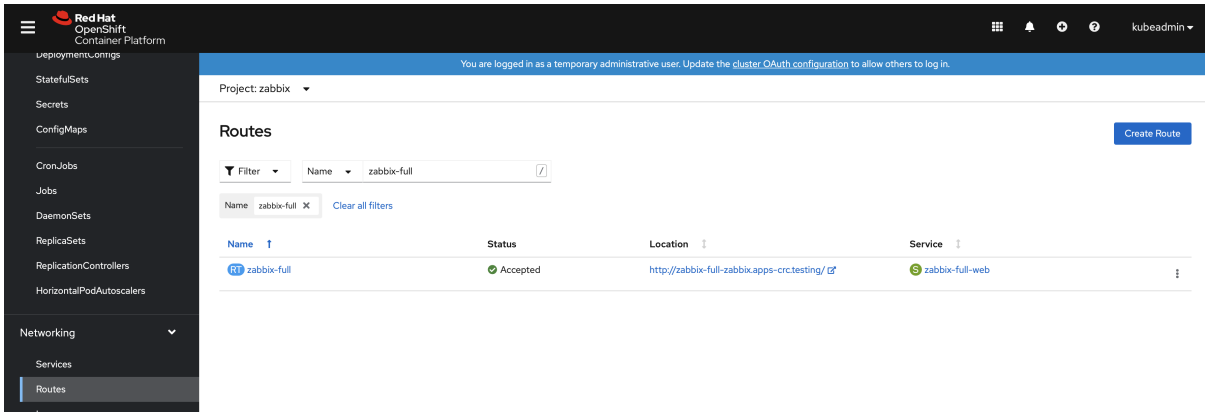
All configuration options are available using the form view, but it is possible to use the YAML view as well. For example:



Finally, the operand will create multiple pods. It is possible to examine them in the Workloads → Pods section:

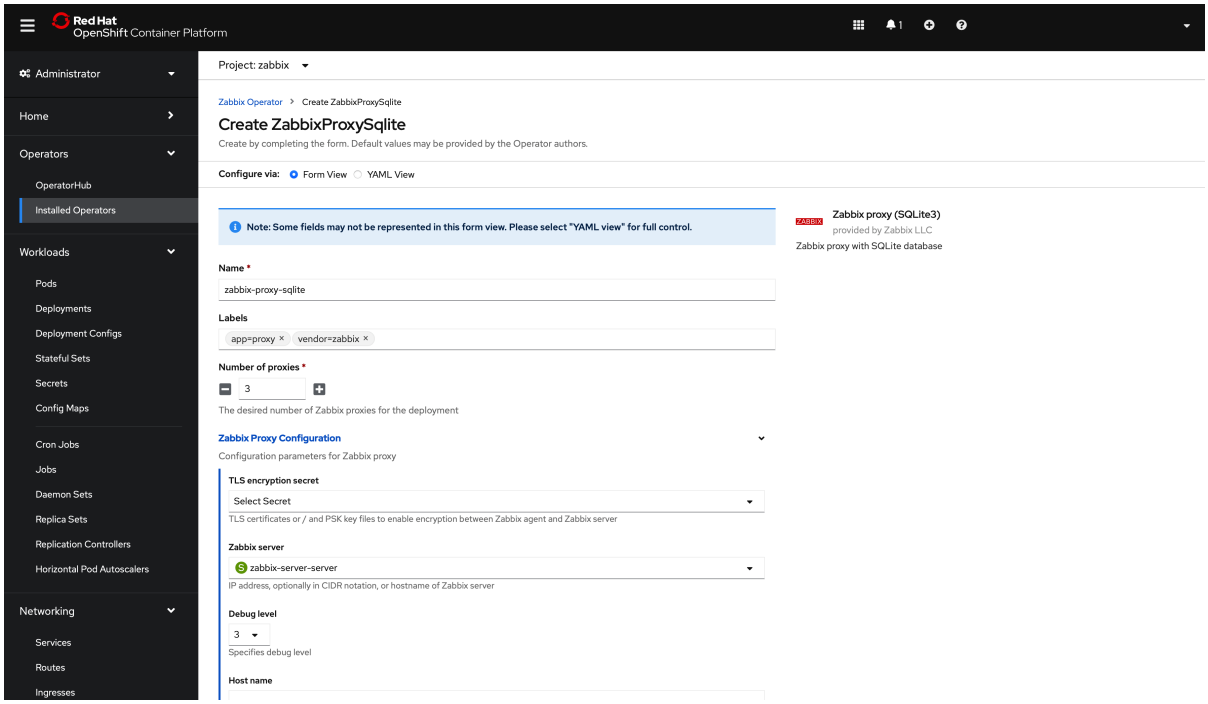


The route for Zabbix web interface is located under Networking → Routes. The URL provides access to the Zabbix web interface. In the following example it is `http://zabbix-full-zabbix.apps-crc.testing/`:

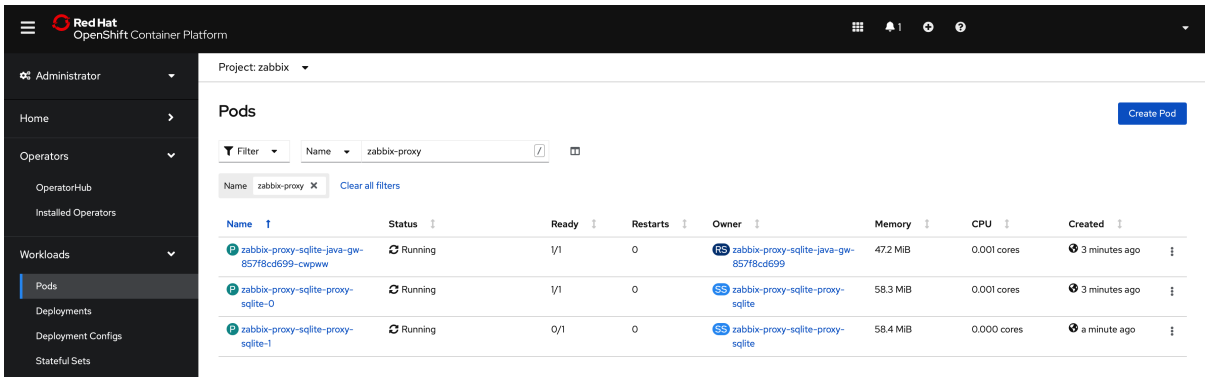


Zabbix proxy (SQLite3)

An example of ”Zabbix proxy (SQLite3)” operand configuration:



Finally, the operand will create multiple pods. It is possible to examine them in the Workloads → Pods section:

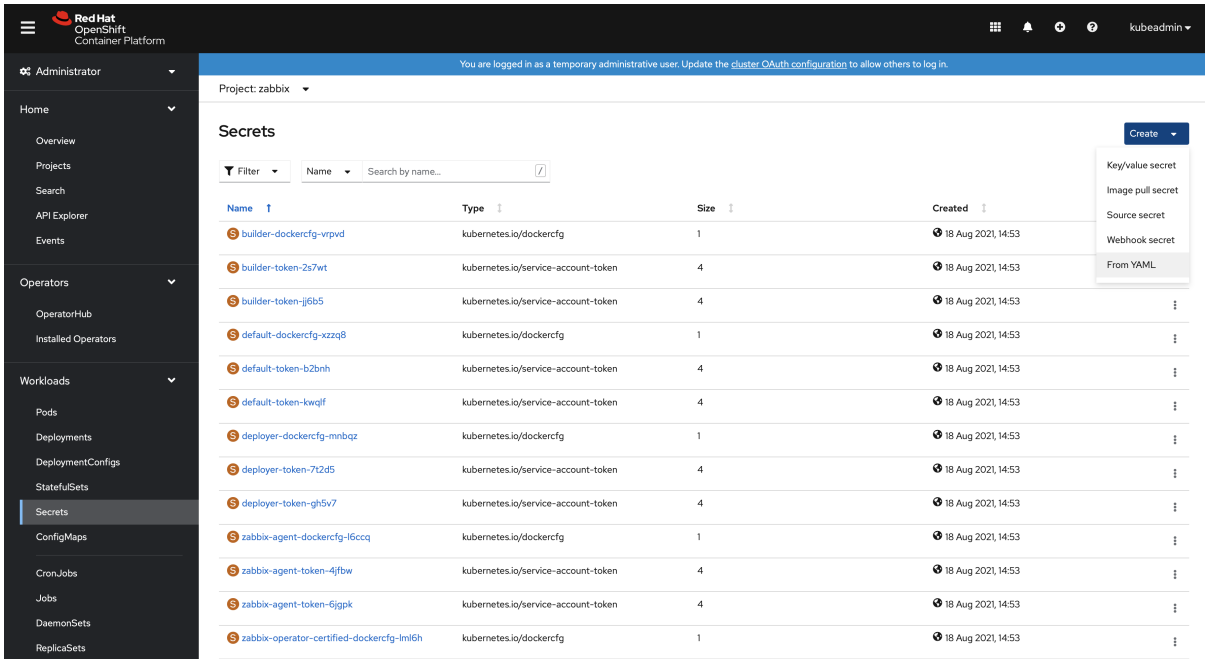


Additional information

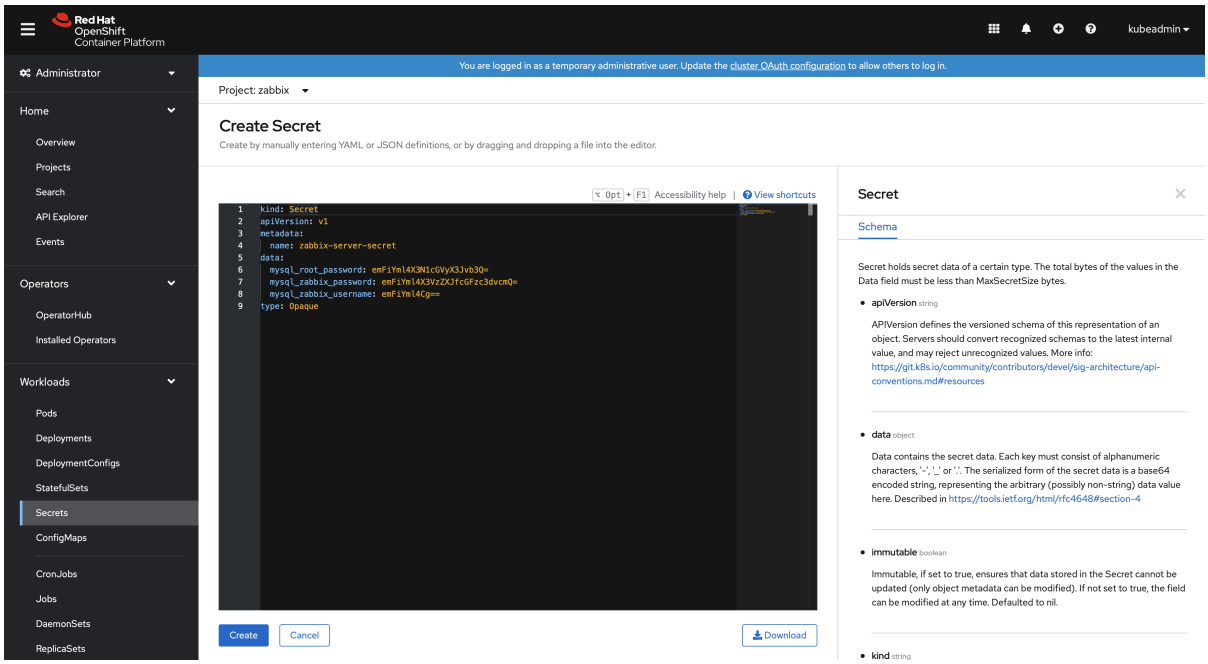
Creating new secret

The following procedure describes how to create a new secret using Openshift Console.

1. Open the Workloads → Secrets section and switch project to the Zabbix Operator project (by default, ”zabbix”).



2. Create a new secret using the *From YAML* option.



SSL certificates for HTTPS

It is possible to enable HTTPS directly in the Zabbix web interface pods. In this case create the following secret using the YAML option:

```
kind: Secret
apiVersion: v1
metadata:
  name: zabbix-web-sslsecret
data:
  ssl.crt: >-
   < ssl.crt data>
  ssl.key: >-
     < ssl.key data >
  dhparam.pem: >-
   <  dhparam.pem data >
```

The names of certificates and DH Parameters file are static. Please use the listed in the above example only!

MySQL database certificate base encryption

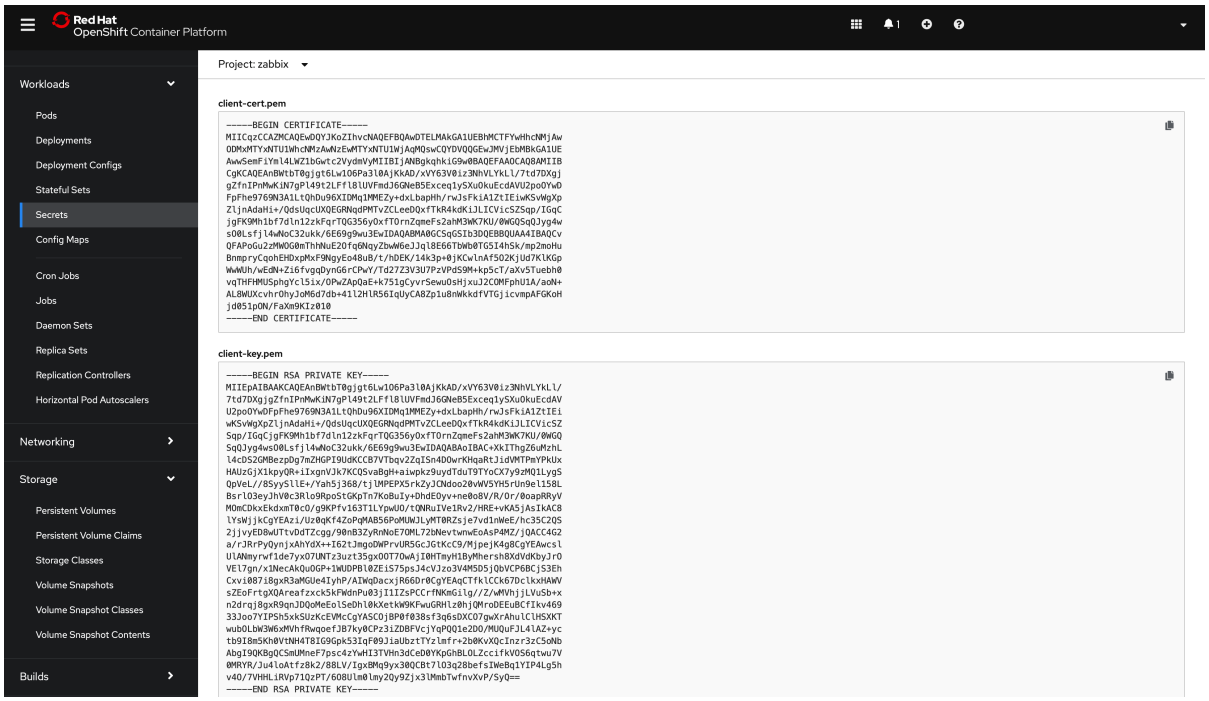
MySQL database side:

```
apiVersion: v1
data:
 root-ca.pem: >-
  < root-ca.pem data>
 server-cert.pem: >-
  < server-cert.pem data>
 server-key.pem: >-
  < server-key.pem data>
kind: Secret
metadata:
 name: zabbix-db-server-tls-secret
type: Opaque
```

Zabbix components side:

```
apiVersion: v1
data:
 client-cert.pem: >-
  < client-cert.pem data>
 client-key.pem: >-
  < client-key.pem data>
 root-ca.pem: >-
  < root-ca.pem data>
kind: Secret
metadata:
 name: zabbix-db-client-tls-secret
type: Opaque
```

Certificates must include "-----BEGIN RSA PRIVATE KEY-----" and "-----END RSA PRIVATE KEY-----". For example:



Then, during deployment, in the Zabbix component section and MySQL server (if using built-in server) choose the proper ”TLS connection to database” option value and the ”MySQL database certificates (client)” secret value.

Known issues

1. Zabbix agent does not have the possibility to determine proper node name. It always has dynamic hostname.

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.

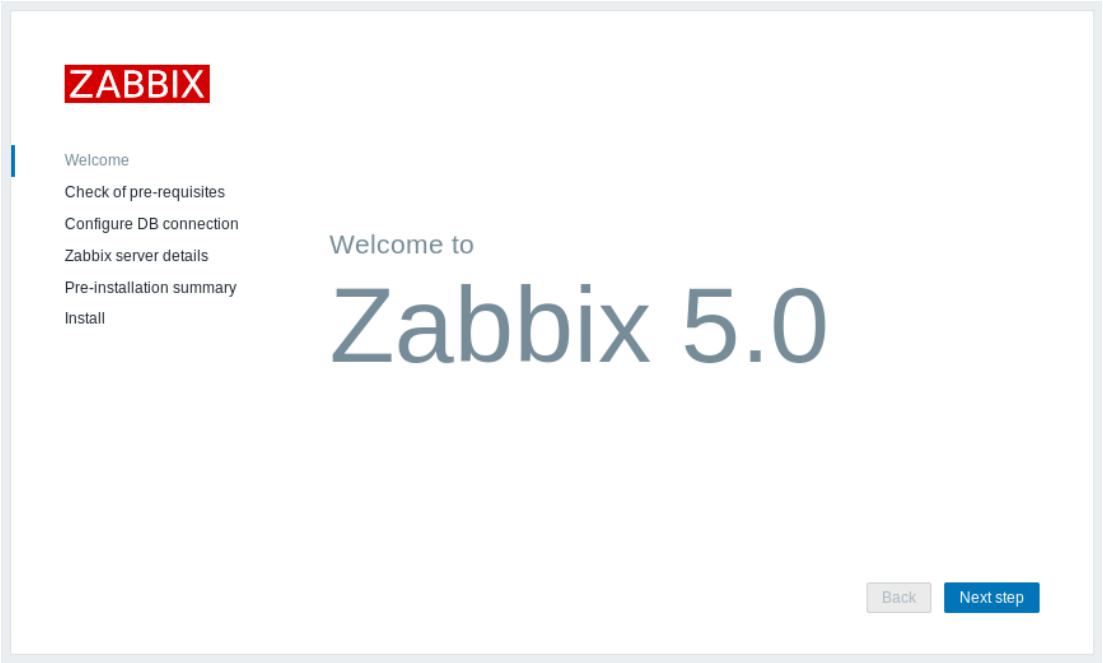
Note:
You can find out more about setting up SSL for Zabbix frontend by referring to these best practices.

Welcome screen

Open Zabbix frontend URL in a browser. If you have installed Zabbix from packages, the URL is:

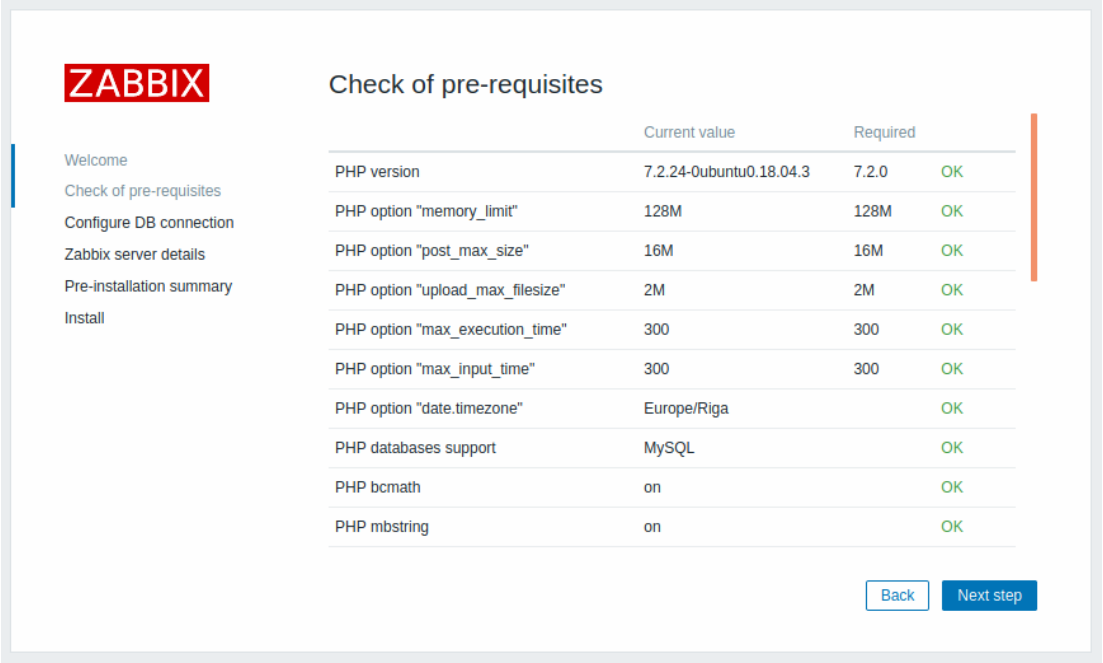
- for Apache: *http://<server_ip_or_name>/zabbix*
- for Nginx: *http://<server_ip_or_name>*

You should see the first screen of the frontend installation wizard.



Check of pre-requisites

Make sure that all software prerequisites are met.



Pre-requisite	Minimum value	Description
PHP version	7.2.0	

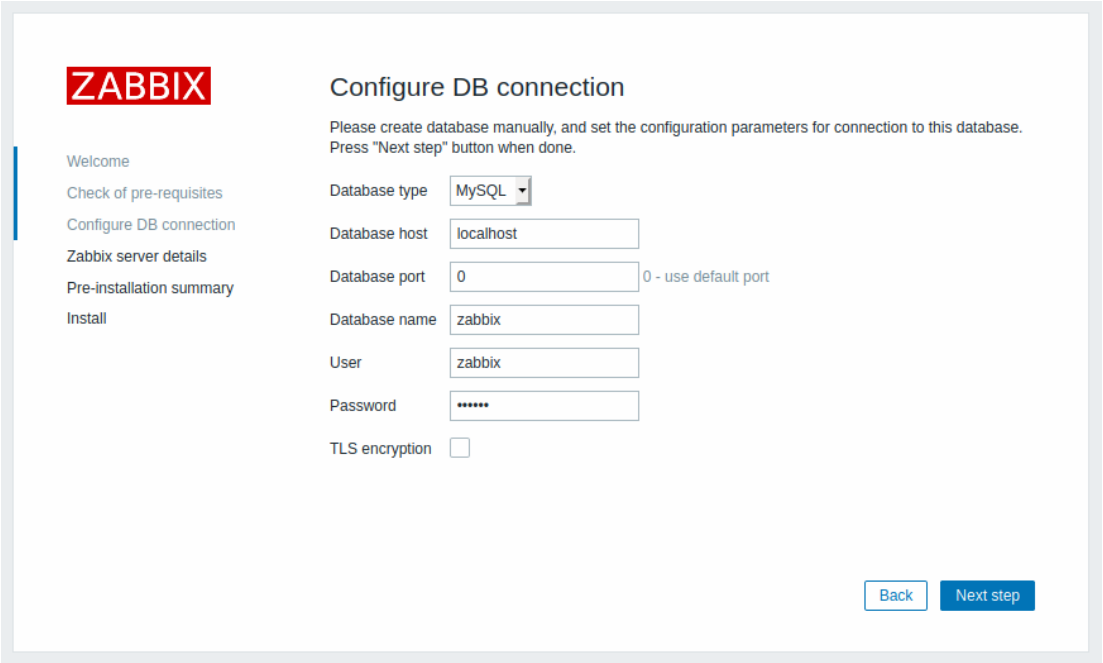
Pre-requisite	Minimum value	Description
PHP memory_limit option	128MB	In php.ini: memory_limit = 128M
PHP post_max_size option	16MB	In php.ini: post_max_size = 16M
PHP upload_max_filesize option	2MB	In php.ini: upload_max_filesize = 2M
PHP max_execution_time option	300 seconds (values 0 and -1 are allowed)	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
bcmath		php-bcmath
mbstring		php-mbstring
PHP mbstring.func_overload option	must be disabled	In php.ini: mbstring.func_overload = 0
sockets		php-net-socket. Required for user script support.
gd	2.0.28	php-gd. PHP GD extension must support PNG images (*--with-png-dir*), JPEG (*--with-jpeg-dir*) images and FreeType 2 (*--with-freetype-dir*).
libxml	2.6.15	php-xml
xmlwriter		php-xmlwriter
xmlreader		php-xmlreader
ctype		php-ctype
session		php-session
gettext		php-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.

Attention:
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

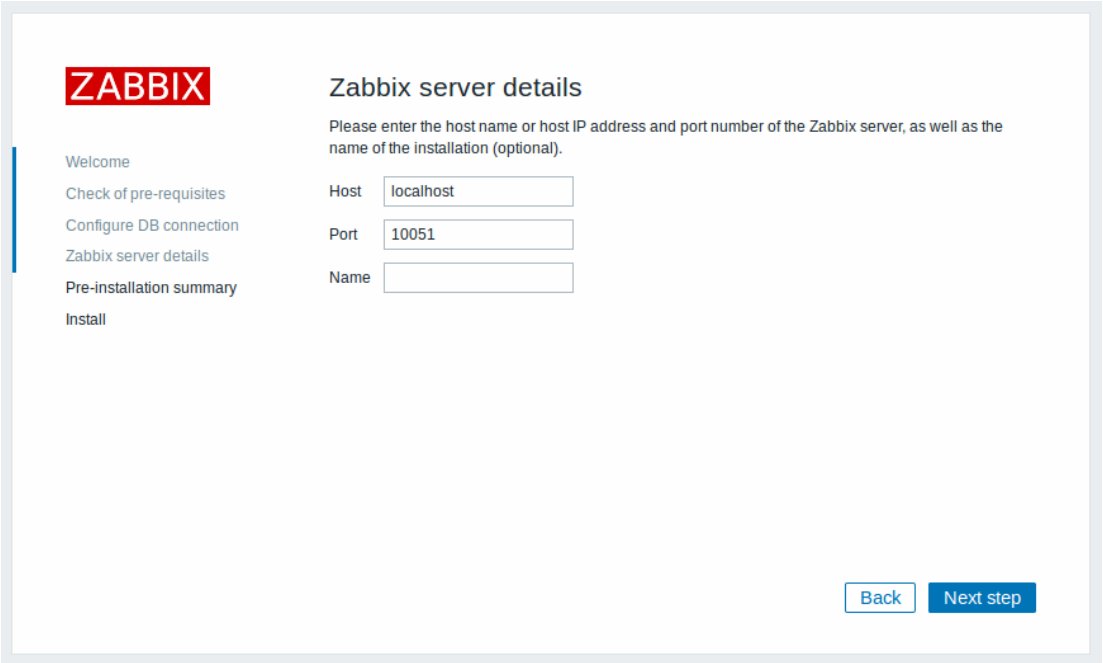
Enter details for connecting to the database. Zabbix database must already be created.



If the *TLS encryption* option is checked, then additional fields for configuring the TLS connection to the database appear in the form (MySQL or PostgreSQL only).

Zabbix server details

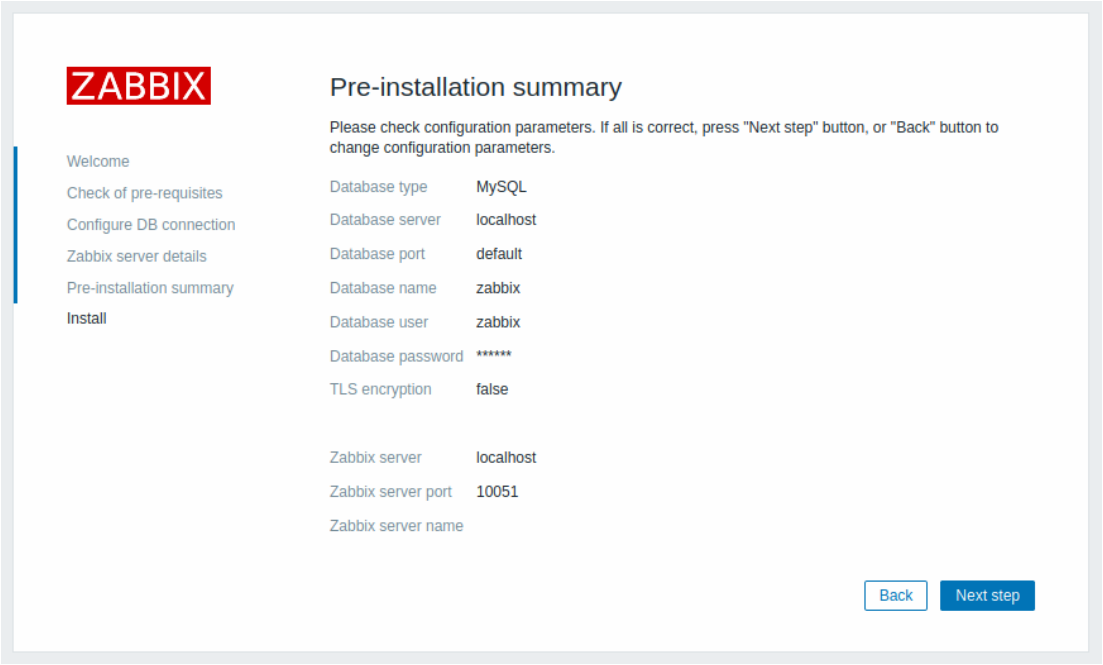
Enter Zabbix server details.



Entering a name for Zabbix server is optional, however, if submitted, it will be displayed in the menu bar and page titles.

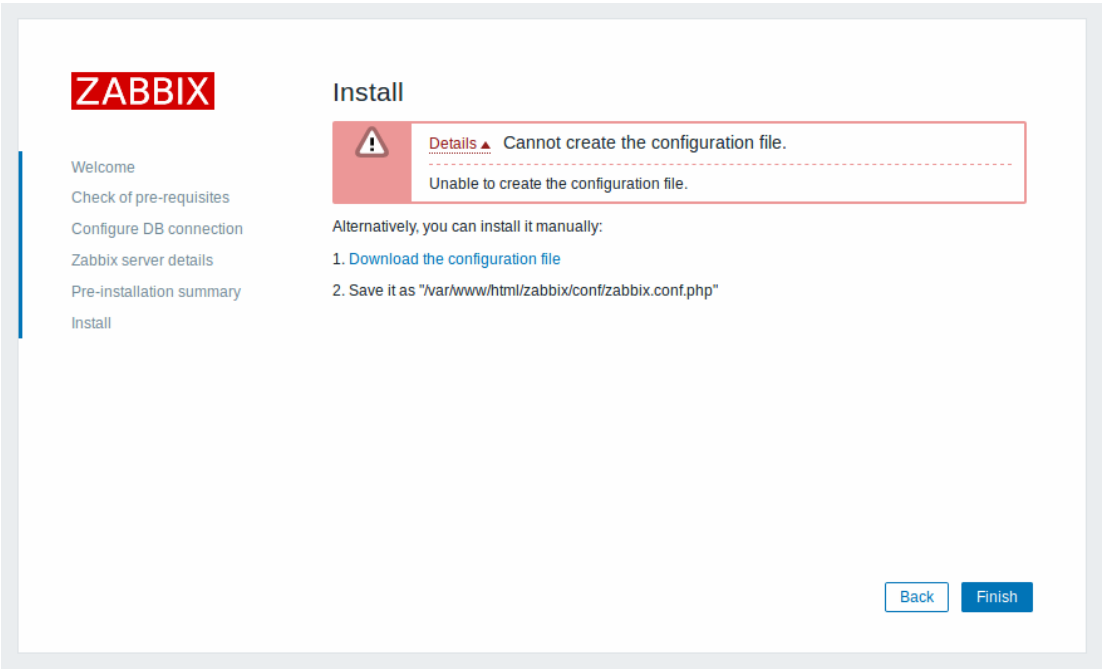
Pre-installation summary

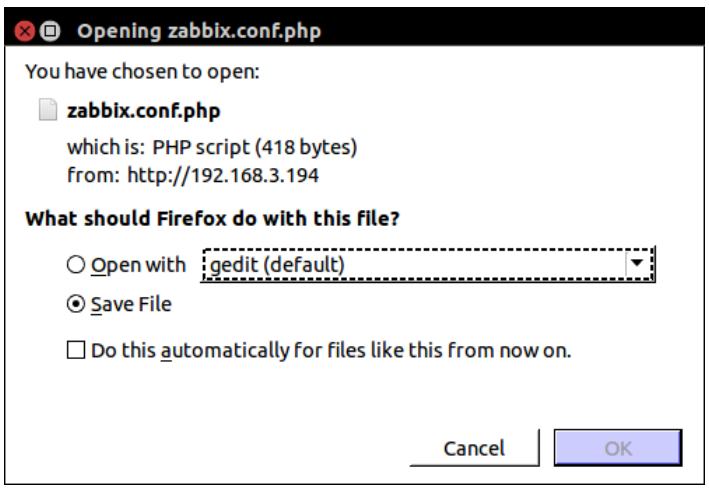
Review a summary of settings.



Install

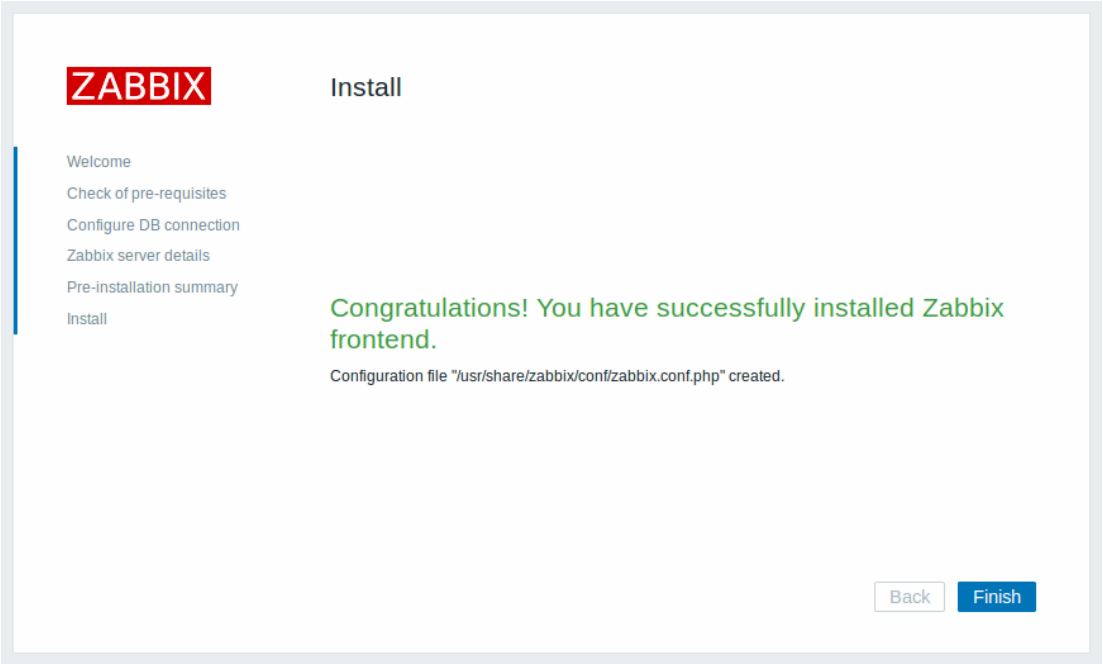
If installing from sources, download the configuration file and place it under conf/ in the webserver HTML documents subdirectory where you copied Zabbix PHP files to.





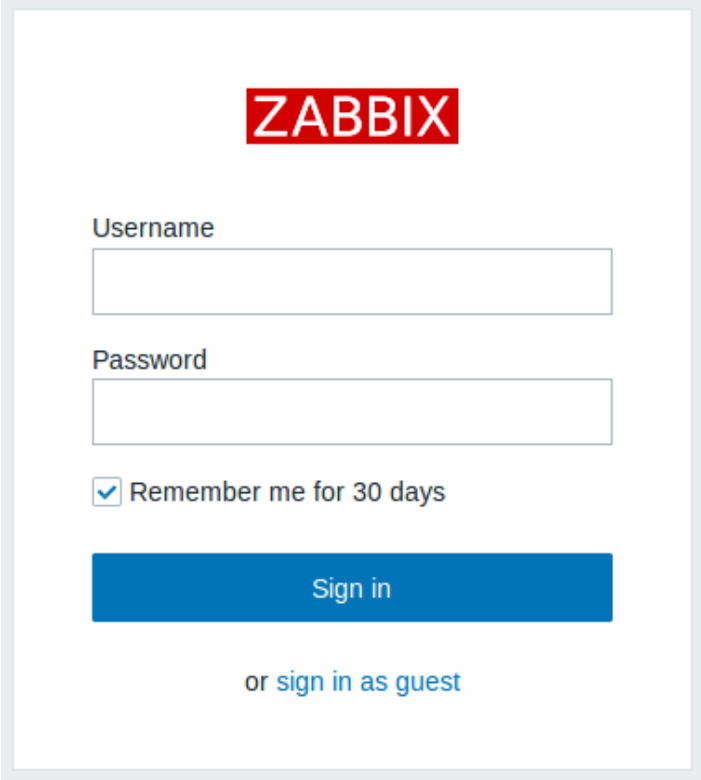
Note:
Providing the webserver user has write access to conf/ directory the configuration file would be saved automatically and it would be possible to proceed to the next step right away.

Finish the installation.



Log in

Zabbix frontend is ready! The default user name is **Admin**, password **zabbix**.



Proceed to getting started with Zabbix.

1 RHEL/CentOS 7 frontend installation

Overview

The current Zabbix frontend requires PHP version 7.2 or higher. Unfortunately, RHEL/CentOS 7 only provides PHP 5.4 by default. This page describes the proposed method of installing Zabbix frontend on RHEL/CentOS 7.

See also: Updating to PHP 7.3

Using PHP and Nginx from Red Hat Software Collections

If you do a clean installation of Zabbix 5.0 using official packages from repo.zabbix.com, you may notice that frontend packages are missing when searching for Zabbix with yum.

```
zabbix-agent.x86_64 : Old Zabbix Agent
zabbix-get.x86_64 : Zabbix Get
zabbix-java-gateway.x86_64 : Zabbix java gateway
zabbix-js.x86_64 : Zabbix JS
zabbix-proxy-mysql.x86_64 : Zabbix proxy for MySQL or MariaDB database
zabbix-proxy-pgsql.x86_64 : Zabbix proxy for PostgreSQL database
zabbix-proxy-sqlite3.x86_64 : Zabbix proxy for SQLite3 database
zabbix-release.noarch : Zabbix repository configuration
zabbix-sender.x86_64 : Zabbix Sender
zabbix-server-mysql.x86_64 : Zabbix server for MySQL or MariaDB database
zabbix-server-pgsql.x86_64 : Zabbix server for PostgreSQL database
```

This is due to the fact that frontend packages were moved to a dedicated `frontend` sub-repository.
However, they can still be installed, if PHP 7.2 dependencies are provided.

Note:
For your convenience, any direct dependency on PHP has been removed from the main `zabbix-web` package. This gives more flexibility in choosing the way to resolve PHP 7.2 dependency.

It is recommended to use PHP packages from Red Hat Software Collections.
To enable them run:

On RHEL

```
# yum-config-manager --enable rhel-server-rhscl-7-rpms
```

On CentOS

```
# sudo yum install centos-release-scl
```

On Oracle Linux

```
# yum install scl-utils
# yum install oraclelinux-release-el7
# /usr/bin/ol_yum_configure.sh
# yum-config-manager --enable software_collections
# yum-config-manager --enable ol7_latest ol7_optional_latest
```

At this point

```
# yum list rh-php7\*
```

Should return a list of new rh-php7* packages.

Next, edit `/etc/yum.repos.d/zabbix.repo` file (if there is no such file, install zabbix-release first). Enable `zabbix-frontend` repository.

```
[zabbix-frontend]
...
enabled=1
...
```

Replace `enabled=0` with `enabled=1`.

At this stage searching for `Zabbix` with `yum` shall return `zabbix-web` package together with four new packages. These are:

```
zabbix-nginx-conf-scl.noarch : Zabbix frontend configuration for Nginx (scl version)
zabbix-web-deps-scl.noarch : Convenience package for installing PHP dependencies of zabbix-web package fro
zabbix-web-mysql-scl.noarch : Zabbix web frontend for MySQL (scl version)
zabbix-web-pgsql-scl.noarch : Zabbix web frontend for PostgreSQL (scl version)
```

Install either zabbix-web-mysql-scl for MySQL or zabbix-web-pgsql-scl package for PostgreSQL. Also install also zabbix-apache-conf-scl or zabbix-nginx-conf-scl package, depending on web server used.

Note:
In Zabbix 4.4 support for Nginx was added, but the web server was not available in the official RHEL/CentOS 7 repositories. Thus, it had to be provided by the user via third-party repositores, in particular epel. With Zabbix 5.0, if you choose to use Red Hat Software Collections, there is no need to use any third-party repositories, since Nginx is available in SCL. Simply install zabbix-nginx-conf-scl package.

Technical details of new packages

zabbix-web-deps-scl

This package pulls common PHP dependenencies of Zabbix frontend from Red Hat Software Collections.

```
# repoquery --requires zabbix-web-deps-scl
rh-php72
rh-php72-php-bcmath
rh-php72-php-fpm
rh-php72-php-gd
rh-php72-php-ldap
rh-php72-php-mbstring
rh-php72-php-xml
```

It also contains php-fpm pool for Zabbix, since in this configuration frontend works through fastcgi with both Apache and Nginx. Configuration file is located at /etc/opt/rh/rh-php72/php-fpm.d/zabbix.conf.

zabbix-web-mysql-scl

Meta package that pulls zabbix-web package and MySQL module for PHP, together with common PHP dependencies.

```
# repoquery --requires zabbix-web-mysql-scl
rh-php72-php-mysqlnd
zabbix-web
zabbix-web-deps-scl
```

zabbix-web-pgsql-scl

Meta package that pulls zabbix-web package and PostgreSQL module for PHP, together with common PHP dependencies.

```
# repoquery --requires zabbix-web-pgsql-scl
rh-php72-php-pgsql
zabbix-web
zabbix-web-deps-scl
```

zabbix-apache-conf-scl

This package pulls apache and contains /etc/httpd/cond.d/zabbix.conf file.

```
# repoquery --requires zabbix-apache-conf-scl
httpd
zabbix-web-deps-scl
```

zabbix-nginx-conf-scl

This package pulls Nginx from Red Hat Software Collections.

```
# repoquery --requires zabbix-nginx-conf-scl
rh-nginx116-nginx
zabbix-web
```

It also contains Zabbix configuration file for Nginx server at /etc/opt/rh/rh-nginx116/nginx/conf.d/zabbix.conf.

Using third party PHP repositories

If for some reasons Red Hat Software Collections can not be used, these alternative methods are available:

- Using any third-party repository that provides PHP.
- Building PHP from source.

PHP modules needed for Zabbix frontend are php-gd, php-bcmath, php-mbstring, php-xml, php-ldap and php-json.

Upgrading to Zabbix 5.0 from older versions

Special care must be taken during upgrade to Zabbix 5.0 from previous versions.

> **Attention:**
> See general upgrade instructions.

Packages from Red Hat Software Collections are designed to avoid conflicts with files from the main repositories.
Every such package is installed into a separate environment dedicated to its group.
For example, packages from rh-php72-php* group have their configuration installed under `/etc/opt/rh/rh-php72/` directory, logs are under `/var/opt/rh/rh-php72/log/` directory, etc. Services provided by these packages have unusual names such as `rh-php72-php-fpm` or `rh-nginx116-nginx`.

Official Zabbix 5.0 frontend packages use php-fpm with both Apache and Nginx.

Upgrade process with Apache

This section provides Apache-specific instructions for upgrading Zabbix frontend and server from version 4.0 or 4.4 to 5.0. For Nginx-specific instructions see Upgrade process with Nginx.

Instructions below are for installing Zabbix with MySQL support. Substitute 'mysql' in the commands with 'pgsql' to use PostgreSQL
It is assumed that both frontend and server run on the same box. If you have different setup, make adjustments accordingly.
Remove old frontend

Existing Zabbix frontend must be removed before starting an upgrade. Old configuration file will be moved to `/etc/httpd/conf.d/zabbix.` by rpm.

```
yum remove zabbix-web-*
```

Install SCL repository

On `RHEL` run

```
yum-config-manager --enable rhel-server-rhscl-7-rpms
```

On `CentOS` run

```
yum install centos-release-scl
```

On `Oracle Linux` run

```
yum install scl-utils
yum install oraclelinux-release-el7
/usr/bin/ol_yum_configure.sh
yum-config-manager --enable software_collections
yum-config-manager --enable ol7_latest ol7_optional_latest
```

Install Zabbix 5.0 release package and enable zabbix-frontend repository

Install `zabbix-release-5.0` package.

```
rpm -Uvh https://repo.zabbix.com/zabbix/5.0/rhel/7/x86_64/zabbix-release-5.0-1.el7.noarch.rpm
yum clean all
```

Edit `/etc/yum.repos.d/zabbix.repo` file. Replace `enabled=0` with `enabled=1`.

```
[zabbix-frontend]
...
enabled=1
...
```

Install new frontend packages

```
yum install zabbix-web-mysql-scl zabbix-apache-conf-scl
```

Official Zabbix 5.0 frontend packages use php-fpm. Update timezone in `/etc/opt/rh/rh-php72/php-fpm.d/zabbix.conf` file.

Update remaining packages and restart Zabbix server

```
yum update zabbix-*
```

> **Attention:**
> Restarting Zabbix server will upgrade the database. Make sure the database is backed up.

```
systemctl restart zabbix-server
```

Update remaining services

Start and enable php-fpm service.

```
systemctl start rh-php72-php-fpm
systemctl enable rh-php72-php-fpm
```

Restart Apache.

```
systemctl restart httpd
```

Upgrade process with Nginx

Follow upgrade procedure for apache, described above, but make some adjustments.
A few more steps have to be performed:

Make sure to stop and disable old Nginx and php-fpm before upgrading. To do this, run:

```
systemctl stop nginx php-fpm
systemctl disable nginx php-fpm
```

When editing `zabbix.conf` file for php-fpm, add user `nginx` to `listen.acl_users` directive

```
listen.acl_users = apache,nginx
```

Make sure `zabbix-nginx-conf-scl` package is installed instead of `zabbix-apache-conf-scl` package.

```
yum install zabbix-nginx-conf-scl
```

Edit `/opt/rh/rh-nginx116/nginx/conf.d/zabbix.conf` file.
Configure `listen` and `server_name` directives.

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

Start and enable Nginx and php-fpm

```
systemctl start rh-nginx116-nginx rh-php72-php-fpm
systemctl enable rh-nginx116-nginx rh-php72-php-fpm
```

Updating to PHP 7.3 packages

Overview

If you installed Zabbix frontend on RHEL/CentOS with support of PHP 7.2 software collections on RHEL 7, these steps will allow to upgrade your frontend with the support of PHP 7.3 packages.

Steps

1. Update to Zabbix 5.0.8 and check for new packages:

```
# yum search zabbix-web

...

zabbix-web-deps-scl.noarch : Convenience package for installing php dependencies of zabbix-web package fro
zabbix-web-deps-scl-php73.noarch : Convenience package for installing php dependencies of zabbix-web packa
zabbix-web.noarch : Zabbix web frontend common package
zabbix-web-japanese.noarch : Japanese font settings for Zabbix frontend
zabbix-web-mysql-scl.noarch : Zabbix web frontend for MySQL (scl version)
zabbix-web-mysql-scl-php73.noarch : Zabbix web frontend for MySQL (scl version)
zabbix-web-pgsql-scl.noarch : Zabbix web frontend for PostgreSQL (scl version)
zabbix-web-pgsql-scl-php73.noarch : Zabbix web frontend for PostgreSQL (scl version)
```

2. Install **zabbix-web-mysql-scl-php73** or **zabbix-web-pgsql-scl-php73** package

```
yum install zabbix-web-mysql-scl-php73
```

or

```
yum install zabbix-web-pgsql-scl-php73
```

3. Edit the /etc/opt/rh/rh-php73/php-fpm.d/zabbix.conf file

If you are using Nginx, add it to the listen.acl_users directive.

```
listen.acl_users = apache,nginx
```

Set your timezone.

```
; php_value[date.timezone] = Europe/Riga
```

4. Update the web server configuration

For Apache, edit the /etc/httpd/conf.d/zabbix.conf file. Configure the appropriate php-fpm socket.

```
#         SetHandler "proxy:unix:/var/opt/rh/rh-php72/run/php-fpm/zabbix.sock|fcgi://localhost"
        SetHandler "proxy:unix:/var/opt/rh/rh-php73/run/php fpm/zabbix.sock|fcgi://localhost"
```

Do a similar configuration in /etc/opt/rh/rh-nginx116/nginx/conf.d/zabbix.conf, if Nginx is used.

```
#         fastcgi_pass    unix:/var/opt/rh/rh-php72/run/php-fpm/zabbix.sock
        fastcgi_pass    unix:/var/opt/rh/rh-php73/run/php-fpm/zabbix.sock
```

5. Stop and disable the old rh-php72-php-fpm service

```
systemctl stop rh-php72-php-fpm
systemctl disable rh-php72-php-fpm
```

6. Start and enable the new rh-php73-php-fpm service

```
systemctl start rh-php73-php-fpm
systemctl enable rh-php73-php-fpm
```

Restart the web server.

```
systemctl restart httpd
```

or

```
systemctl restart rh-nginx116-nginx
```

Remove the old rh-php72 packages.

```
yum remove rh-php72*
```

2 Debian/Ubuntu frontend installation

Overview

Starting from version 5.0, Zabbix frontend requires PHP version 7.2 or later. Unfortunately, older versions of Debian & Ubuntu provide only PHP versions below 7.2.

Supported PHP versions by distribution

Distribution	PHP Version
Debian 10 (buster)	7.3
Debian 9 (stretch)	7.0
Debian 8 (jessie)	5.6
Ubuntu 20.04 (focal)	7.4
Ubuntu 18.04 (bionic)	7.2
Ubuntu 16.04 (xenial)	7.0
Ubuntu 14.04 (trusty)	5.5
Raspbian 10 (buster)	7.3
Raspbian 8 (stretch)	7.0

On *stretch, jessie, xenial and trusty* distributions, PHP 7.2 dependency is not available, and therefore Zabbix frontend 5.0 or newer cannot be easily installed. Considering this, `zabbix-frontend-php` package has been replaced with `zabbix-frontend-php-deprecated` package on aforementioned distributions.
The main difference is absence of direct dependencies on any php or web-server packages. Thus, the user can (and must) provide these dependencies on their own. In other words, installing `zabbix-frontend-php-deprecated` package on its own will not give you a working frontend. A web server as well as PHP 7.2 with its modules have to be installed manually (use PPAs / build PHP from source). We don't endorse any particular method.

Note:
The official way of getting PHP 7.2 or later on older versions of Debian/Ubuntu is to upgrade to buster/bionic.

PHP modules required for Zabbix frontend are `php-gd`, `php-bcmath`, `php-mbstring`, `php-xml`, `php-ldap` and `php-json`.

7 Upgrade procedure

Overview

This section provides upgrade information for Zabbix **5.0**:

- using packages:
  - for Red Hat Enterprise Linux/CentOS
  - for Debian/Ubuntu
- using sources

Direct upgrade to Zabbix 5.0.x is possible from Zabbix **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 **4.4**.x to Zabbix **5.0**.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 5.0 may take a long time.

Warning:
Before the upgrade make sure to read the relevant **upgrade notes!**

Warning:
RHEL/CentOS 7 users, be aware of PHP >= 7.2 version requirements when upgrading frontend!
Be sure to read relevant documentation!

The following upgrade notes are available:

Upgrade from	Read full upgrade notes	Most important changes between versions
4.4.x	For: Zabbix 5.0	Support of IBM DB2 dropped; Minimum required PHP version upped from 5.4.0 to 7.2.0; Minimum required database versions upped.
4.2.x	For: Zabbix 4.4 Zabbix 5.0	Jabber, Ez Texting media types removed.
4.0.x LTS	For: Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	Older proxies no longer can report data to an upgraded server; Newer agents no longer will be able to work with an older Zabbix server.
3.4.x	For: Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	'libpthread' and 'zlib' libraries now mandatory; Support for plain text protocol dropped and header is mandatory; Pre-1.4 version Zabbix agents are no longer supported; The Server parameter in passive proxy configuration now mandatory.
3.2.x	For: Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	SQLite support as backend database dropped for Zabbix server/frontend; Perl Compatible Regular Expressions (PCRE) supported instead of POSIX extended; 'libpcre' and 'libevent' libraries mandatory for Zabbix server; Exit code checks added for user parameters, remote commands and system.run[] items without the 'nowait' flag as well as Zabbix server executed scripts; Zabbix Java gateway has to be upgraded to support new functionality.
3.0.x LTS	For: Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	Database upgrade may be slow, depending on the history table size.

You may also want to check the requirements for 5.0.

Note:
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
```

Attention:
It is no longer possible to start the upgraded server and have older and unupgraded proxies report data to a newer server. This approach, which was never recommended nor supported by Zabbix, now is officially disabled when upgrading to 5.0 (or later) from any version before 4.4, 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

Before proceeding with the upgrade, update your current repository package to the latest version to ensure compatibility with the newest packages and to include any recent security patches or bug fixes.

On **RHEL/CentOS 8**, run:

```
# rpm -Uvh https://repo.zabbix.com/zabbix/5.0/rhel/8/x86_64/zabbix-release-latest.el8.noarch.rpm
```

On **RHEL/CentOS 7**, run:

```
# rpm -Uvh https://repo.zabbix.com/zabbix/5.0/rhel/7/x86_64/zabbix-release-5.0-1.el7.noarch.rpm
```

For RHEL/CentOS 9 or versions older than RHEL/CentOS 7, replace the link above with the correct one from Zabbix repository. Note, however, that packages for those versions may not include all Zabbix components. For a list of components included, see Zabbix packages.

Then, update the repository information:

```
# yum update
```

See also: Known issues for updating the repository configuration package on RHEL/CentOS.

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 following changes to the `/etc/php-fpm.d/zabbix.conf` file (uncomment and set the right timezone for you):

```
; php_value[date.timezone] = Europe/Riga
```

To upgrade the web frontend **on RHEL 7** follow instructions on this page (extra steps are required to install PHP 7.2 or newer). In particular, make sure to install `zabbix-apache-conf-scl` package if you use Apache web server.

```
 # yum install zabbix-apache-conf-scl
```

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 5.0.x (for example, from 5.0.1 to 5.0.3). 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 **4.4**.x to Zabbix **5.0**.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 5.0 may take a long time.

Warning:
Before the upgrade make sure to read the relevant **upgrade notes!**

Warning:
Users of Ubuntu **16.04 or older** and **Debian 9 or older** beware of the lack of official PHP 7.2 packages on your distributions.
Be sure to read relevant documentation!

The following upgrade notes are available:

Upgrade from	Read full upgrade notes	Most important changes between versions
4.4.x	For: Zabbix 5.0	Support of IBM DB2 dropped; Minimum required PHP version upped from 5.4.0 to 7.2.0; Minimum required database versions upped.
4.2.x	For: Zabbix 4.4 Zabbix 5.0	Jabber, Ez Texting media types removed.
4.0.x LTS	For: Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	Older proxies no longer can report data to an upgraded server; Newer agents no longer will be able to work with an older Zabbix server.

Upgrade from	Read full upgrade notes	Most important changes between versions
3.4.x	For: Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	'libpthread' and 'zlib' libraries now mandatory; Support for plain text protocol dropped and header is mandatory; Pre-1.4 version Zabbix agents are no longer supported; The Server parameter in passive proxy configuration now mandatory.
3.2.x	For: Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	SQLite support as backend database dropped for Zabbix server/frontend; Perl Compatible Regular Expressions (PCRE) supported instead of POSIX extended; 'libpcre' and 'libevent' libraries mandatory for Zabbix server; Exit code checks added for user parameters, remote commands and system.run[] items without the 'nowait' flag as well as Zabbix server executed scripts; Zabbix Java gateway has to be upgraded to support new functionality.
3.0.x LTS	For: Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	Database upgrade may be slow, depending on the history table size.

You may also want to check the requirements for 5.0.

Note:
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 Zabbix proxy, stop proxy too.

```
# systemctl stop zabbix-proxy
```

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

Before proceeding with the upgrade, uninstall your current repository package:

```
# rm -Rf /etc/apt/sources.list.d/zabbix.list
```

Then, install the latest repository configuration package to ensure compatibility with the newest packages and to include any recent security patches or bug fixes.

On **Debian 11**, run:

```
# wget https://repo.zabbix.com/zabbix/5.0/debian/pool/main/z/zabbix-release/zabbix-release_5.0-2+debian11_
# dpkg -i zabbix-release_5.0-2+debian11_all.deb
```

On **Debian 10**, run:

```
# wget https://repo.zabbix.com/zabbix/5.0/debian/pool/main/z/zabbix-release/zabbix-release_5.0-1+buster_al
# dpkg -i zabbix-release_5.0-1+buster_all.deb
```

For older Debian versions, replace the link above with the correct one from Zabbix repository. Note, however, that packages for those versions may not include all Zabbix components. For a list of components included, see Zabbix packages.

On **Ubuntu 20.04**, run:

```
# wget https://repo.zabbix.com/zabbix/5.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_5.0-1+focal_all
# dpkg -i zabbix-release_5.0-1+focal_all.deb
```

On **Ubuntu 18.04**, run:

```
# wget https://repo.zabbix.com/zabbix/5.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_5.0-1+bionic_al
# dpkg -i zabbix-release_5.0-1+bionic_all.deb
```

For Ubuntu 24.04, Ubuntu 22.04, and versions older than Ubuntu 18.04, replace the link above with the correct one from Zabbix repository. Note, however, that packages for those versions may not include all Zabbix components. For a list of components included, see Zabbix packages.

Then, 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 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 this page for information about installing Zabbix frontend on older distributions.

6 Review component configuration parameters

See the upgrade notes for details on mandatory changes.

For new optional parameters, see the What's new section.

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 minor versions of 5.0.x (for example, from 5.0.1 to 5.0.3). 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 **4.4**.x to Zabbix **5.0**.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.

> **Attention:**
> It is no longer possible to start the upgraded server and have older and unupgraded proxies report data to a newer server. This approach, which was never recommended nor supported by Zabbix, now is officially disabled when upgrading to 5.0 (or later) from any version before 5.0, 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 5.0 may take a long time.

> **Warning:**
> Before the upgrade make sure to read the relevant **upgrade notes!**

The following upgrade notes are available:

Upgrade from	Read full upgrade notes	Most important changes between versions
4.4.x	For: Zabbix 5.0	Support of IBM DB2 dropped; Minimum required PHP version upped from 5.4.0 to 7.2.0; Minimum required database versions upped; Changed Zabbix PHP file directory.
4.2.x	For: Zabbix 4.4 Zabbix 5.0	Jabber, Ez Texting media types removed.
4.0.x LTS	For: Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	Older proxies no longer can report data to an upgraded server; Newer agents no longer will be able to work with an older Zabbix server.
3.4.x	For: Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	'libpthread' and 'zlib' libraries now mandatory; Support for plain text protocol dropped and header is mandatory; Pre-1.4 version Zabbix agents are no longer supported; The Server parameter in passive proxy configuration now mandatory.

Upgrade from	Read full upgrade notes	Most important changes between versions
3.2.x	For: Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	SQLite support as backend database dropped for Zabbix server/frontend; Perl Compatible Regular Expressions (PCRE) supported instead of POSIX extended; 'libpcre' and 'libevent' libraries mandatory for Zabbix server; Exit code checks added for user parameters, remote commands and system.run[] items without the 'nowait' flag as well as Zabbix server executed scripts; Zabbix Java gateway has to be upgraded to support new functionality.
3.0.x LTS	For: Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	Database upgrade may be slow, depending on the history table size.
2.4.x	For: Zabbix 3.0 Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	Minimum required PHP version upped from 5.3.0 to 5.4.0 LogFile agent parameter must be specified
2.2.x LTS	For: Zabbix 2.4 Zabbix 3.0 Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	Node-based distributed monitoring removed
2.0.x	For: Zabbix 2.2 Zabbix 2.4 Zabbix 3.0 Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0	Minimum required PHP version upped from 5.1.6 to 5.3.0; 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'

You may also want to check the requirements for 5.0.

Note:
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 current database version (mandatory/optional): 03040000/03040000
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 7.2.0. 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. For new optional parameters, see the What's new section.

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

> **Attention:**
> 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 5.0.x (for example from 5.0.1 to 5.0.3) 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

See also: Compilation 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';
```

Installation from packages

It has been observed that it is impossible to install a specific frontend version by running, e.g.:

```
yum install -v zabbix-web-mysql-scl-5.0.0
```

As a workaround to this issue, if you wish to install a specific frontend version, specify version for all components, e.g.:

```
yum install zabbix-web-mysql-scl-5.0.0 zabbix-apache-conf-scl-5.0.0 zabbix-web-5.0.0 zabbix-web-deps-scl-5
```

Zabbix packages for RHEL on Red Hat UBI environments

When installing Zabbix from Red Hat Enterprise Linux packages on Red Hat Universal Base Image environments, ensure access to required repositories and dependencies. Zabbix packages depend on `libOpenIPMI.so` and `libOpenIPMIposix.so` libraries, which are not provided by any package in the default package manager repositories enabled on UBI systems and will result in installation failures.

The `libOpenIPMI.so` and `libOpenIPMIposix.so` libraries are available in the `OpenIPMI-libs` package, which is provided by the `redhat-#-for-<arch>-appstream-rpms` repository. Access to this repository is curated by subscriptions, which, in the case of UBI environments, get propagated by mounting repository configuration and secrets directories of the RHEL host into the container file-system namespace.

For more information, see ZBX-24291.

Expired signing key for RHEL/CentOS packages

When upgrading Zabbix on Red Hat Enterprise Linux/CentOS, you may encounter an expired signing key issue for packages on Zabbix repository. When a signing key expires, attempts to verify package signatures will result in an error indicating that the certificate or key is no longer valid. For example:

```
error: Verifying a signature using certificate D9AA84C2B617479C6E4FCF4D19F2475308EFA7DD (Zabbix LLC (Jul 2
  1. Certificate 19F2475308EFA7DD invalid: certificate is not alive
      because: The primary key is not live
      because: Expired on 2024-07-04T11:41:23Z
  2. Key 19F2475308EFA7DD invalid: key is not alive
      because: The primary key is not live
      because: Expired on 2024-07-04T11:41:23Z
```

To resolve such issues, manually reinstall the latest `zabbix-release` package for your specific variant of RHEL/CentOS (replace the link below with the correct one from Zabbix repository).

For example, on **RHEL/CentOS 8**, run:

```
rpm -Uvh https://repo.zabbix.com/zabbix/5.0/rhel/8/x86_64/zabbix-release-latest.el8.noarch.rpm
```

Then, update the repository information:

```
yum update
```

For more information, see ZBX-24761.

Database connection encryption

Database connection encryption is not supported on RHEL/CentOS 7, because these systems ship with packages that are too old:

```
Name        : mariadb-libs
Version     : 5.5.65

Name        : postgresql-libs
Version     : 9.2.24
```

Timescale DB: high memory usage with 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 is used. It has been resolved since TimescaleDB 2.5.1.

For more information, please see TimescaleDB Issue #3773.

Upgrade with MySQL below 5.7.0 or MariaDB below 10.2.2

Upgrading Zabbix may fail if database tables were created with older DB versions (MySQL versions prior to 5.7 or MariaDB versions prior to 10.2.2), 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 (MariaDB) and ZBX-20165 (MySQL).

Database TLS connection with MariaDB

Database TLS connection is not supported with the 'verify_ca' option for the DBTLSConnect parameter if MariaDB is used.

Possible deadlocks with MySQL/MariaDB

When running under high load, and with more than one LLD worker involved, it is possible to run into a deadlock caused by an InnoDB error related to the row-locking strategy (see upstream bug). The error has been fixed in MySQL since 8.0.29, but not in MariaDB. For more details, see ZBX-21506.

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 -1.34E-154 to 1.34E+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.
- Using the pipe "|" in the SSH script may lead to a "Cannot read data from SSH server" error. In this case it is recommended to upgrade the libssh library version. See also ZBX-21337 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

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.
- If using FreeTDS UnixODBC driver, you need to prepend a 'SET NOCOUNT ON' statement to an SQL query (for example, `SET NOCOUNT ON DECLARE @strsql NVARCHAR(max) SET @strsql = ....`). Otherwise, database monitor item in Zabbix will fail to retrieve the information with an error "SQL query returned empty result".
  See ZBX-19917 for more information.

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 `icmpping`, `icmppingloss`, `icmppingsec` items. It is recommended to use the latest version of **fping**. Please see ZBX-11726 for more details.

Errors with fping execution in rootless containers

When containers are running in rootless mode or in a specific-restrictions environment, you may face errors related to fping execution when performing ICMP checks, such as `fping: Operation not permitted` or all packets to all resources lost.

To fix this problem add `--cap-add=net_raw` to "docker run" or "podman run" commands.

Additionally fping execution in non-root environments may require sysctl modification, i.e.:

```
sudo sysctl -w "net.ipv4.ping_group_range=0 1995"
```

where "1995" is the zabbix GID. For more details, see ZBX-22833.

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.

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

Daylight Saving Time

Changes to Daylight Saving Time (DST) result in irregularities when displaying X axis labels (date duplication, date missing, etc.).

Sum aggregation

When using sum aggregation in a graph for period that is less than one hour, graphs display incorrect (multiplied) values when data come from trends.

Text overlapping

For some frontend languages (e.g., Japanese), local fonts can cause text overlapping in graph legend. To avoid this, use version 2.3.0 (or later) of PHP GD extension.

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 queries

Slow database queries have been observed in the cases listed below.

Permission checks for dashboards with graphs

If SVG graphs are used in Zabbix dashboards, Zabbix frontend may generate non-optimized API queries when checking user permissions to hosts and items. This is happening because such searches support wildcards and case-insensitive name matching.

To improve performance of SQL statements, manually apply the patch *index_host_and_item_name_upper_field.sql* provided in Zabbix 5.0.31 and newer.

For MySQL, run:

```
shell> mysql -uzabbix -p<password> zabbix < index_host_and_item_name_upper_field.sql
```

For PostgreSQL, run:

```
shell> cat index_host_and_item_name_upper_field.sql | sudo -u zabbix psql zabbix
```

For Oracle, run:

```
sqlplus> @index_host_and_item_name_upper_field.sql
```

Creation of deterministic triggers should be enabled for the time of patch application. 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 patch has been successfully applied, `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.

Non-existing item value retrieval with MySQL 5.6/5.7

Zabbix server generates slow `SELECT` queries in case of non-existing values for items. This issue is known to occur in MySQL 5.6/5.7 versions (for an extended discussion, see ZBX-10652), and, in specific cases, may also occur in later MySQL versions. A workaround to this is disabling the `index_condition_pushdown` or `prefer_ordering_index` optimizer in MySQL. Note, however, that this workaround may not fix all issues related to slow queries.

Event info retrieval with MySQL 5.X and 8.0.19

Zabbix 5.0 installations with MySQL 5.X and 8.0.19 might run a slow query when retrieving problem/event information from the database. In particular, this affects the *Problems by severity* widget, `event.get` and `problem.get` API methods. To improve performance of SQL statements, apply the patch provided in ZBX-18080 (available for Zabbix 5.0.4 and newer).

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`.

Persistent filter settings from links

When opening a link to Zabbix frontend page that contains filter settings, including the time selector, the filter is automatically saved in the database for the user, replacing the previously saved filter and/or time selector settings for that page. These settings remain active until the user manually updates or resets them.

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

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.

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.

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.

Use case with global variables shared across webhook calls

As global variables are shared across different webhook calls, the following code will result in the tag value counter gradually increasing:

```
try
{
   aa = aa + 1;
}
catch(e)
{
   aa = 0;
}

result = {
        'tags': {
            'endpoint': aa
        }
    };
return JSON.stringify(result);
```

Using local variables instead of global ones is recommended to make sure that each script operates on its own data and that there are no collisions between simultaneous calls.

Incorrect information from nested host groups in maps

Information from nested host groups is incorrectly displayed in maps, for example:

- Host group label displays the problem summary not including all hosts in nested host groups;
- ”Host group elements” view does not display a separate map element for each host in the nested host groups;
- Map label displays summary of all problems not including those in nested host groups.

1 Compilation issues

These are the known issues regarding Zabbix compilation from sources. For all other cases, see the Known issues page.

Compiling Zabbix agent on HP-UX

If you install the PCRE library from the popular HP-UX package site http://hpux.connect.org.uk (for example, from file `pcre-8.42-ia64_64-11.31.depot`), only the 64-bit version of the library will be installed in the `/usr/local/lib/hpux64` directory.

In this case, for successful agent compilation, a customized option is needed for the `configure` script, for example:

```
CFLAGS="+DD64" ./configure --enable-agent --with-libpcre-include=/usr/local/include --with-libpcre-lib=/us
```

Library in a non-standard location

Zabbix allows you to specify a library located in a non-standard location. In the example below, Zabbix will run `curl-config` from the specified non-standard location and use its output to determine the correct libcurl to use.

```
$ ./configure --enable-server --with-mysql --with-libcurl=/usr/local/bin/curl-config
```

This will work if it is the only libcurl installed in the system, but might not if there is another libcurl installed in a standard location (by the package manager, for example). Such is the case when you need a newer version of the library for Zabbix and the older one for other applications.

Therefore, specifying a component in a non-standard location will not always work when the same component also exists in a standard location.

For example, if you use a newer libcurl installed in `/usr/local` with the libcurl package still installed, Zabbix might pick up the wrong one and compilation will fail:

```
usr/bin/ld: ../../src/libs/zbxhttp/libzbxhttp.a(http.o): in function 'zbx_http_convert_to_utf8':
/tmp/zabbix-master/src/libs/zbxhttp/http.c:957: undefined reference to 'curl_easy_header'
collect2: error: ld returned 1 exit status
```

Here, the function `curl_easy_header()` is not available in the older `/usr/lib/x86_64-linux-gnu/libcurl.so`, but is available in the newer `/usr/local/lib/libcurl.so`.

The problem lies with the order of linker flags, and one solution is to specify the full path to the library in an LDFLAGS variable:

```
$ LDFLAGS="-Wl,--no-as-needed /usr/local/lib/libcurl.so" ./configure --enable-server --with-mysql --with-l
```

Note the `-Wl,--no-as-needed` option which might be needed on some systems (see also: default linking options on Debian-based systems).

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).

New templates

See the list of new templates in Zabbix 5.0.0

Changes in 5.0.1

New PostgreSQL template is available:

- *Template DB PostgreSQL Agent 2* - collects metrics from PostgreSQL with Zabbix agent 2.

Changes in 5.0.2

New templates are available:

- *Template App Etcd by HTTP* - collects metrics from Etcd's `/metrics` endpoint with HTTP agent (see description).
- *Template DB MSSQL by ODBC* - collects metrics from DBMS Microsoft SQL Server via ODBC (see description).
- *Template App IIS by Zabbix agent*, *Template App IIS by Zabbix agent active* - collect metrics from Internet Information Services for Windows Server version 2012R2 and newer via Zabbix agent.

Since Zabbix 5.0 SNMP credentials are carried at the host interface level instead of item level, therefore SNMP templates are now valid for all devices, no matter the protocol version. To reflect this change, the following templates have been replaced:

- *Template Module Brocade_Foundry Performance SNMPv2* → *Template Module Brocade_Foundry Performance SNMP*
- *Template Module Cisco CISCO-MEMORY-POOL-MIB SNMPv2* → *Template Module Cisco CISCO-MEMORY-POOL-MIB SNMP*
- *Template Module EtherLike-MIB SNMPv1*, *Template Module EtherLike-MIB SNMPv2* → *Template Module EtherLike-MIB SNMP*
- *Template Module Generic SNMPv1*, *Template Module Generic SNMPv2* → *Template Module Generic SNMP*
- *Template Module HOST-RESOURCES-MIB storage SNMPv1*, *Template Module HOST-RESOURCES-MIB storage SNMPv2* → *Template Module HOST-RESOURCES-MIB storage SNMP*
- *Template Module Interfaces SNMPv1*, *Template Module Interfaces SNMPv2* → *Template Module Interfaces SNMP*
- *Template Module Interfaces Simple SNMPv1*, *Template Module Interfaces Simple SNMPv2* → *Template Module Interfaces Simple SNMP*
- *Template Module Interfaces Windows SNMPv2* → *Template Module Interfaces Windows SNMP*
- *Template Module Linux memory SNMPv2* → *Template Module Linux memory SNMP*
- *Template Net Alcatel Timetra TiMOS SNMPv2* → *Template Net Alcatel Timetra TiMOS SNMP*
- *Template Net Arista SNMPv2* → *Template Net Arista SNMP*
- *Template Net Brocade FC SNMPv2* → *Template Net Brocade FC SNMP*
- *Template Net D-Link DES 7200 SNMPv2* → *Template Net D-Link DES 7200 SNMP*
- *Template Net D-Link DES_DGS Switch SNMPv2* → *Template Net D-Link DES_DGS Switch SNMP*
- *Template Net Dell Force S-Series SNMPv2* → *Template Net Dell Force S-Series SNMP*
- *Template Net Extreme EXOS SNMPv2* → *Template Net Extreme EXOS SNMP*
- *Template Net HP Comware HH3C SNMPv2* → *Template Net HP Comware HH3C SNMP*
- *Template Net HP Enterprise Switch SNMPv2* → *Template Net HP Enterprise Switch SNMP*
- *Template Net Huawei VRP SNMPv2* → *Template Net Huawei VRP SNMP*
- *Template Net Intel_Qlogic Infiniband SNMPv2* → *Template Net Intel_Qlogic Infiniband SNMP*
- *Template Net Juniper SNMPv2* → *Template Net Juniper SNMP*
- *Template Net Mellanox SNMPv2* → *Template Net Mellanox SNMP*

- *Template Net Mikrotik SNMPv2* → *Template Net Mikrotik SNMP*
- *Template Net Netgear Fastpath SNMPv2* → *Template Net Netgear Fastpath SNMP*
- *Template Net Network Generic Device SNMPv1*, *Template Net Network Generic Device SNMPv2* → *Template Net Network Generic Device SNMP*
- *Template Net QTech QSW SNMPv2* → *Template Net QTech QSW SNMP*
- *Template Net TP-LINK SNMPv2* → *Template Net TP-LINK SNMP*
- *Template Net Ubiquiti AirOS SNMPv1* → *Template Net Ubiquiti AirOS SNMP*
- *Template OS Windows SNMPv2* → *Template OS Windows SNMP*
- *Template Server Cisco UCS SNMPv2* → *Template Server Cisco UCS SNMP*
- *Template Server Dell iDRAC SNMPv2* → *Template Server Dell iDRAC SNMP*
- *Template Server HP iLO SNMPv2* → *Template Server HP iLO SNMP*
- *Template Server IBM IMM SNMPv1*, *Template Server IBM IMM SNMPv2* → *Template Server IBM IMM SNMP*
- *Template Server Supermicro Aten SNMPv2* → *Template Server Supermicro Aten SNMP*

Changes in 5.0.3

A new Oracle DB template is available: *Template DB Oracle by ODBC* - see setup instructions for ODBC templates.

Template VM VMware has been updated:

- the type of information for the "VMware: Free space on datastore (percentage)" item has been corrected and is now set to Numeric (float).
- The following template macros now have more specific names:
  - {$URL} renamed to {$VMWARE.URL}
  - {$USERNAME} renamed to {$VMWARE.USERNAME}
  - {$PASSWORD} renamed to {$VMWARE.PASSWORD}

Changes in 5.0.4

A new Oracle DB template is available: *Template DB Oracle by Zabbix agent 2* - see setup instructions for Zabbix agent 2 templates.

Changes in 5.0.5

New templates are available:

- *Template App Ceph by Zabbix agent 2* - see setup instructions for Zabbix agent 2 templates.
- *Template App PHP-FPM by Zabbix agent* - see setup instructions for Zabbix agent templates.
- *Template App PHP-FPM by HTTP* - see setup instructions for HTTP templates.
- *Template App Squid SNMP* - see description.
- *Template Tel Asterisk by HTTP* - see setup instructions for HTTP templates.

Changes in 5.0.6

Template *Apache Tomcat by JMX* has been updated. Now it is possible to use the template to automatically discover and monitor Apache Tomcat hosts with any configuration settings. See also: JMX template operation.

New templates are available:

- *Apache Cassandra by JMX* - see setup instructions for JMX templates;
- *Apache Kafka by JMX* - - see setup instructions for JMX templates;
- *Hadoop by HTTP* - see setup instructions for HTTP templates;
- *Morningstar ProStar MPPT SNMP* - monitoring of Prostar MPPT solar charge controller via SNMP;
- *Morningstar ProStar PWM SNMP* - monitoring of ProStar pulse width modulation (PWM) solar charge controller via SNMP;
- *Morningstar SunSaver MPPT SNMP* - monitoring of SunSaver MPPT solar charge controller via SNMP;
- *Morningstar SureSine SNMP* - monitoring of SureSine pure sine wave inverter via SNMP;
- *Morningstar TriStar MPPT 600V SNMP* - monitoring of TriStar MPPT 600V solar charge controller via SNMP;
- *Morningstar TriStar MPPT SNMP* - monitoring of TriStar MPPT solar charge controller via SNMP;
- *Morningstar TriStar PWM SNMP* - monitoring of TriStar PWM solar charge controller via SNMP;
- *ZooKeeper by HTTP* - see setup instructions for HTTP templates.

Changes in 5.0.8

New templates are available:

- *Apache ActiveMQ by JMX* - see setup instructions for JMX templates;
- *Microsoft Exchange Server 2016 by Zabbix agent*, *Microsoft Exchange Server 2016 by Zabbix agent active* - see setup instructions for Zabbix agent templates;
- *NetApp FAS3220 SNMP* - monitoring of NetApp FAS3220 via SNMP.

Changes in 5.0.9

A new template is available:

- *Ignite by JMX* - see setup instructions for JMX templates.

Changes in 5.0.10

New templates are available:

- *APC UPS SNMP* - monitoring of APC UPS Symmetra LX by Zabbix SNMP agent;
- *Cisco Catalyst 3750V2-24FS SNMP* - monitoring of Cisco Catalyst 3750V2-24FS switch via SNMP;
- *Cisco Catalyst 3750V2-24PS SNMP* - monitoring of Cisco Catalyst 3750V2-24PS switch via SNMP;
- *Cisco Catalyst 3750V2-24TS SNMP* - monitoring of Cisco Catalyst 3750V2-24TS switch via SNMP;
- *Cisco Catalyst 3750V2-48TS SNMP* - monitoring of Cisco Catalyst 3750V2-48TS switch via SNMP;
- *Cisco Catalyst 3750V2-48TS SNMP* - monitoring of Cisco Catalyst 3750V2-48TS switch via SNMP;
- *Huawei OceanStor 5300 V5 SNMP* - monitoring of SAN Huawei OceanStor 5300 V5 via SNMP;
- *MongoDB cluster by Zabbix agent 2* - see setup instructions for Zabbix agent 2 templates;
- *MongoDB node by Zabbix agent 2* - see setup instructions for Zabbix agent 2 templates;
- *SMART by Zabbix agent 2* - see setup instructions for Zabbix agent 2 templates;
- *SMART by Zabbix agent 2 active* - see setup instructions for Zabbix agent 2 templates;
- *Template SAN NetApp AFF A700 by HTTP* - see setup instructions for HTTP templates.

Changes in 5.0.11

New templates are available:

- *APC UPS Galaxy 3500 SNMP* - monitoring of APC UPS Galaxy 3500 by Zabbix SNMP agent LX by Zabbix SNMP agent;
- *APC Smart-UPS 2200 RM SNMP* - monitoring of APC Smart-UPS 2200 RM by Zabbix SNMP agent;
- *APC Smart-UPS 3000 XLM SNMP* - monitoring of APC Smart-UPS 3000 XLM by Zabbix SNMP agent;
- *APC Smart-UPS RT 1000 RM XL SNMP* - monitoring of APC Smart-UPS RT 1000 RM XL by Zabbix SNMP agent;
- *APC Smart-UPS RT 1000 XL SNMP* - monitoring of APC Smart-UPS RT 1000 XL by Zabbix SNMP agent;
- *APC Smart-UPS SRT 5000 SNMP* - monitoring of APC Smart-UPS SRT 5000 by Zabbix SNMP agent;
- *APC Smart-UPS SRT 8000 SNMP* - monitoring of APC Smart-UPS SRT 8000 by Zabbix SNMP agent;
- *APC UPS SNMP* - monitoring of APC UPS with NMC by Zabbix SNMP agent;
- *APC UPS Symmetra RM SNMP* - monitoring of APC UPS Symmetra RM by Zabbix SNMP agent;
- *APC UPS Symmetra RX SNMP* - monitoring of APC UPS Symmetra RX by Zabbix SNMP agent.

The template *APC UPS SNMP* introduced in Zabbix 5.0.10 for monitoring APC UPS Symmetra LX has been renamed to *APC UPS Symmetra LX SNMP* and updated with new metrics, triggers, a discovery rule, etc., which allow to monitor battery capacity and phase output voltage and load.

Changes in 5.0.12

New templates are available:

- *WildFly Domain by JMX* - see setup instructions for JMX templates;
- *WildFly Server by JMX* - see setup instructions for JMX templates.

Changes in 5.0.13

New templates are available:

- *AAM1212-51 IES-612 SNMP* - monitoring of Zyxel AAM1212-51 / IES-612 via SNMP;
- *Cisco UCS Manager SNMP* - monitoring of Cisco UCS Manager via SNMP;
- *DELL PowerEdge R720 by HTTP* - monitoring of DELL PowerEdge R720 servers with iDRAC 8/9 firmware 4.32 and later with Redfish API enabled via HTTP (see setup instructions);
- // DELL PowerEdge R720 SNMP// - monitoring of DELL PowerEdge R720 servers with iDRAC version 7 and later via SNMP;
- *DELL PowerEdge R740 by HTTP* - monitoring of DELL PowerEdge R740 servers with iDRAC 8/9 firmware 4.32 and later with Redfish API enabled via HTTP (see setup instructions);
- *DELL PowerEdge R740 SNMP* - monitoring of DELL PowerEdge R740 servers with iDRAC version 7 and later via SNMP;
- *DELL PowerEdge R820 by HTTP* - monitoring of DELL PowerEdge R820 servers with iDRAC 8/9 firmware 4.32 and later with Redfish API enabled via HTTP (see setup instructions);
- *DELL PowerEdge R820 SNMP* - monitoring of DELL PowerEdge R820 servers with iDRAC version 7 and later via SNMP;
- *DELL PowerEdge R840 by HTTP* - monitoring of DELL PowerEdge R840 servers with iDRAC 8/9 firmware 4.32 and later with Redfish API enabled via HTTP (see setup instructions);
- *DELL PowerEdge R840 SNMP* - monitoring of DELL PowerEdge R840 servers with iDRAC version 7 and later via SNMP;
- *ES3500-8PD SNMP* - monitoring of Zyxel ES3500-8PD via SNMP;
- *GS-4012F SNMP* - monitoring of Zyxel GS-4012F via SNMP;
- *HPE ProLiant BL460 SNMP* - monitoring of HPE ProLiant BL460 servers with HP iLO version 4 and later via SNMP;
- *HPE ProLiant BL920 SNMP* - monitoring of HPE ProLiant BL920 servers with HP iLO version 4 and later via SNMP;
- *HPE ProLiant DL360 SNMP* - monitoring of HPE ProLiant DL360 servers with HP iLO version 4 and later via SNMP;
- *HPE ProLiant DL380 SNMP* - monitoring of HPE ProLiant DL380 servers with HP iLO version 4 and later via SNMP;

- *IES-500x SNMP* - monitoring of Zyxel IES-500x via SNMP;
- *IES1248-51 SNMP* - monitoring of Zyxel IES1248-51 via SNMP;
- *MES-3528 SNMP* - monitoring of Zyxel MES-3528 via SNMP;
- *MES3500-10 SNMP* - monitoring of Zyxel MES3500-10 via SNMP;
- *MES3500-24 SNMP* - monitoring of Zyxel MES3500-24 via SNMP;
- *MGS-3712 SNMP* - monitoring of Zyxel MGS-3712 via SNMP;
- *MGS-3712F SNMP* - monitoring of Zyxel MGS-3712F via SNMP;
- *MES3500-24S SNMP* - monitoring of Zyxel MES3500-24S via SNMP;
- *MGS3520-28x SNMP* - monitoring of Zyxel MGS3520-28x via SNMP;
- *NGINX Plus by HTTP* - see setup instructions for HTTP templates;
- *XGS-4728F SNMP* - monitoring of Zyxel XGS-4728F via SNMP.

Zabbix server and *Remote Zabbix server* templates have been updated according to the latest template guidelines.

Changes in 5.0.14

New templates are available:

- *Big-IP SNMP* - monitoring of BIG-IP application services;
- *GridGain by JMX* is available - see setup instructions for JMX templates;
- *Systemd by Zabbix agent 2* - see setup instructions for Zabbix agent 2 templates.

Templates *Oracle by ODBC* and *Asterisk by HTTP* have been removed and will no longer be available in Zabbix 5.0 due to compatibility issues. Note that these templates are still available in Zabbix 5.2 and newer.

Changes in 5.0.15

New templates are available:

- *Cisco ASAv SNMP* - monitoring of Cisco Adaptive Security Virtual Appliance (ASAv) via SNMP;
- *Website certificate by Zabbix agent 2* - monitoring of TLS/SSL website certificates by Zabbix agent 2.

Changes in 5.0.20

A new template *pfSense SNMP* is available.

Changes in 5.0.22

The following templates have been updated:

- The template *Generic Java JMX* now contains two discovery rules: Garbage collector discovery and Memory pool discovery
- The templates *Template Module Windows services by Zabbix agent* and *Template Module Windows services by Zabbix agent active* now contain a new macro {$SERVICE.NAME.NOT_MATCHES.EXTENDED}, which is used to filter out additional services.
- The template *PostgreSQL by Zabbix agent 2* now will check the number of slow queries and generate a problem if the amount exceeds a threshold.

Changes in 5.0.23

The templates *SMART by Zabbix agent 2* and *SMART by Zabbix agent 2 (active)* have been updated: - the *Attribute discovery* LLD rule has been deleted, whereas the *Disk discovery* LLD rule will now discover disks based on the pre-defined vendor-specific set of attributes; - **smart.disk.get** item can now return information about a specific disk only, instead of all disks.

New macros allowing to define warning and critical thresholds of the filesystem utilization for virtual file system monitoring have been added to the templates *Template App PFSense SNMP*, *Template Module HOST-RESOURCES-MIB storage SNMP*, *Template Module Linux filesystems SNMP*, *Template Module Linux filesystems by Zabbix agent active*, *Template Module Linux filesystems by Zabbix agent*, *Template Module Windows filesystems by Zabbix agent active*, *Template Module Windows filesystems by Zabbix agent*, *Template Net Mellanox SNMP*, *Template OS Linux by Prom*. Filesystem utilization triggers have been updated to use these macros.

Changes in 5.0.26

PostgreSQL Agent 2 template updated.

A trigger for detecting checksum failures has been added to the Dbstat item of the PostgreSQL Agent 2 template. According to PostgreSQL documentation, you can use checksums on data pages to help detect corruption by the I/O system that would otherwise be silent.

Changes in 5.0.27

A new template *Template App OPNsense SNMP* is now available.

Changes in 5.0.31

Template DB Oracle by Zabbix agent 2 has been updated:

- The following static items, that requested data for all existing relevant DB objects in a single query, have been removed:
  - "Oracle: Get archive log info"
  - "Oracle: Get ASM stats"
  - "Oracle: Get CDB and No-CDB info"
  - "Oracle: Get PDB info"
  - "Oracle: Get tablespaces stats"
- The following agent item prototypes have been added to the corresponding discovery rules:
  - Archive log discovery rule: "Archivelog '{#DEST_NAME}': Get archive log info"
  - ASM disk groups discovery: "ASM '{#DGNAME}': Get ASM stats"
  - Database discovery: "Oracle Database '{#DBNAME}': Get CDB and No-CDB info"
  - PDB discovery: "Oracle Database '{#DBNAME}': Get PDB info"
  - Tablespace discovery: "Oracle TBS '{#TABLESPACE}': Get tablespace stats"

10 Upgrade notes for 5.0.0

These notes are for upgrading from Zabbix 4.4.x to Zabbix 5.0.0. 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 5.0.0 from versions before Zabbix 4.4.0. See the upgrade procedure section for all relevant information about upgrading from previous Zabbix versions.

Critical Minimum required PHP version

The minimum required PHP version has been upped to **7.2.0** from 5.4.0.

This change also affects the ability to install Zabbix frontend from packages in some distributions. See detailed instructions for installing Zabbix frontend from packages on RHEL/CentOS 7 and the affected Debian/Ubuntu versions.

Support of IBM DB2 dropped

The IBM DB2 database can no longer be used as a back-end database for Zabbix.

Support of Internet Explorer 11 dropped

Microsoft Internet Explorer 11 is no longer supported by Zabbix.

Support of mbedTLS (PolarSSL) crypto library dropped

mbedTLS (PolarSSL) crypto library is no longer supported by Zabbix. Supported crypto libraries are GnuTLS and OpenSSL.

Minimum required database versions

Minimum database versions required for Zabbix 5.0.0 have been upped to:

- MySQL 5.7 (For older MySQL versions DB upgrade may fail due to the InnoDB key prefix limit set at 767 bytes. The issue is fixed by increasing the limit to 3072 bytes in MySQL 5.7 and newer.)
- MariaDB 10.0.37
- PostgreSQL 9.2.24
- Oracle 11.2

Upgrade with MySQL below 5.7.0 or MariaDB below 10.2.2

Upgrading Zabbix may fail if database tables were created with older DB versions (MySQL versions prior to 5.7 or MariaDB versions prior to 10.2.2), 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 (MariaDB) and ZBX-20165 (MySQL).

Enabling extended range of numeric (float) values

Numeric (float) data type now supports precision of approximately 15 digits and range from approximately -1.79E+308 to 1.79E+308 (with exception of PostgreSQL 11 and earlier versions). This is by default for new installations. However, when upgrading existing installations, a manual database upgrade patch must be applied.

If you do not apply the patch, System information in the frontend will display: "Database history tables upgraded: No".

Attention:
The patch will alter data columns of history and trends tables, which usually contain lots of data, therefore it is expected to take some time to complete. Since the exact estimate depends on server performance, database management system configuration and version, and it cannot be predicted, it is recommended to first test the patch outside the production environment, even though with MySQL 8.0 and MariaDB 10.5 configured by default the patch is known to be executed instantly for large tables due to efficient algorithm and the fact that previously the same double type was used but with limited precision, meaning that data itself does not need to be modified.

Please execute the appropriate patch (SQL file) for your database; you may find these scripts in the Zabbix Git repository for:

- MySQL
- PostgreSQL
- Oracle

Warning:
Important!
* Run these scripts for the server database only.
* Make sure Zabbix server is stopped before running these scripts. Start the server afterwards.

Note that with TimescaleDB the compression support must only be turned on after applying this patch.

Note:
After upgrading database tables, please also set or update $DB['DOUBLE_IEEE754'] value to true in /ui/conf/zabbix.conf.php.

Non-root permissions implemented for Docker images

Zabbix Docker images have been updated to implement non-root container best practices. Due to the change:

- All directories have been restricted for the container user, except directories which are required for the container. For example, /etc/zabbix/ with Zabbix component configuration files.
- Ports 80 and 443 have been changed to 8080 and 8443, because usage of all ports <1024 is restricted for non-privileged users. Zabbix web interface images have been updated to use non-privileged ports 8080, 8443; Zabbix snmptrap images port 1162.
- All Zabbix images are updated to use a non-privileged user. By default, 'zabbix' with UID 1997.

A known issue: Nginx based images do not run under root. Will be fixed soon.

API changes

See the list of API changes in Zabbix 5.0.0.

Informational SNMP credentials at host interface level

Setting SNMP interface credentials has been moved from item level to host interface level. There is an **automatic** upgrade procedure that moves existing SNMP items to their appropriate interfaces. So, for example, if before upgrade there was:

```
1 SNMP interface with 1 SNMP v1 item and 1 SNMP v2 item
```

after the upgrade there will be 2 SNMP interfaces:

```
1 SNMPv1 interface with 1 SNMP v1 item
1 SNMPv2 interface with 1 SNMP v2 item
```

If there were 2 identical SNMPv3 items with different passwords before the upgrade:

```
1 SNMP interface with 1 SNMP v3 item with password="alpha" and 1 SNMP v3 item with password="beta"
```

after the upgrade there will be 2 SNMP interfaces:

```
1 SNMPv3 interface with 1 SNMP v3 item with password="alpha"
1 SNMPv3 interface with 1 SNMP v3 item with password="beta"
```

Changed Zabbix PHP file directory

The downloaded Zabbix frontend PHP files are now located in the ui directory instead of frontends/php. This is relevant when using Zabbix sources for the installation.

Changed acknowledgment screen URL

URL parameters of the problem update (acknowledgment) screen have changed. For example, if previously the page parameters were:

`?action=acknowledge.edit&eventids[]=100`

in the new version they are:

`?action=popup&popup_action=acknowledge.edit&eventids[]=100`

In a related development, when successfully updating a problem from a dashboard widget, only the widget gets reloaded, not the whole page. So the content of another widget displaying the same problem will remain unchanged until the next scheduled widget refresh or complete page refresh.

No data triggers sensitive to proxy availability

No data triggers are now, by default, sensitive to proxy availability.

Fullscreen mode replaced by hiding menu

The fullscreen mode has been removed from the Monitoring sections of the frontend. Frontend URLs containing 'fullscreen' will work no more. The same effect (showing only page title and content) now can be achieved by hiding the new vertical menu. The kiosk mode (page content only, no page title at all) remains.

Option for dropdown first entry removed

The screen for configuring frontend defaults no longer has a *Dropdown first entry* option, because dropdowns for host group and host selection have been replaced with multiselect fields in the frontend.

Configuration parameters

EnableRemoteCommands agent parameter is still supported (it may get deprecated and removed in the future) alongside the new DenyKey/AllowKey parameters. When upgrading existing agents, remote commands will not be allowed unless you:

- Set EnableRemoteCommands=1
- Remove or comment out DenyKey=system.run[*]

In this case remote commands will be allowed without restrictions. To create restrictions, use a combination of AllowKey and DenyKey parameters.

Item key limit

The maximum allowed length of an item key has been raised from 256 to 2048 characters.

Minimum Net-SNMP version

It is now possible to manually clear the SNMP cache on Zabbix server and proxy. Due to adding a new runtime control option, Net-SNMP version 5.3.0 or higher is now required for SNMP support.

Redis plugin update

Configuration parameter `Plugins.Redis.Password` was removed and an opportunity to pass a password as a key parameter has now been added. See Redis plugin for details.

Supported Elasticsearch versions changed

Elasticsearch version 7.X is now supported. Support of the older versions has been dropped.

11 Upgrade notes for 5.0.1

This minor version does not have any upgrade notes.

12 Upgrade notes for 5.0.2

Latest data

In the latest data page:

- Sorting by *Last check* has been removed
- Expanding/collapsing applications is not available (it is available again in Zabbix 5.0.3)

Regular expression support in user macro context

Regular expressions are now supported in user macro context, using the following syntax:

```
{$MACRO:regex:"regular expression"}
```

If existing macros have the unlikely `regex:something` string in the context, they will not be automatically converted to `regex:"^quoted something$"`. The change would have to be done manually.

See user macros with context for more information.

EnableRemoteCommands deprecated/unsupported by agents

The EnableRemoteCommands agent parameter is now:

- deprecated by Zabbix agent
- unsupported by Zabbix agent2

Use the AllowKey/DenyKey parameters instead.

Enhanced URL widget security

The URL dashboard widget and the URL screen element now put retrieved URL content into the sandbox. By default, all sandbox restrictions are enabled. It is possible to modify `sandbox` attribute settings in the *defines.inc.php* file, however turning sandboxing off is not recommended for security reasons. To learn more about the sandbox attribute, please see the sandbox section of the iframe HTML element description.

13 Upgrade notes for 5.0.3

Zabbix agent on IBM AIX

Zabbix agent for AIX has been enhanced with capability to monitor physical CPU utilization. A `system.cpu.util[]` item key now has an additional 4th parameter:

- system.cpu.util[<cpu>,<type>,<mode>,**<logical_or_physical>**]

The <logical_or_physical> key parameter allows to specify whether an item should report logical or physical CPU utilization (supported values: `logical`, `physical`). This new functionality uses AIX libperfstat function `perfstat_cpu_util()` which is available from AIX 6.1 TL07 and AIX 7.1 TL01 (as far as we could find). If your AIX system does not have these technology levels installed, please use an older agent version. You can check which AIX version and technology level (TL) is installed by running `oslevel` command (see description in the IBM documentation)

14 Upgrade notes for 5.0.4

Naming change for Agent 2 runtime control parameters

Agent 2 runtime-control parameters for log-level increase/decrease have been changed to be consistent with Zabbix agent.

Naming in 5.0.4	Naming before 5.0.4
log_level_increase	loglevel increase
log_level_decrease	loglevel decrease

15 Upgrade notes for 5.0.5

Dropping values outside history/trend periods

From now on values older than the configured history and trend storage period will be rejected, even if the internal housekeeping is disabled. You may want to adjust history/trend storage periods after the upgrade.

DB connection encryption setup

Parameters for encrypting the connection between frontend and the database, available during Zabbix frontend installation, have been modified. The *Configure DB connection* step of Zabbix web interface now features a new checkbox *Verify certificate* with additional options appearing upon marking it. Parameters that are unavailable in the particular configuration will be disabled. For

example, the *Database TLS encryption* checkbox cannot be checked if using MySQL and *Database host* is set to localhost. See Secure connection to the database page for the full description.

SourceIP and webhooks

The SourceIP configuration parameter is now used for webhooks.

API changes

See the list of API changes in Zabbix 5.0.5.

16 Upgrade notes for 5.0.6

Positional macros in web monitoring items

Deprecated positional macros ($1, $2, ...) are no longer used in the item names when creating web monitoring items.

In those cases where positional macros are used in existing web monitoring items, the item names will be updated in the database not to use positional macros as soon as the web scenario is updated.

Web monitoring items

Removed support of {HOST.*} macro in name property of web scenario and web scenario step.

17 Upgrade notes for 5.0.7

Zabbix agent 2 plugins

For Ceph, Docker, Memcached, MySQL, Oracle, and Redis plugins, a Uri can no longer be specified as a 1st level plugin parameter. For PostgreSQL plugin a Uri and Database name can no longer be specified as a 1st level plugin parameters. Now these parameters can only be provided at a named session level. All existing parameters in a format Plugins.<PluginName>.Uri and the Plugins.Postgres.Database parameter should be removed from the configuration file, otherwise, the agent will fail to start.

For Uri parameters, specifying a scheme is no longer mandatory.

For the default set of parameters, you can create a named session 'Default' and specify these parameters in the session. See an example below.

Before Zabbix 5.0.7	Since Zabbix 5.0.7
Plugins.Ceph.Uri="https://localhost"	Plugins.Ceph.Sessions.Default.Uri=localhost

18 Upgrade notes for 5.0.8

Host/template cloning

Long lists of objects to be cloned (items, triggers, graphs, etc) have been removed from host and template full clone forms.

Notification report

The number of notifications sent per media type is no longer displayed in parentheses after the total number in *Reports* → *Notifications* if 'All' is selected in the *Media type* dropdown.

19 Upgrade notes for 5.0.9

This minor version has no upgrade notes.

20 Upgrade notes for 5.0.10

This minor version has no upgrade notes.

21 Upgrade notes for 5.0.11

VMware event collector

The behavior of VMware event collector has been changed to fix a memory overload issue.

22 Upgrade notes for 5.0.12

This minor version has no upgrade notes.

23 Upgrade notes for 5.0.13

This minor version has no upgrade notes.

24 Upgrade notes for 5.0.14

This minor version has no upgrade notes.

25 Upgrade notes for 5.0.15

This minor version has no upgrade notes.

26 Upgrade notes for 5.0.16

Items

Zabbix agent 2 items **proc.num**, **proc.cpu.utilization**, **proc.mem** have been updated to use the latest functions introduced in Go 1.16 and thus provide better performance. Go version 1.16 or newer is now required for compiling Zabbix agent 2 to ensure correct work of these items and avoid an issue observed when compiling with older Go versions.

27 Upgrade notes for 5.0.17

This minor version has no upgrade notes.

28 Upgrade notes for 5.0.18

This minor version has no upgrade notes.

29 Upgrade notes for 5.0.19

This minor version has no upgrade notes.

30 Upgrade notes for 5.0.20

Item changes

Native support for the items **system.hw.chassis**, **system.hw.devices**, **vfs.dir.count** and **vfs.dir.size** has been added to Zabbix agent 2.

31 Upgrade notes for 5.0.21

Item changes

Native support for the items **net.dns** and **net.dns.record** has been added to Zabbix agent 2. On Zabbix agent 2 for Windows, these items now allow custom DNS IP addresses in the `ip` parameter and no longer ignore `timeout` and `count` parameters.

32 Upgrade notes for 5.0.22

This minor version has no upgrade notes.

33 Upgrade notes for 5.0.23

This minor version has no upgrade notes.

34 Upgrade notes for 5.0.24

This minor version has no upgrade notes.

35 Upgrade notes for 5.0.25

This minor version has no upgrade notes.

36 Upgrade notes for 5.0.26

This minor version has no upgrade notes.

37 Upgrade notes for 5.0.27

This minor version has no upgrade notes.

38 Upgrade notes for 5.0.28

This minor version has no upgrade notes.

39 Upgrade notes for 5.0.29

This minor version has no upgrade notes.

40 Upgrade notes for 5.0.30

This minor version has no upgrade notes.

41 Upgrade notes for 5.0.31

Improved performance of history syncers The performance of history syncers has been improved by introducing a new read-write lock. This reduces locking between history syncers, trappers and proxy pollers by using a shared read lock while accessing the configuration cache. The new lock can be write locked only by the configuration syncer performing a configuration cache reload.

Patch for API queries optimization Optional patches are now available for optimizing API database queries, created when searching through names in *hosts* and *items* tables. To apply the patches, run the file *index_host_and_item_name_upper_field.sql* manually. Note that deterministic triggers need to be created during patch application. See Known issues for instructions how to apply the patch.

Query separate tablespaces in Oracle databases with Zabbix agent 2 The following Zabbix agent 2 items, supported for the Oracle plugin, now have additional optional parameters:

- oracle.diskgroups.stats[<existingParameters>,**<diskgroup>**]
- oracle.archive.info[<existingParameters>,**<destination>**]
- oracle.cdb.info[<existingParameters>,**<database>**]
- oracle.pdb.info[<existingParameters>,**<database>**]
- oracle.ts.stats[<existingParameters>,**<tablespace>**,**<type>**]

These parameters allow to query separate instances of data instead of all data, thus improving performance.

42 Upgrade notes for 5.0.32

Limits for JavaScript objects in preprocessing The following limits for JavaScript objects in preprocessing have been introduced:

- The total size of all messages that can be logged with the `Log()` method has been limited to 8 MB per script execution.
- The initialization of multiple `CurlHttpRequest` objects has been limited to 10 per script execution.
- The total length of header fields that can be added to a single `CurlHttpRequest` object with the `AddHeader()` method has been limited to 128 Kbytes (special characters and header names included).

43 Upgrade notes for 5.0.33

This minor version does not have any upgrade notes.

44 Upgrade notes for 5.0.34

This minor version does not have any upgrade notes.

45 Upgrade notes for 5.0.35

This minor version does not have any upgrade notes.

46 Upgrade notes for 5.0.36

This minor version does not have any upgrade notes.

47 Upgrade notes for 5.0.37

This minor version does not have any upgrade notes.

48 Upgrade notes for 5.0.38

This minor version does not have any upgrade notes.

49 Upgrade notes for 5.0.39

This minor version does not have any upgrade notes.

50 Upgrade notes for 5.0.40

This minor version does not have any upgrade notes.

51 Upgrade notes for 5.0.41

This minor version does not have any upgrade notes.

52 Upgrade notes for 5.0.42

Zabbix agent 2 support on Windows

To prevent critical security vulnerabilities, the minimum Windows version for Zabbix agent 2 has been raised to Windows 10/Server 2016. See note under Supported platforms for more information.

53 Upgrade notes for 5.0.43

This minor version does not have any upgrade notes.

54 Upgrade notes for 5.0.44

This minor version does not have any upgrade notes.

55 Upgrade notes for 5.0.45

This minor version does not have any upgrade notes.

56 Upgrade notes for 5.0.46

Breaking changes Java 11 required for Java gateway

Zabbix Java gateway now requires Java 11 for runtime (building from source is still possible with Java 8), due to updated minimum logback library versions:

Library	New minimum version	Old minimum version
logback-classic	**1.5.16**	1.2.9
logback-core	**1.5.16**	1.2.9
slf4j-api	**2.0.16**	1.7.32

57 Upgrade notes for 5.0.47

This minor version does not have any upgrade notes.

5. Quickstart

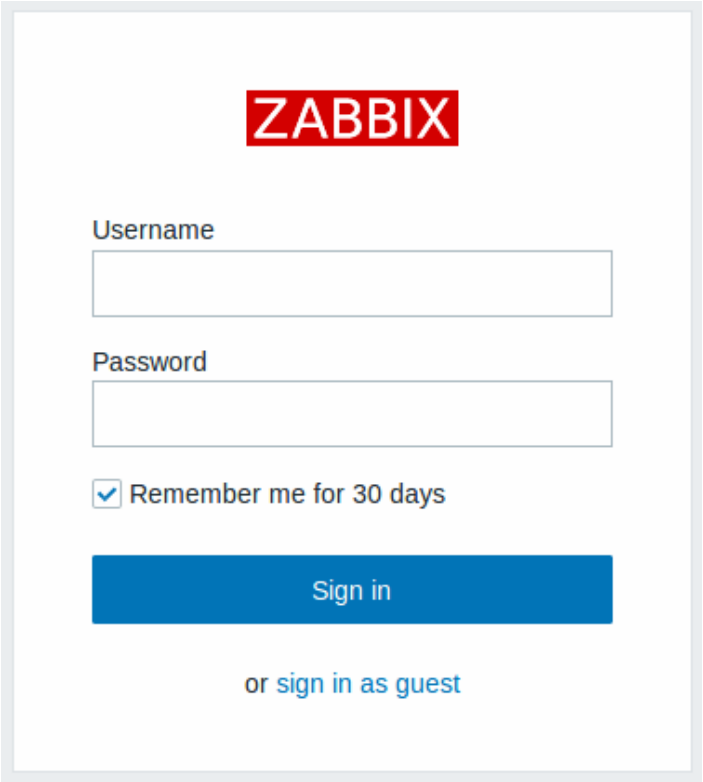
Please use the sidebar to access content in the Quickstart section.

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 user name **Admin** with password **zabbix** to log in as a Zabbix superuser. Access to *Configuration* and *Administration* menus will be granted.

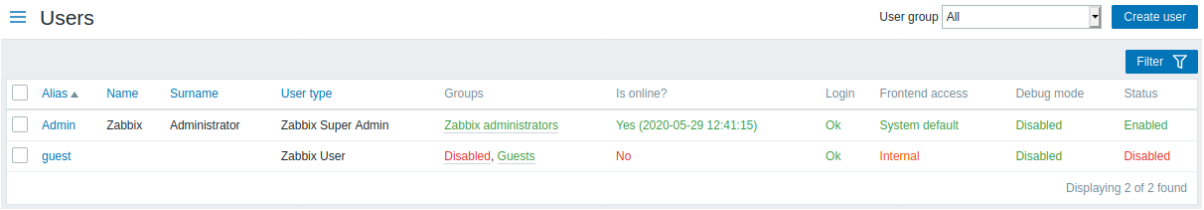
Protection against brute force attacks

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.

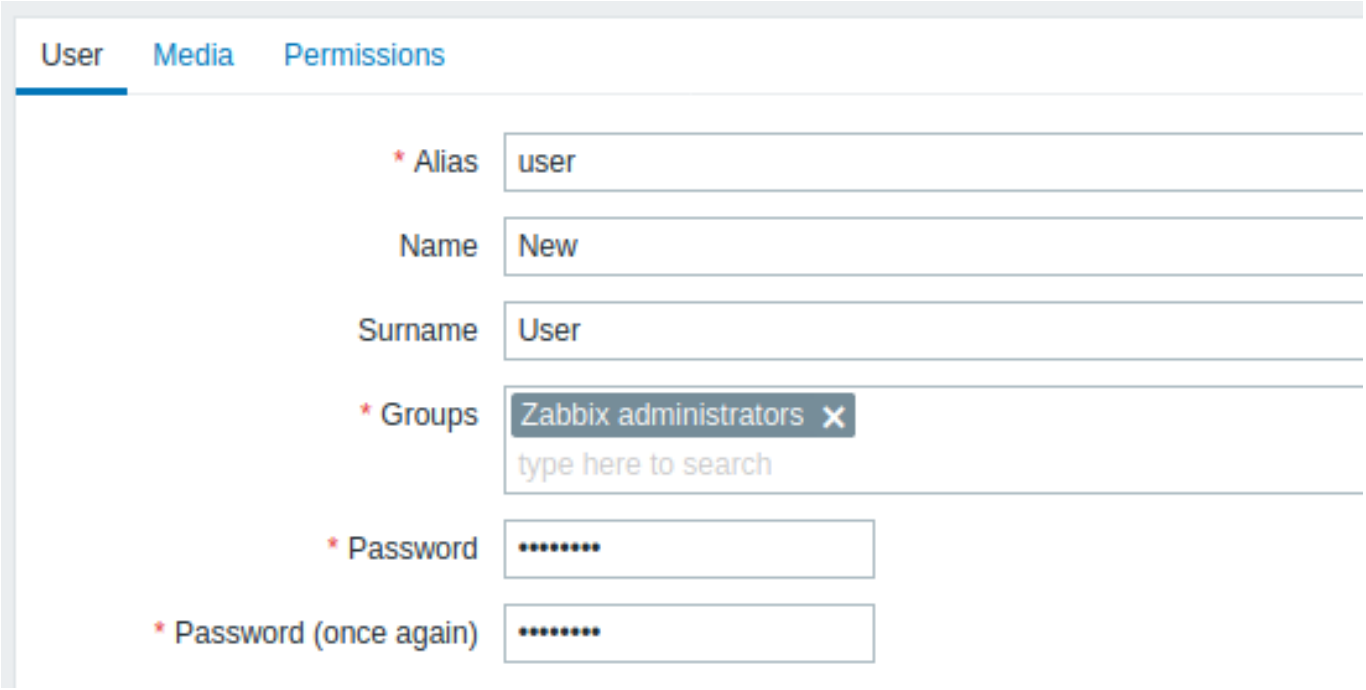
Adding user

To view information about users, go to *Administration* → *Users*.



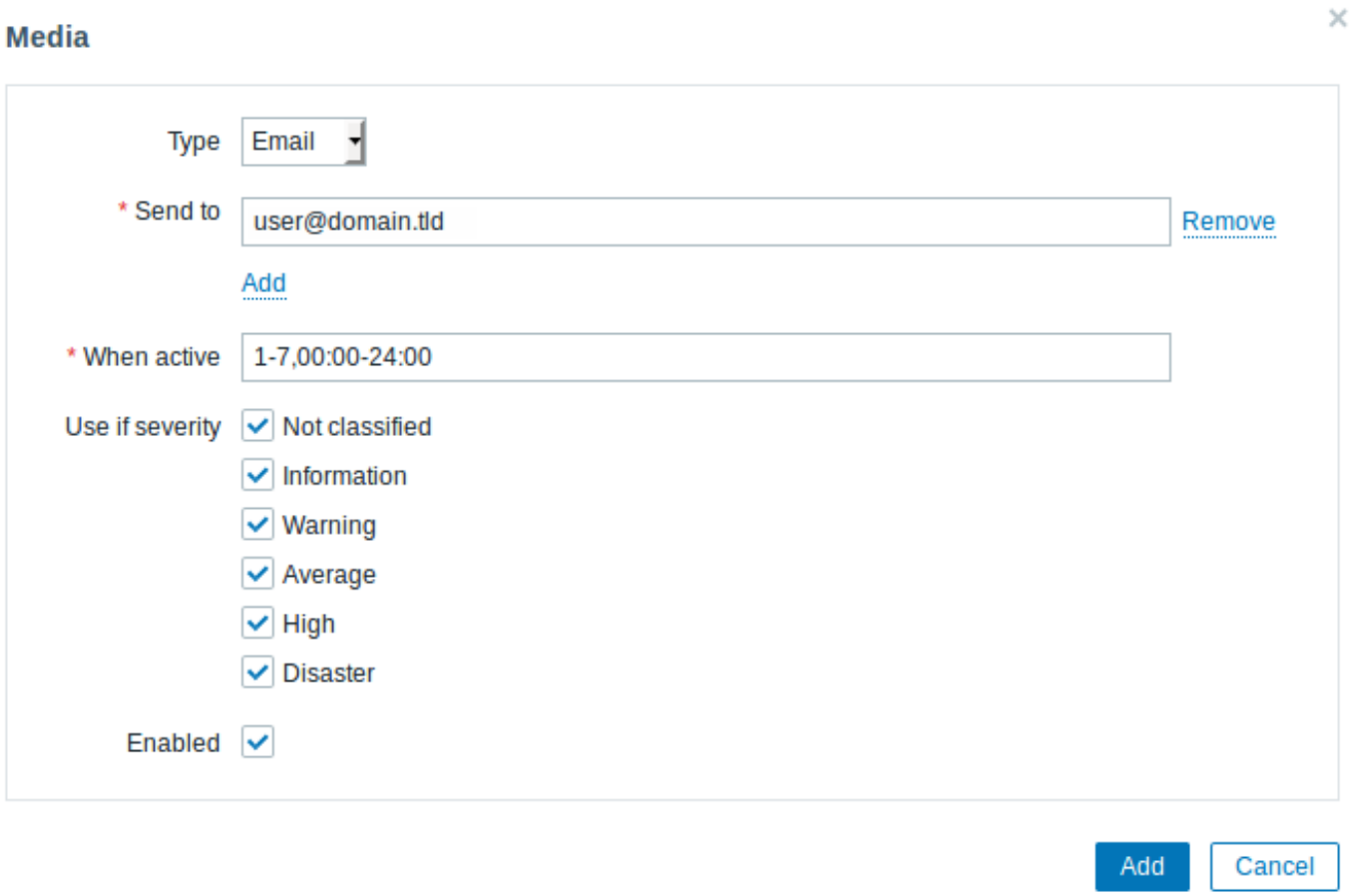
To add a new user, click on *Create user*.

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.

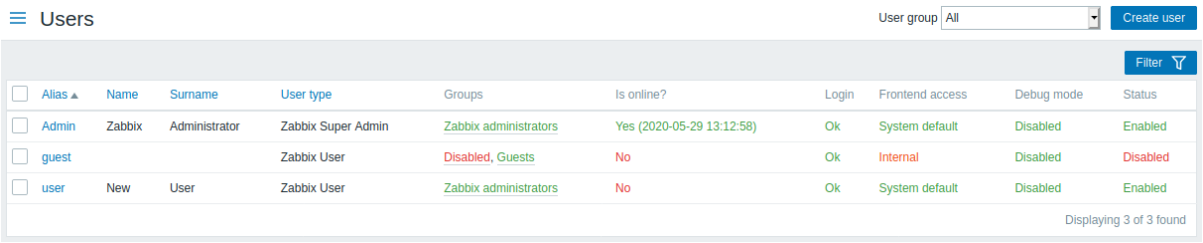
By default, new users have no media (notification delivery methods) defined for them. To create one, go to the 'Media' tab and click on *Add*.



In this pop-up, enter an e-mail address for the user.

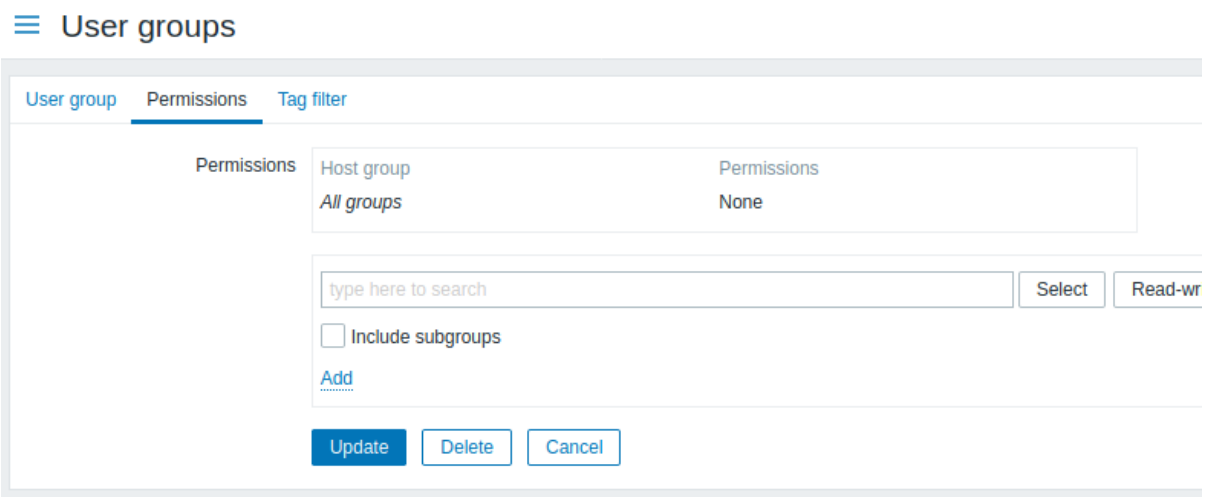
You can specify a time period when the medium will be active (see Time period specification page for description of the format), by default a medium is always active. You can also customize trigger severity levels for which the medium will be active, but leave all of them enabled for now.

Click on *Add*, then click *Add* in the user properties form. The new user appears in the userlist.

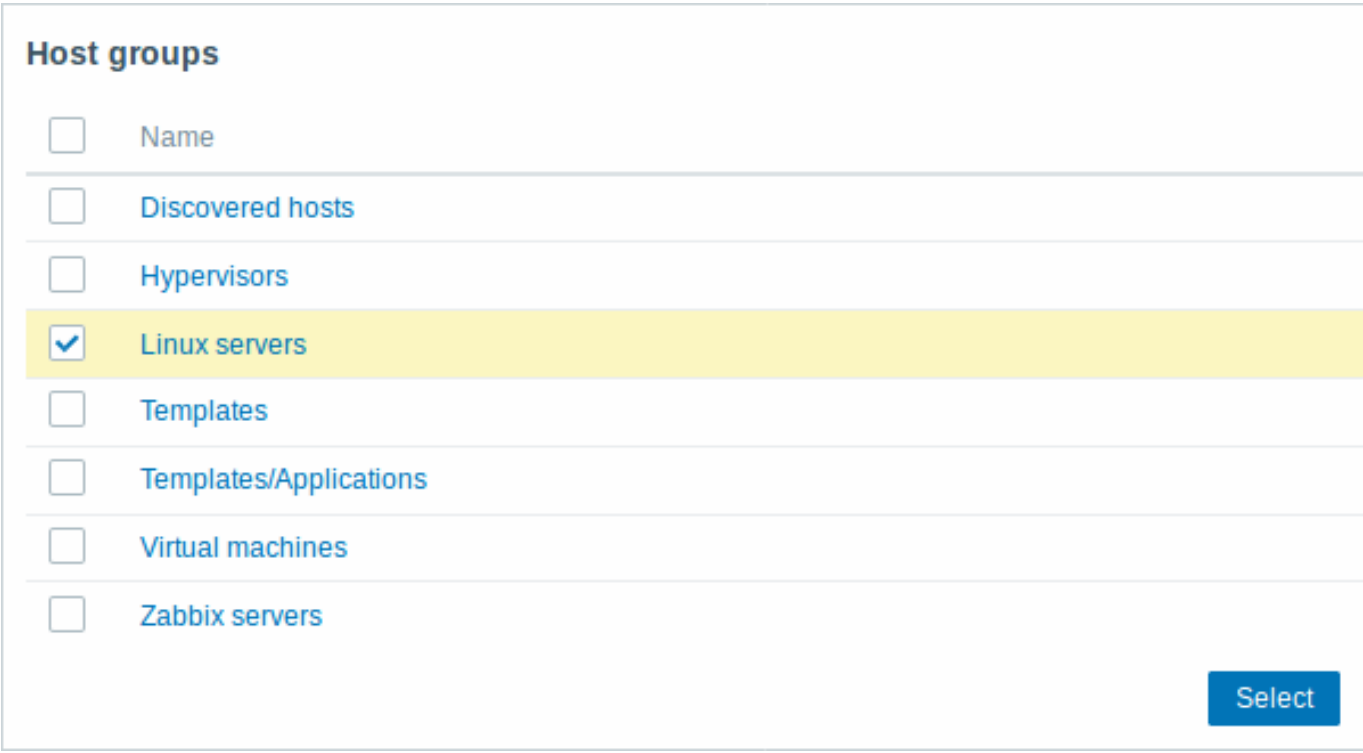


Adding permissions

By default, a new user has no permissions to access hosts. 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 *Permissions* tab.



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 permission level and then *Add* to add the group to the list of permissions. In the user group properties form, click *Update*.

Attention:
In Zabbix, access rights to hosts 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

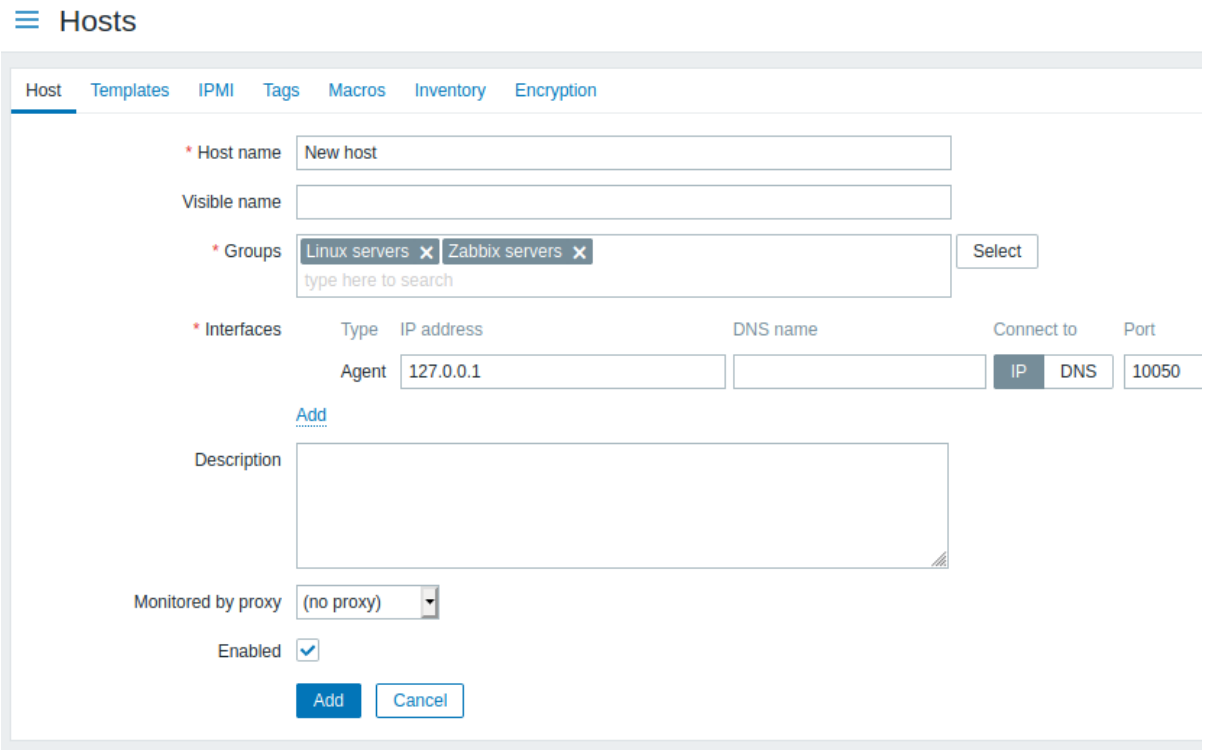
In this section you will learn how to set up a new host.

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*. 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.

Groups

- Select one or several existing groups by clicking *Select* button or enter a non-existing group name to create a new group.

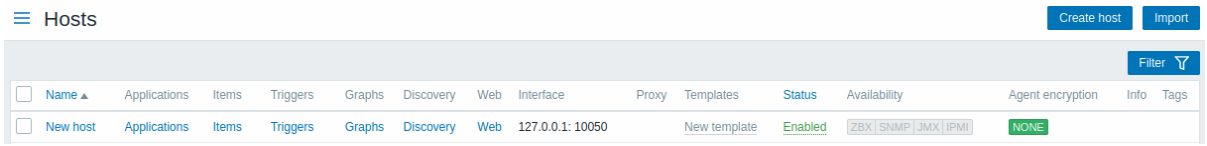
Note:
All access permissions are assigned to host groups, not individual hosts. That is why a host must belong to at least one group.

IP address

- 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:

- ZBX - host status has not been established; no metric check has happened yet
- ZBX - host is available, a metric check has been successful
- ZBX - 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

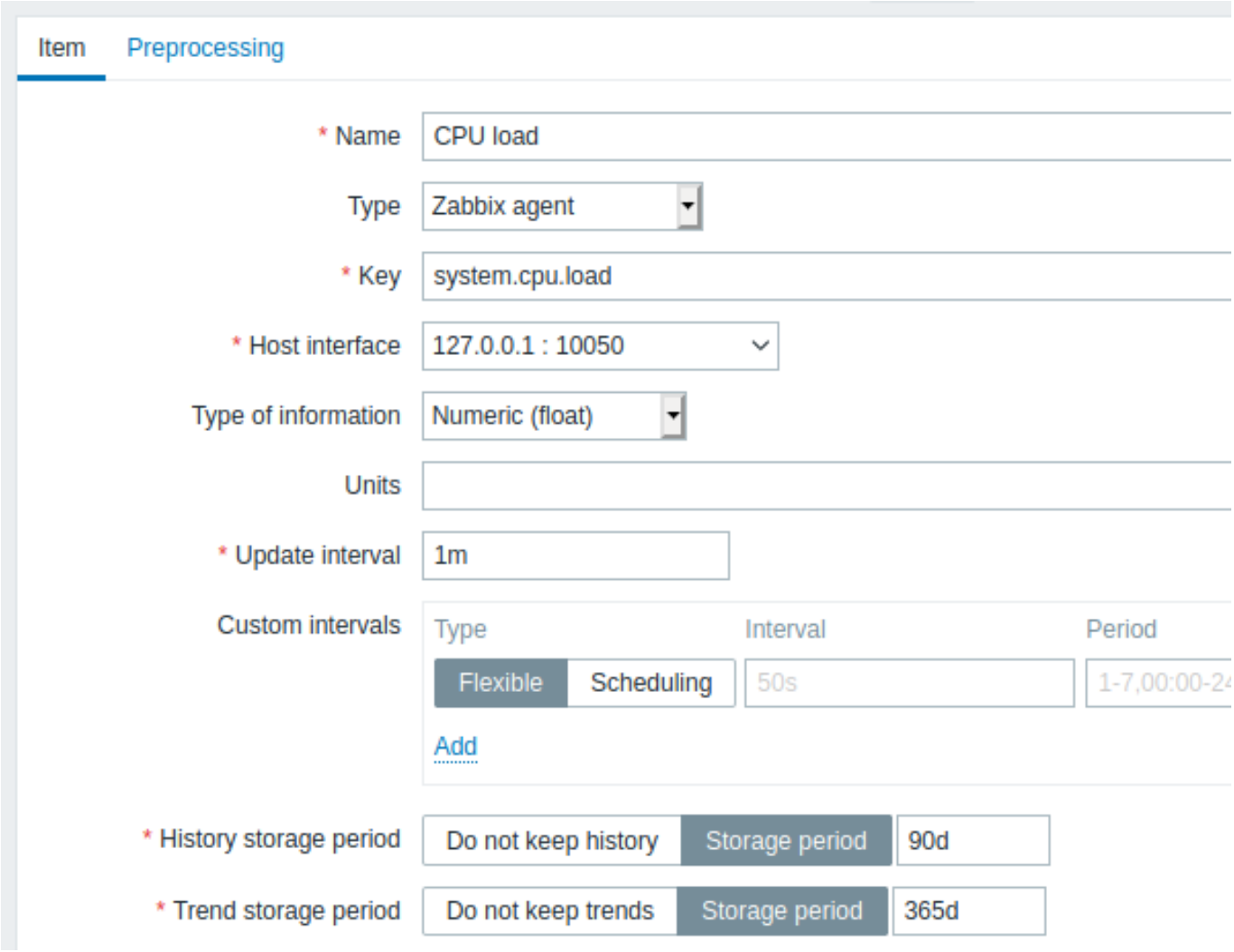
In this section you will learn how to set up an item.

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 data to get off of 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 a 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.

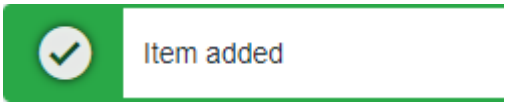
Type of information

- Select *Numeric (float)* here. This attribute defines the format of expected data.

Note:
You may also want to reduce the amount 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.

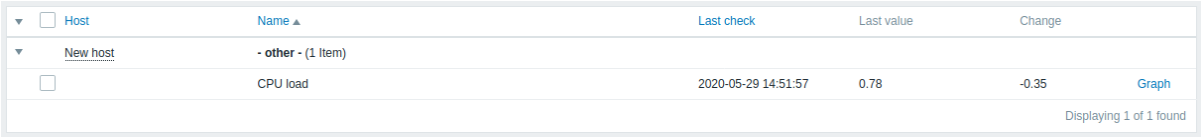
When done, click *Add*. The new item should appear in the item list, and you should see a success message.



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*.

Then click on the **+** before **- other -** and expect your item to be there and displaying data.



With that said, first data may take up to 60 seconds 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.

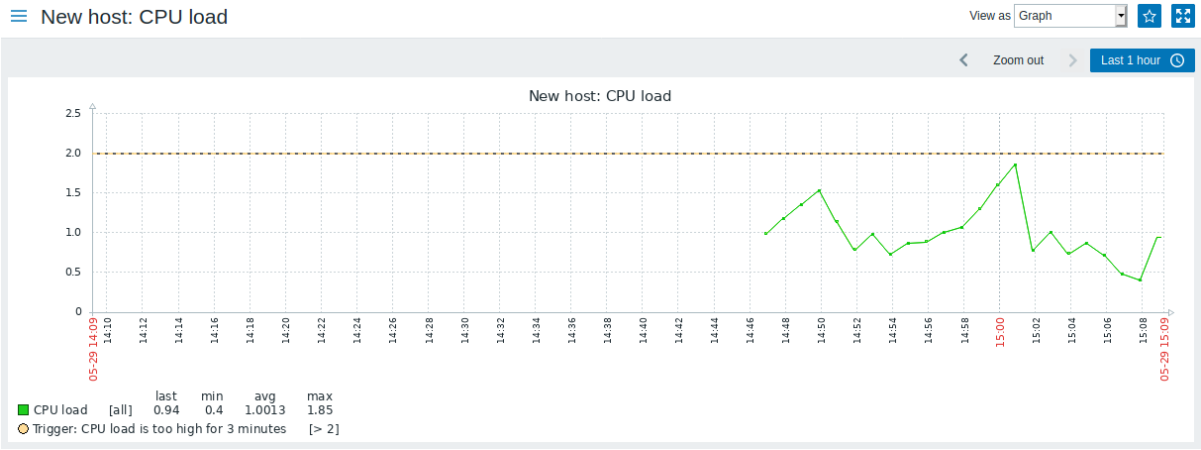
If you do not see information about the item as in the screenshot, make sure that:

- you entered item 'Key' and 'Type of information' fields exactly as in the screenshot
- both agent and server are running
- host status is 'Monitored' and its availability icon is green
- host is selected in the host dropdown, 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

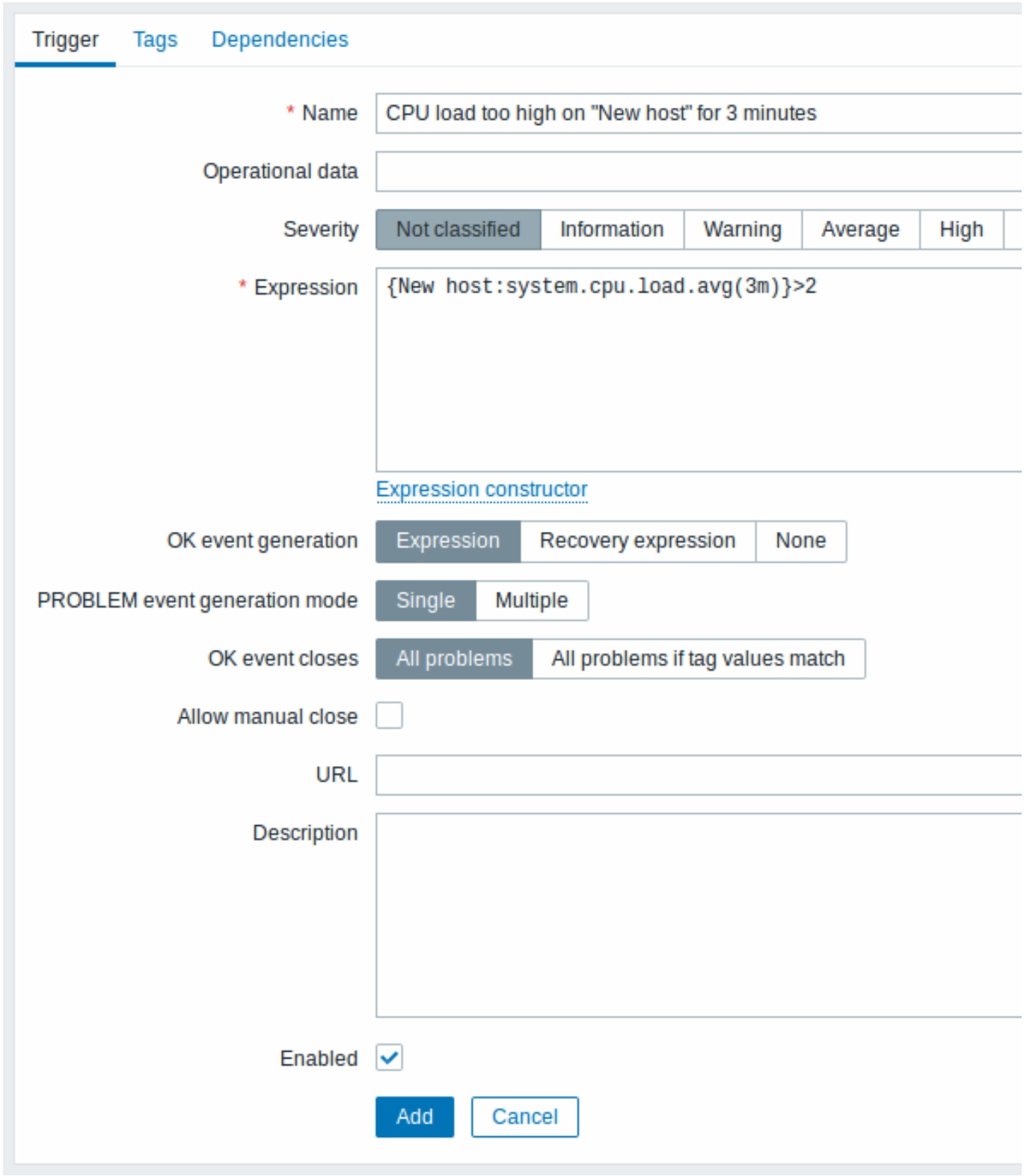
In this section you will learn how to set up a trigger.

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: {New host:system.cpu.load.avg(3m)}>2

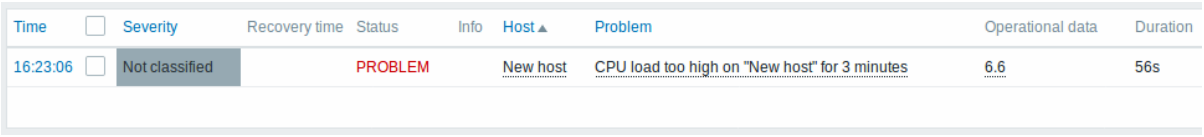
This is the trigger expression. Make sure that the expression is entered right, down to the last symbol. The item key here (system.cpu.load) is used to refer to the item. This particular expression basically says that the problem threshold is exceeded when the CPU load average value for 3 minutes is 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.

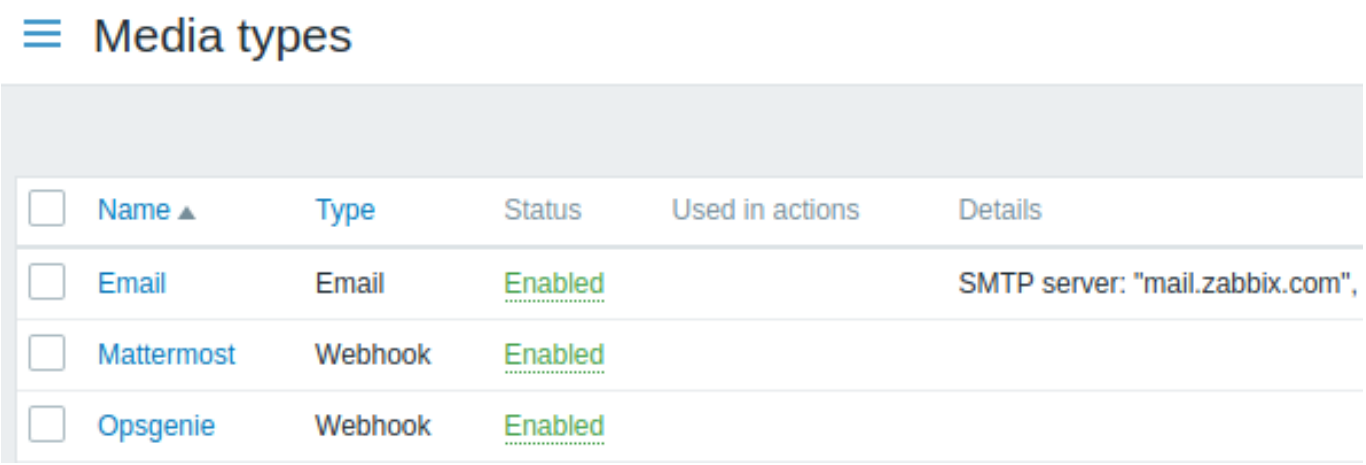
With items collecting data and triggers designed to ”fire” upon problem situations, it would also be useful to have some alerting mechanism in place that would notify us about important events even when we are not directly looking at Zabbix frontend.

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.

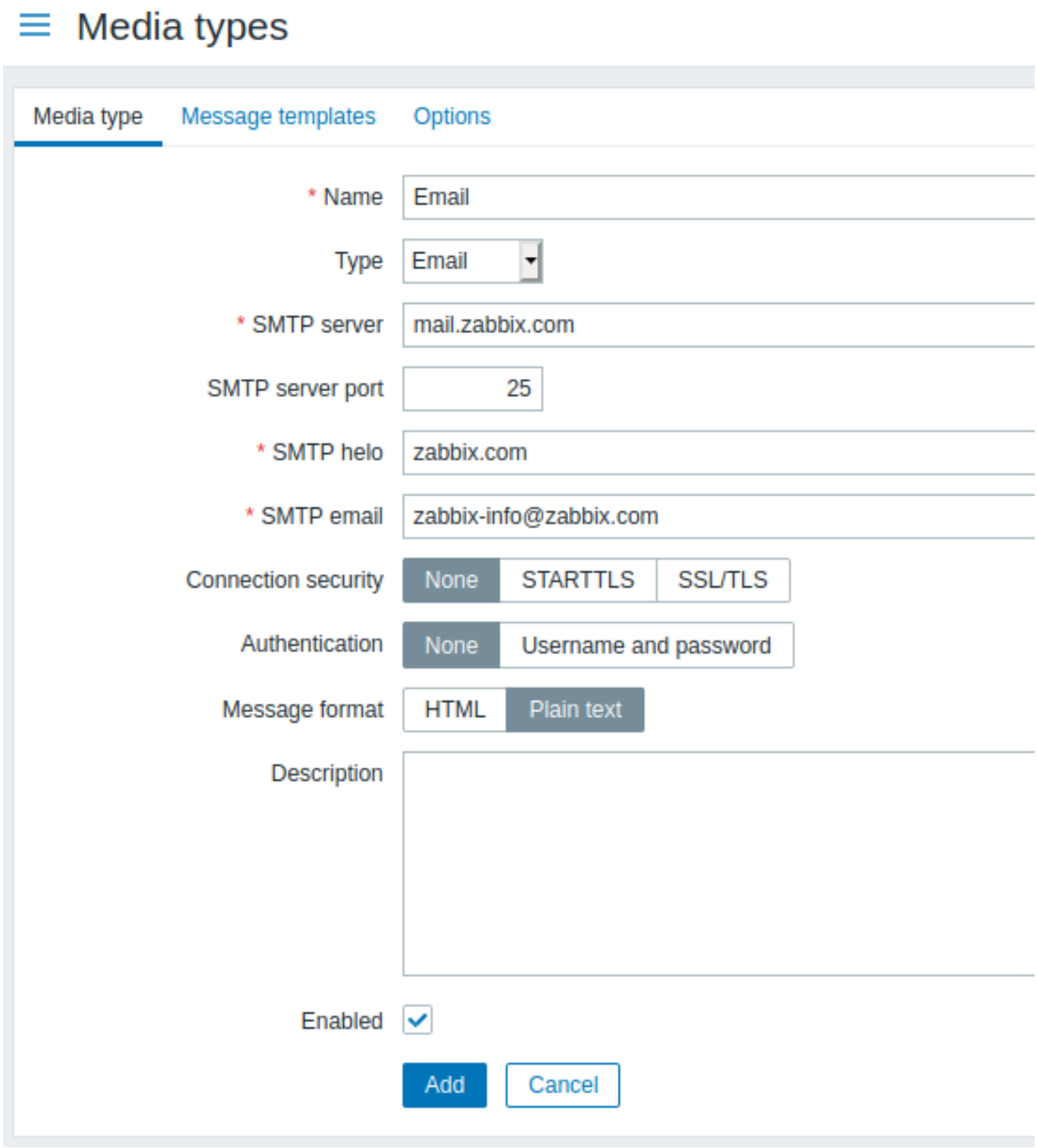
E-mail settings

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.



This will present us with the e-mail settings definition form.



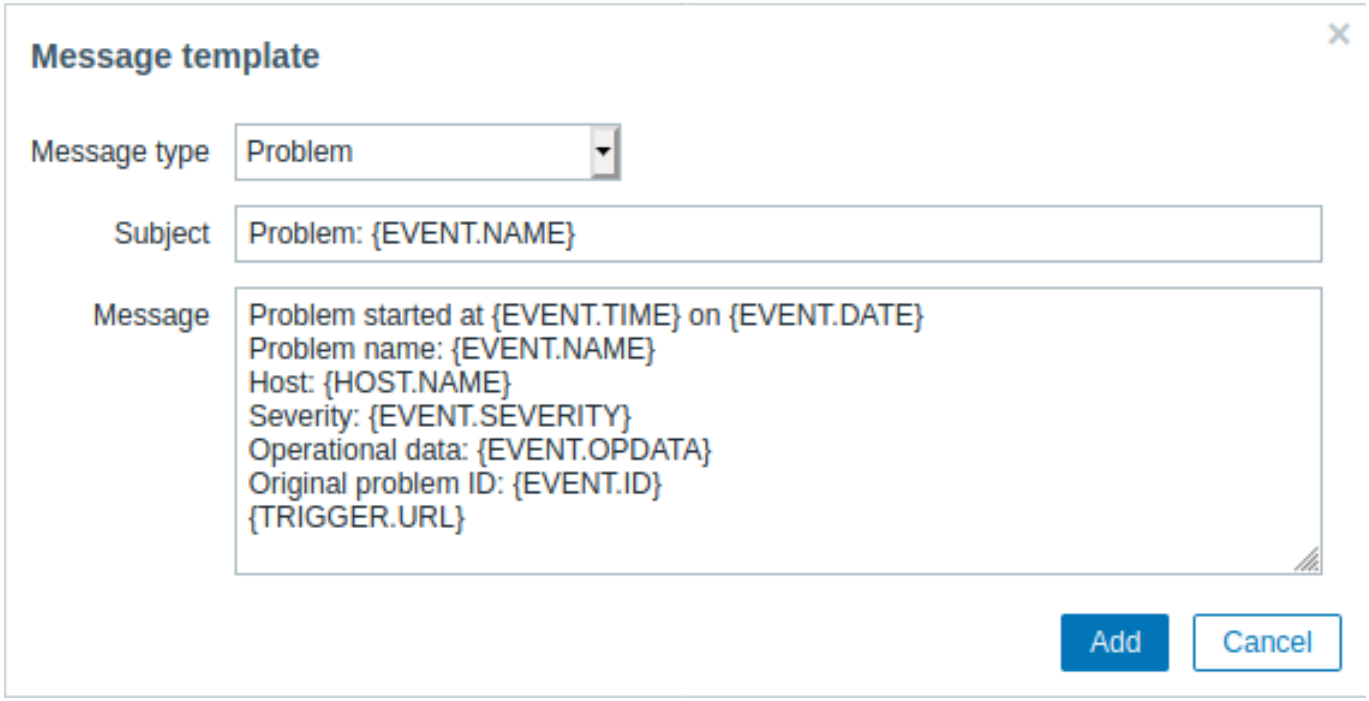
All mandatory input fields are marked with a red asterisk.

In the *Media type* tab, set the values of SMTP server, SMTP helo and SMTP e-mail to the appropriate for your environment.

Note:
'SMTP email' will be used as the 'From' address for the notifications sent from Zabbix.

Next, it is required to define the content of the problem message. The content is defined by means of a message template, configured in the *Message templates* tab.

Click on *Add* to create a message template, and select *Problem* as the message type.

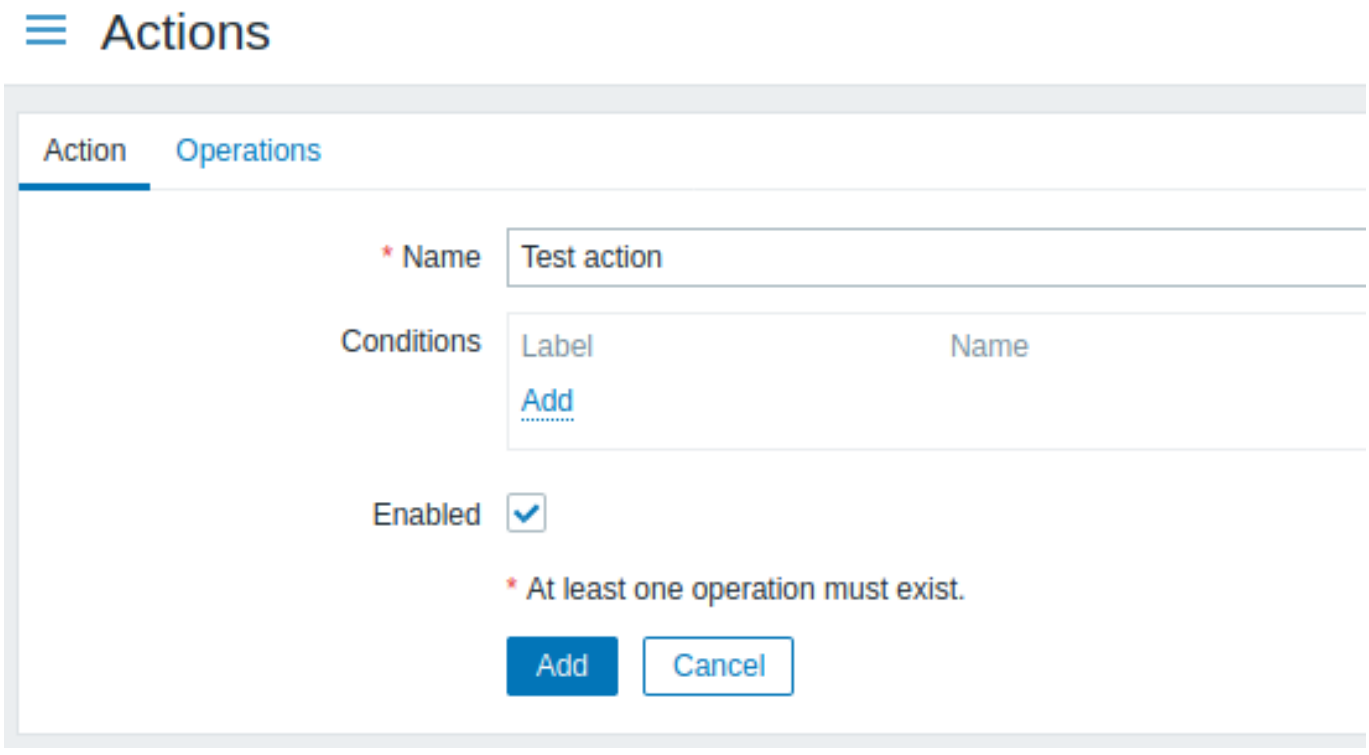


Click on *Add* when ready and save the form.

Now you have configured 'Email' as a working media type. The media type must also 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*.

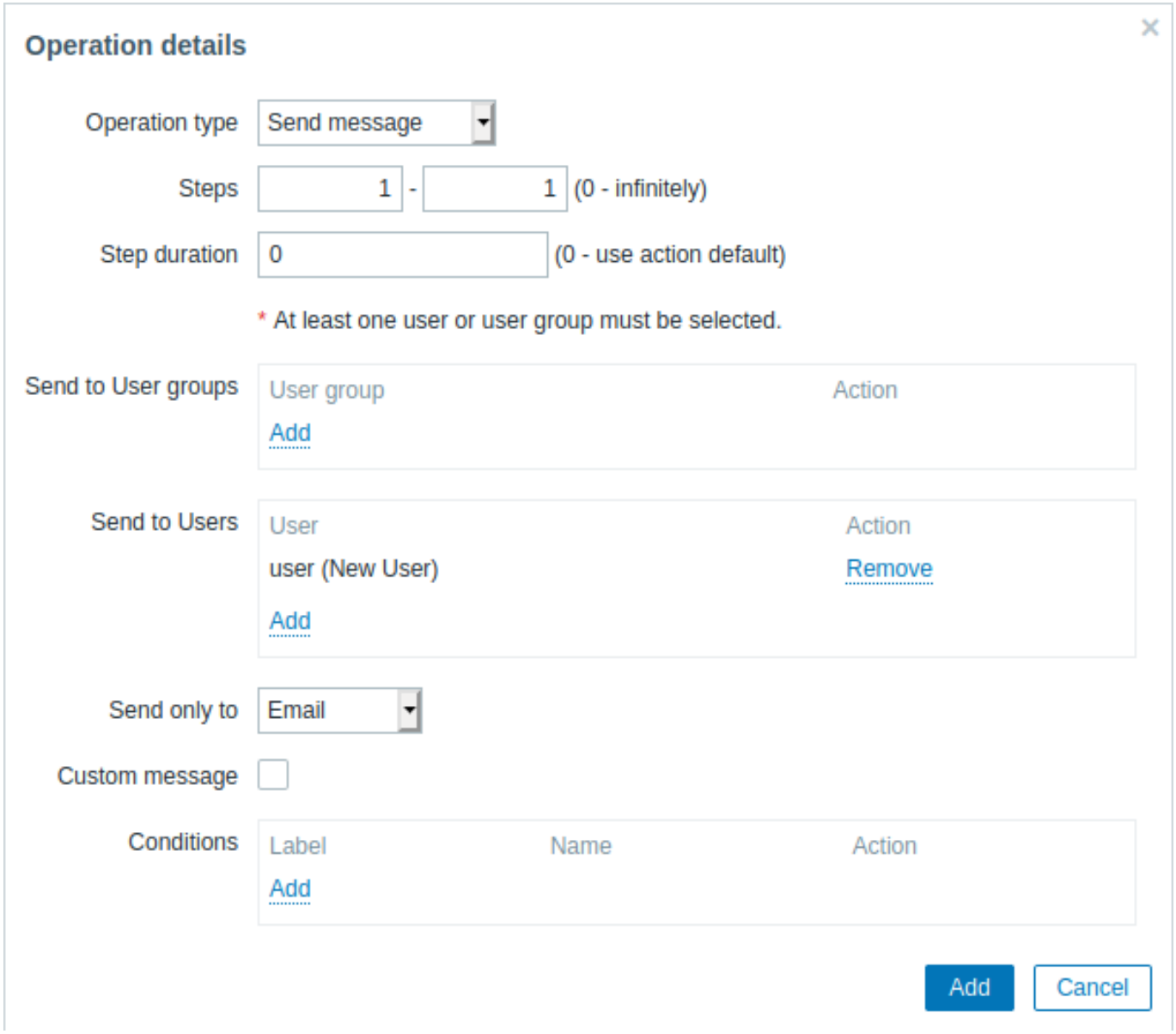


All mandatory input fields are marked with a red asterisk.

In this form, enter a name for the action.

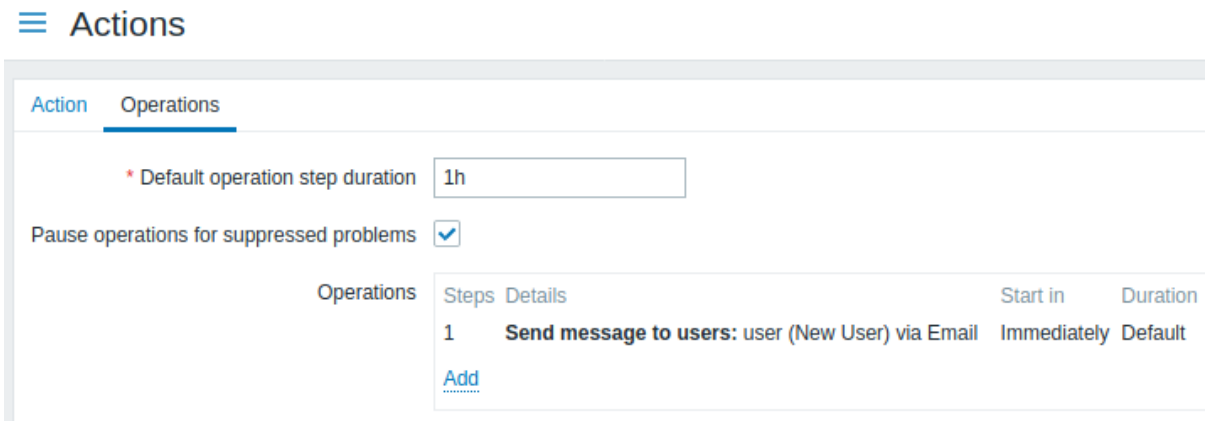
In the most simple case, if we do not add any more specific conditions, the action will be taken upon any trigger change from 'Ok' to 'Problem'.

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

Attention:
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.

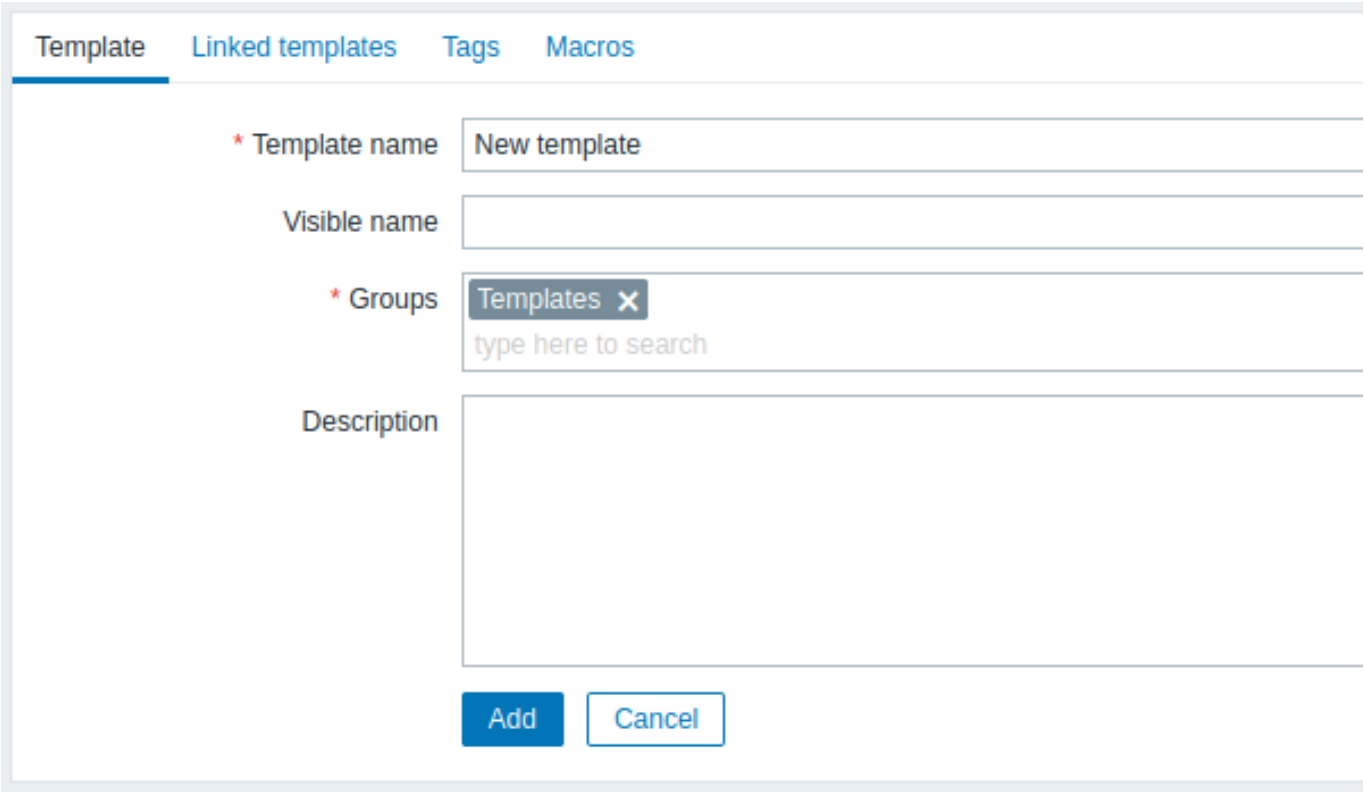
While all of these steps offer a great deal of flexibility in themselves, it may appear like a lot of steps to take if needed for, say, a thousand hosts. Some automation would be handy.

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 by applying 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.



All mandatory input fields are marked with a red asterisk.

The required parameters to enter here are:

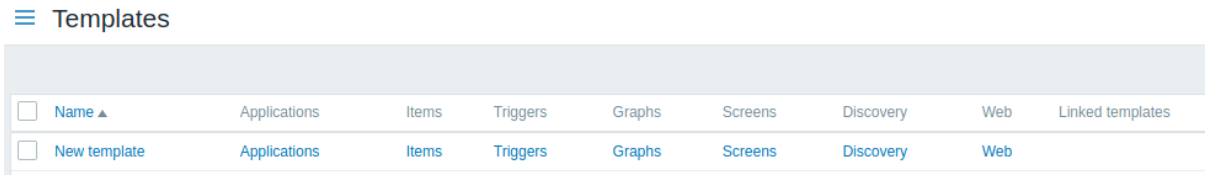
Template name

- Enter a template name. Alpha-numericals, spaces and underscores are allowed.

Groups

- Select one or several groups by clicking *Select* button. The template must belong to a group.

When done, click *Add*. Your new template should be visible in the list of templates.



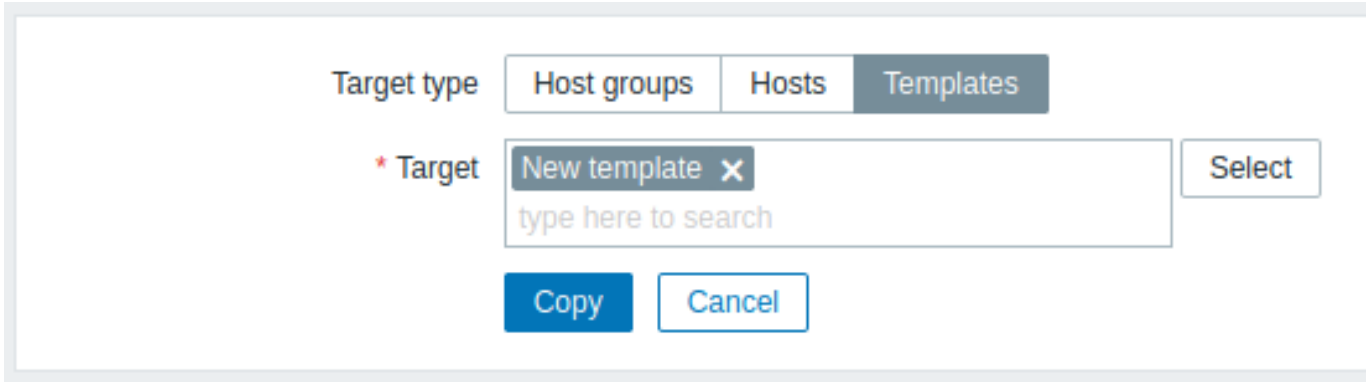
As you may see, the template is there, but it holds nothing in it - no items, triggers or other entities.

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*

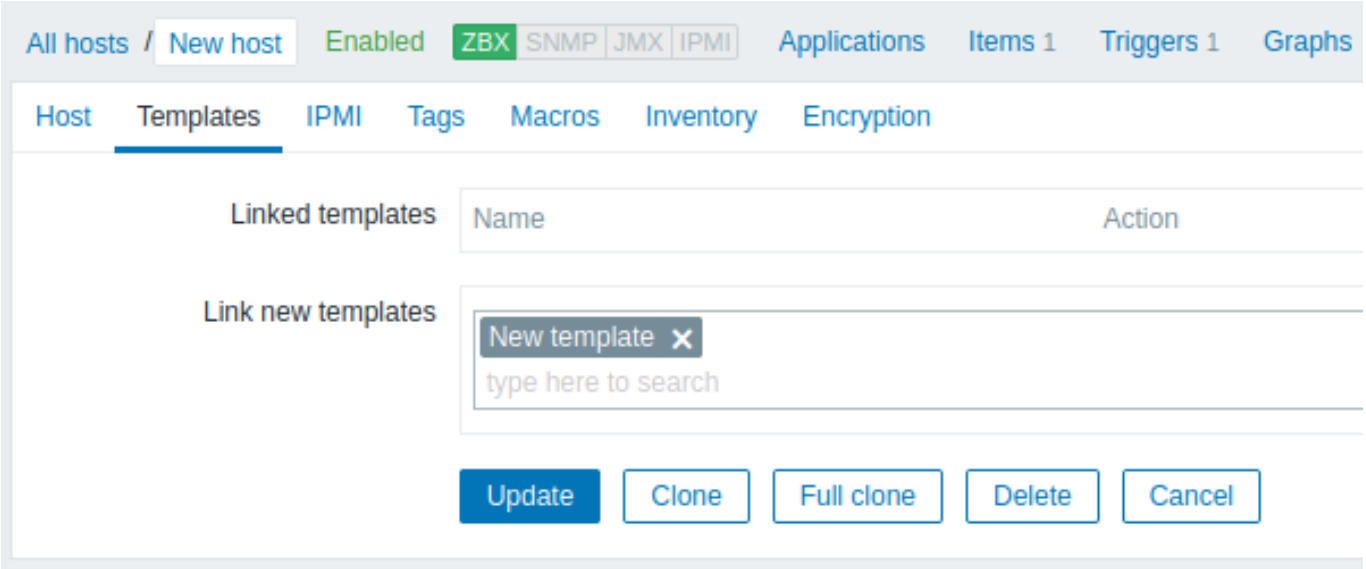
If you now go to *Configuration* → *Templates*, 'New template' should have one new item in it.

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 go to the **Templates** tab.

There, start typing *New template* in the *Link new 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 *Link new 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.

As you may have guessed, 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).

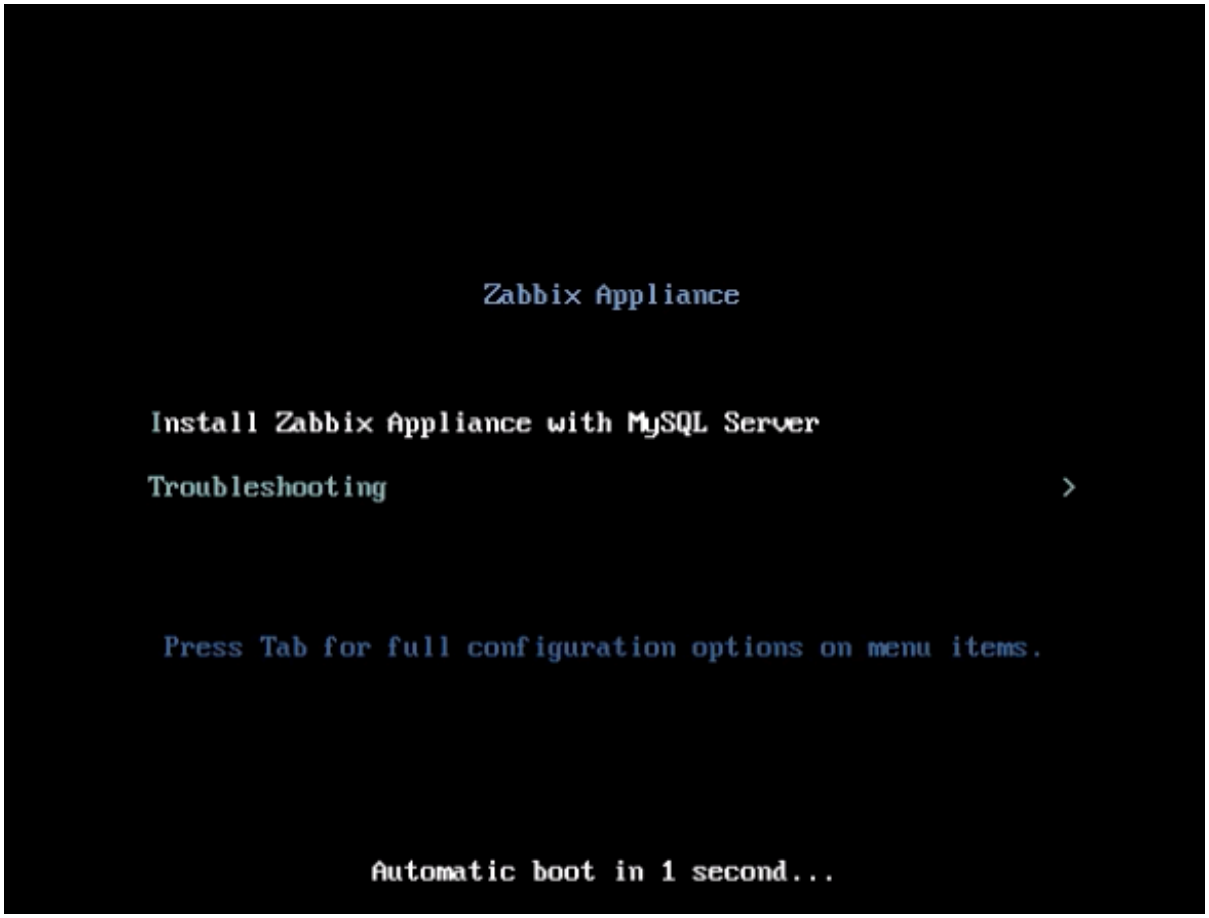
Zabbix appliance installation CD can be used for instant deployment of Zabbix server (MySQL).

Attention:
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.

Attention:
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).

Note:
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 CentOS 8. There are some changes applied to the base CentOS configuration.

1.1 Repositories

Official Zabbix repository has been added to */etc/yum.repos.d*:

```
[zabbix]
name=Zabbix Official Repository - $basearch
baseurl=http://repo.zabbix.com/zabbix/5.0/rhel/8/$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 time zone

By default the appliance uses UTC for the system clock. To change the time zone, 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>

Note:
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.

Note:
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 vCenter Converter (authentication required for download). If you use VMWare vCenter Converter, you may encounter issues with the hybrid network adapter. In that case, you can try specifying the E1000 adapter during the conversion process. Alternatively, after the conversion is complete, you can delete the existing adapter and add an E1000 adapter.

7.2 HDD/flash image (raw)

```
dd if=./zabbix_appliance_5.0.0.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

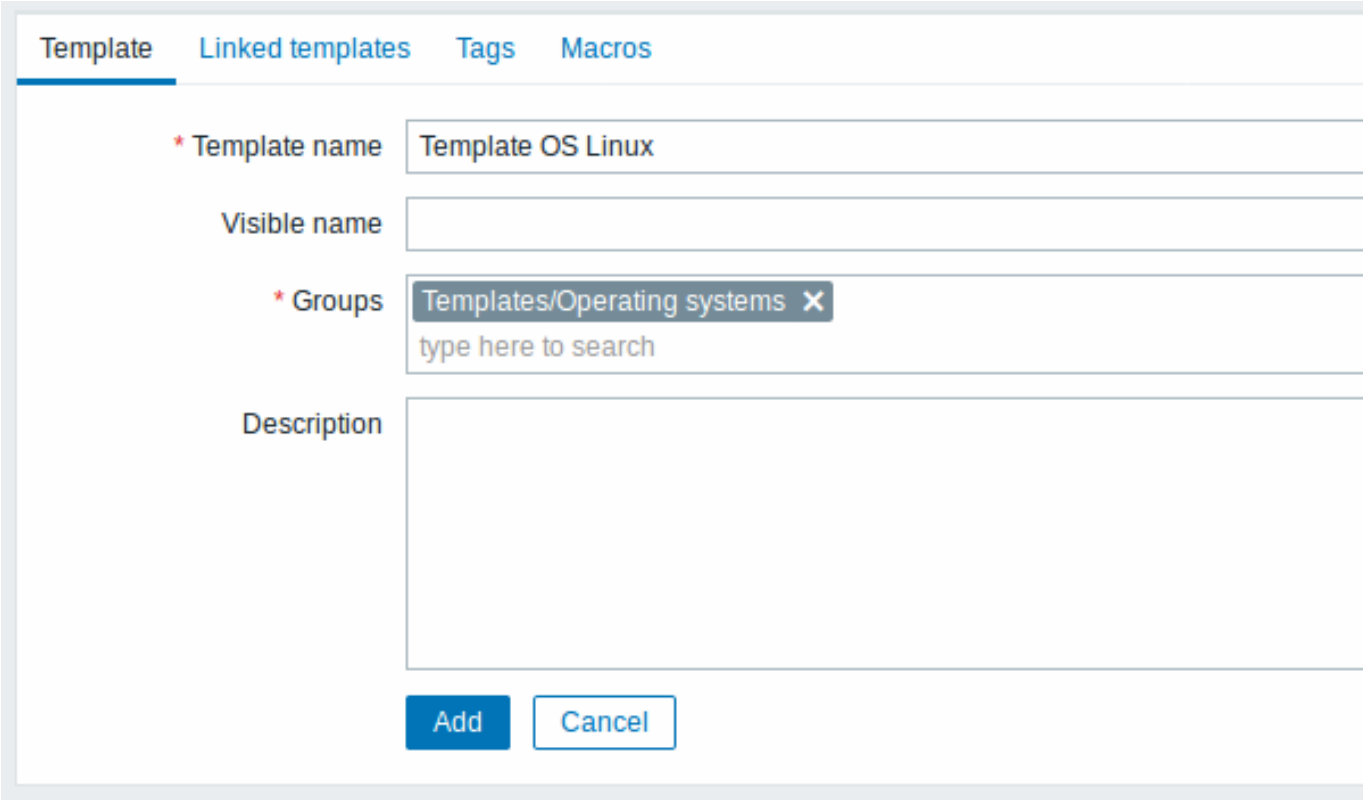
Configuring a template requires that you first create a template by defining its general parameters and then you add entities (items, triggers, graphs etc.) to it.

Creating a template

To create a template, do the following:

- Go to *Configuration → Templates*
- Click on *Create template*
- Edit template attributes

The **Template** tab contains general template attributes.



All mandatory input fields are marked with a red asterisk.

Template attributes:

Parameter	Description
Template name	Unique template name. Alphanumerics, spaces, dots, dashes and underscores are allowed. However, leading and trailing spaces are disallowed.
Visible name	If you set this name, it will be the one visible in lists, maps, etc.
Groups	Host/template groups the template belongs to.
Description	Enter the template description.

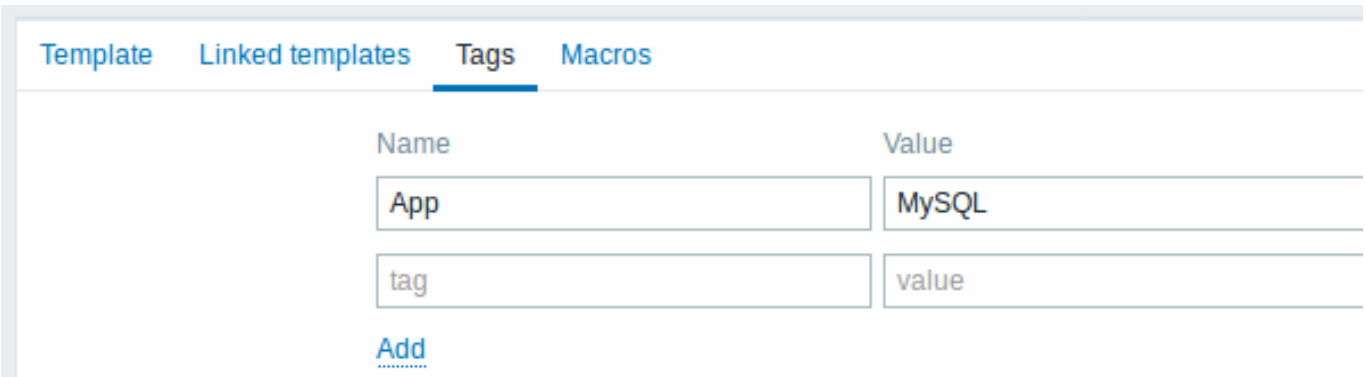
The **Linked templates** tab allows you to link one or more "nested" templates to this template. All entities (items, triggers, graphs etc.) will be inherited from the linked templates.

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

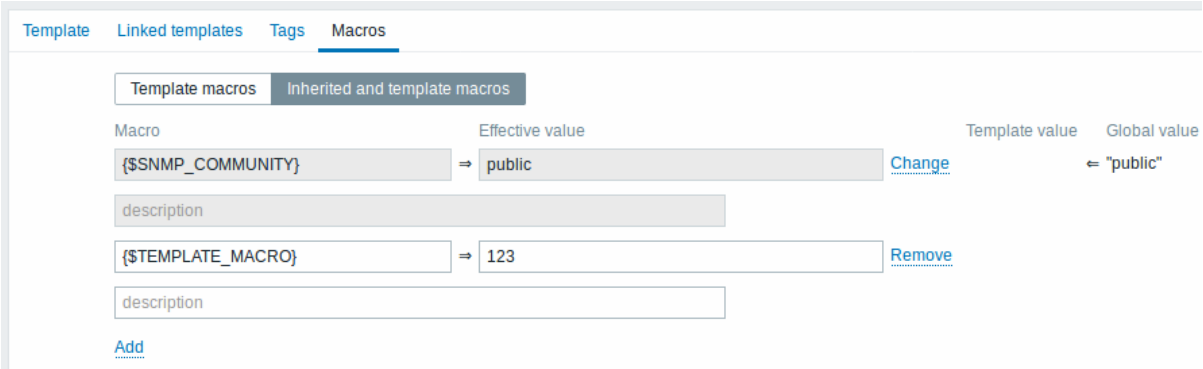
The **Tags** tab allows you to define template-level tags. All problems of hosts linked to this template will be tagged with the values entered here.



User macros, {INVENTORY.*} macros, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {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 macro-value pairs. 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.

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.
Delete and clear	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.

Attention:
Items have to be added to a template first. Triggers and graphs cannot be added without the corresponding item.

Adding items, triggers, graphs

There are two ways to add items to the template:

1. To create new items, follow the guidelines for Creating an item.

2. To add existing items to the template:

- Go to *Data collection* → *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 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 screens

To add screens to a template in *Configuration* → *Templates*, do the following:

- Click on *Screens* in the row of the template
- Configure a screen following the usual method of configuring screens

Attention:
The elements that can be included in a template screen are: simple graph, custom graph, clock, plain text, URL.

Note:
For details on accessing host screens that are created from template screens, see the host screen 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

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.

Attention:
Templates are linked directly to individual hosts and not to host groups. Simply adding a template to a host group will not link it. Host groups are used only for logical grouping of hosts and templates.

Linking a template

To link a template to the host, do the following:

- Go to *Configuration* → *Hosts*
- Click on the required host and switch to the *Templates* tab
- 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 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.

Attention:
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.

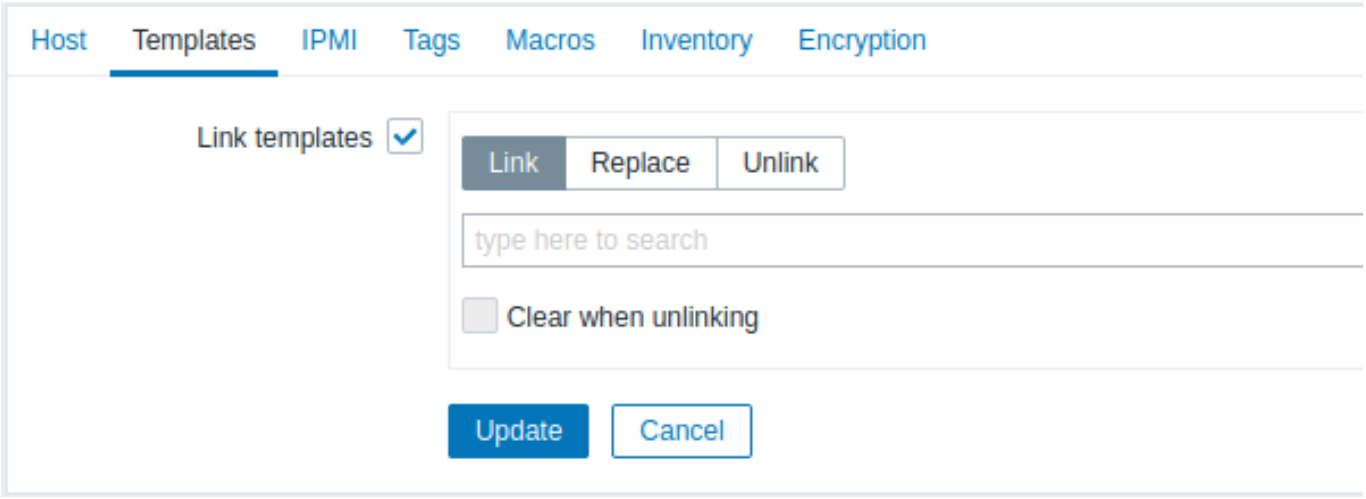
Entity uniqueness criteria

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
- for applications - application name

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 in the *Templates* tab select to link additional templates:



Select *Link templates* and 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.).

Note:
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 you 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.

Attention:
Any customizations to the entities implemented on a template-level will override the previous customizations of the entities on a host-level.

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 switch to the *Templates* tab
- 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 entities on individual templates 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 a single template.

The benefit of nesting is that you have to link only one template (”nest”, parent template) to the host and the host will inherit all entities of the linked templates (”nested”, child templates) automatically. For example, if we link templates T1 and T2 to template T3, we supplement T3 with entities from T1 and T2, and not vice versa. If we link template A to templates B and C, we supplement B and C with entities from A.

Configuring nested templates

To link templates, you need to take an existing template or a new one, and then:

- Open the template configuration form
- Look for the *Linked templates* tab
- Click *Select* to open the *Templates* popup window
- In the popup window, choose required templates, then click *Select* to add the templates to the list
- Click *Add* or *Update* in the template configuration form

Thus, all entities of the parent template, as well as all entities of linked templates (such as items, triggers, graphs, etc.) will now appear in the template configuration, except for linked template screens, 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 *Update*.

Choosing the *Unlink* option will simply remove the association with the linked template, while not removing all its entities (items, triggers, graphs, etc.).

Choosing the *Unlink and clear* option will remove both the association with the linked template and all its entities (items, triggers, graphs, etc.).

4 Mass update

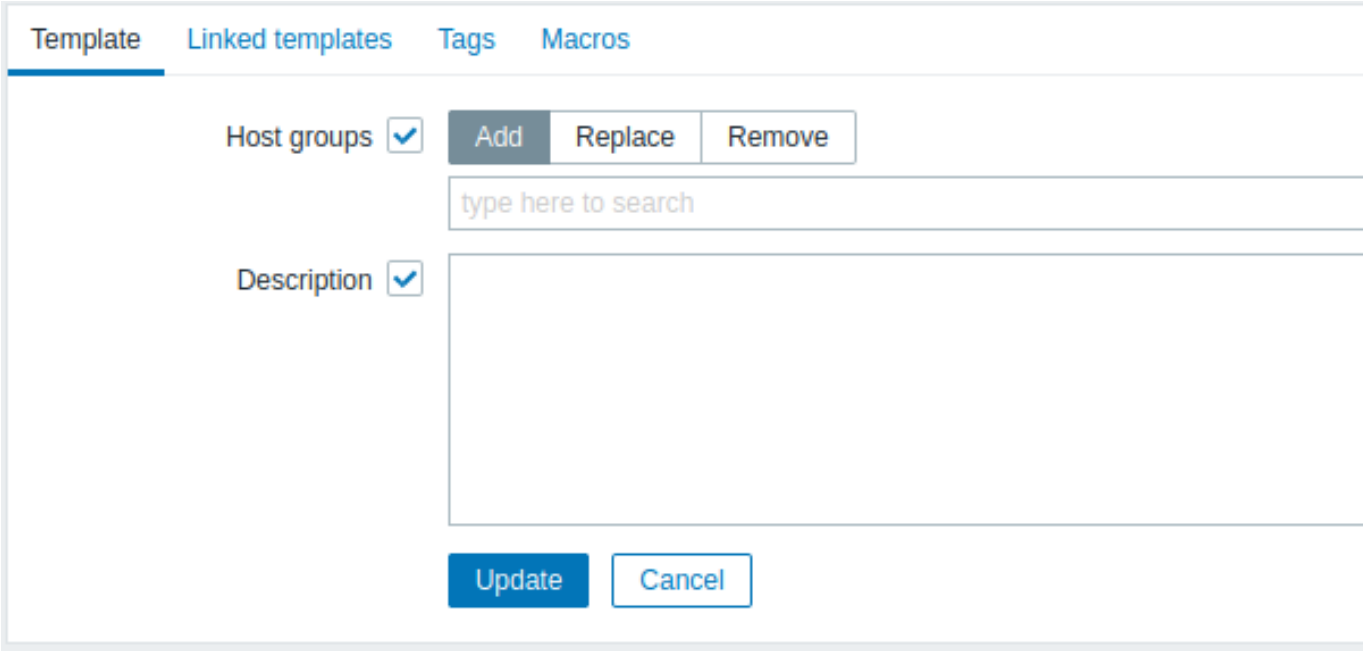
Overview

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:

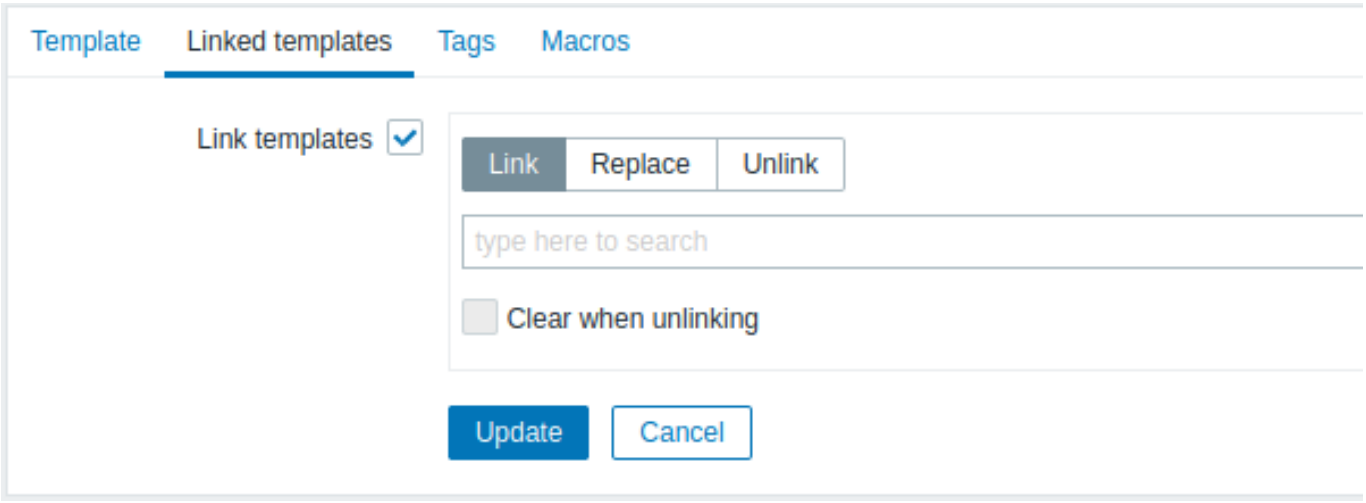
- Mark the checkboxes before the templates you want to update in the template list
- Click on *Mass update* below the list
- Navigate to the tab with required attributes (*Template*, *Linked templates* or *Tags*)
- Mark the checkboxes of any attribute to update and enter a new value for them



The following options are available when selecting the respective button for host group update:

- *Add* - allows to specify additional host groups from the existing ones or enter completely new host groups for the templates.
- *Replace* - will remove the template 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 templates.

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 it is indicated by *(new)* after the string. Just scroll down to select.

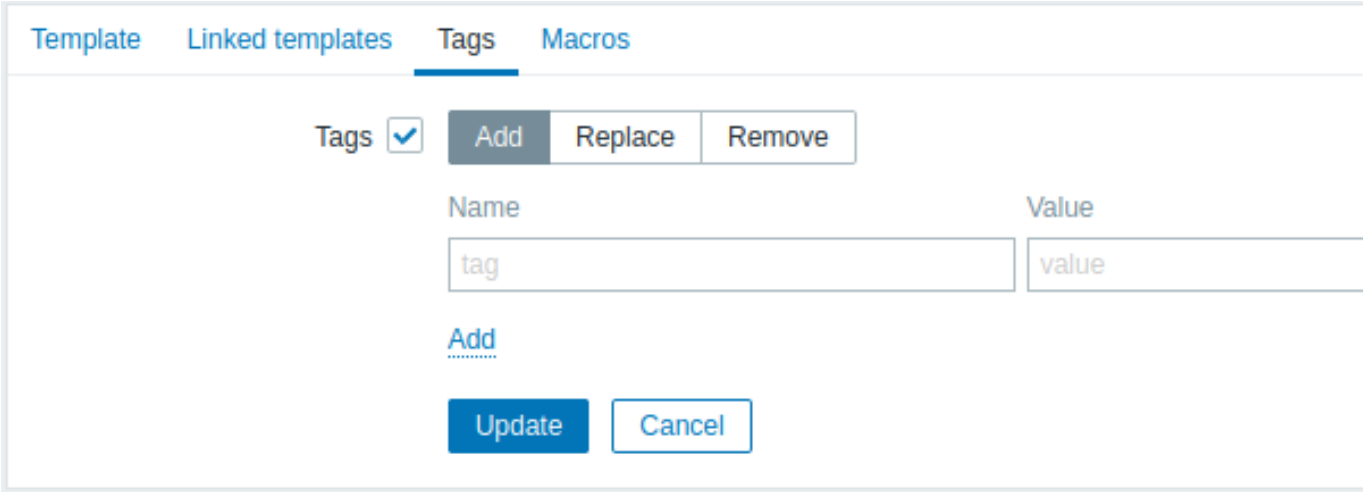


The following options are available when selecting the respective button for template linkage update:

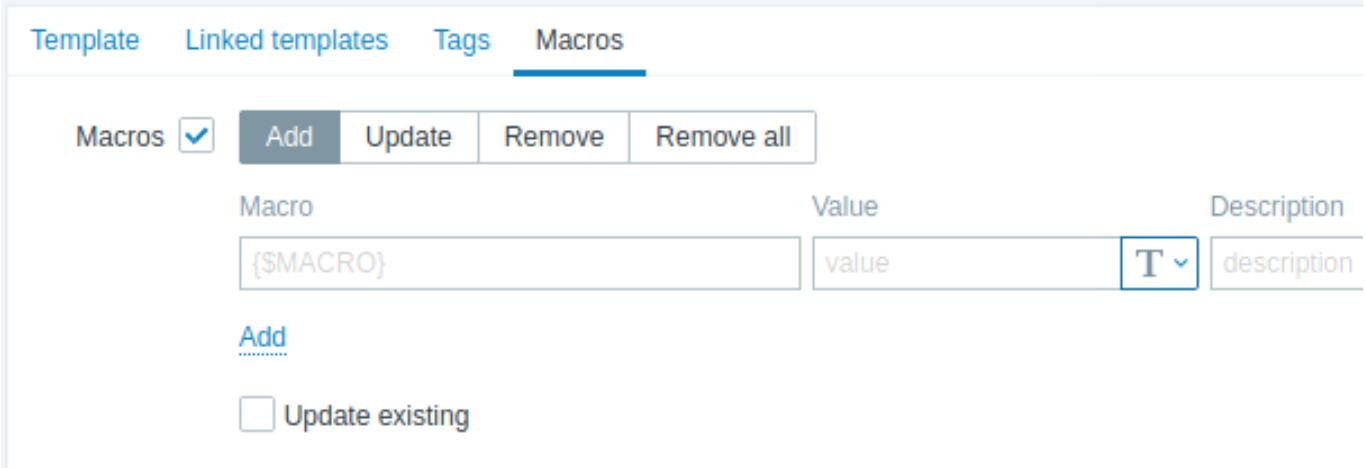
- *Link* - specify which additional templates to link.
- *Replace* - specify which templates to link while unlinking any template that was linked to the templates before.
- *Unlink* - specify which templates to unlink.

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.).



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.

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:

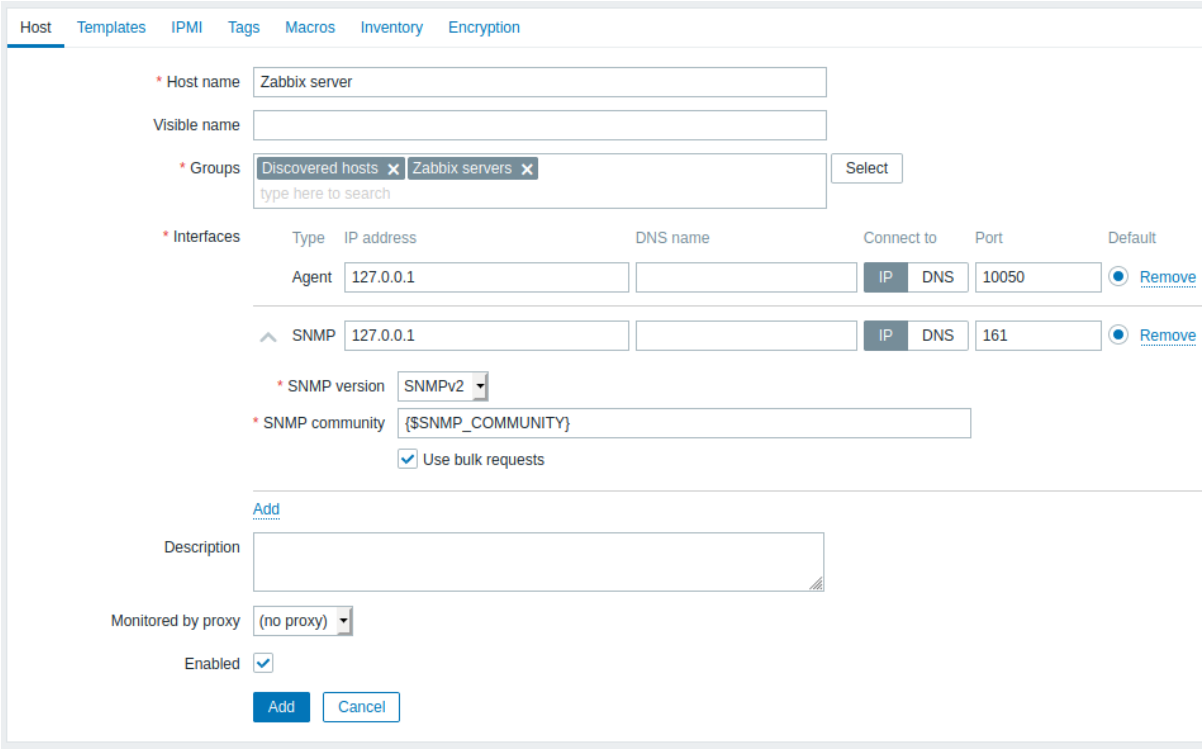
- Go to: *Configuration* → *Hosts*
- Click on *Create host* to the right (or on the host name to edit an existing host)
- Enter parameters of the host in the form

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.

Parameter	Description
Host name	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.
Visible name	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.
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.

Parameter		Description
Interfaces		Several host interface types are supported for a host: *Agent*, *SNMP*, *JMX* and *IPMI*. To add a new interface, click on *Add* in the *Interfaces* block 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 proxy		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 **Templates** tab allows you to link templates to the host. All entities (items, triggers, graphs and applications) 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:

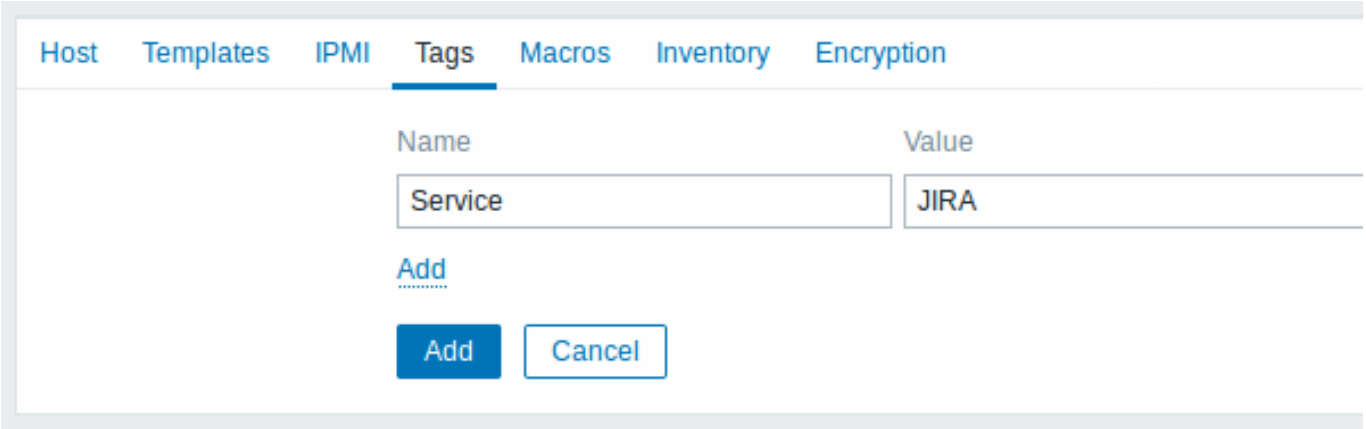
- *Unlink* - unlink the template, but preserve its items, triggers and graphs
- *Unlink and clear* - unlink the template and remove all its items, triggers and graphs

Listed template names are clickable links leading to the template configuration form.

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.

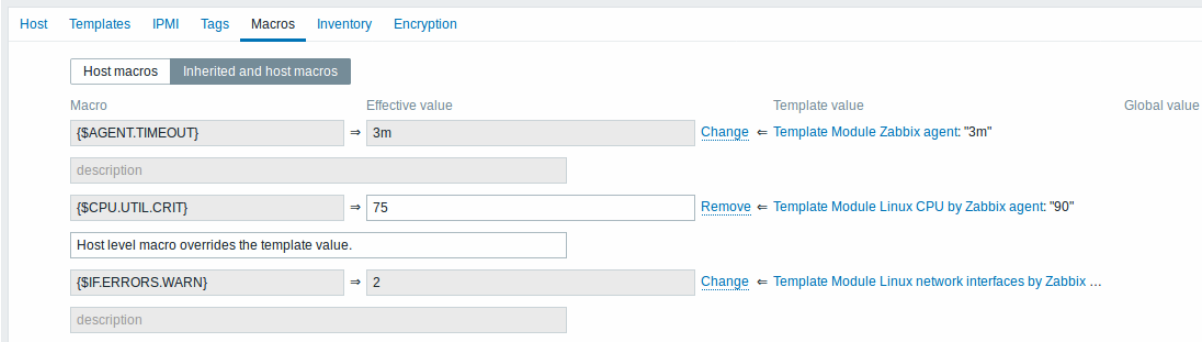
The **Tags** tab allows you to define host-level tags. All problems of this host will be tagged with the values entered here.



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 macro-value pairs. 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.

Encryption

The **Encryption** tab allows you to require encrypted connections with the host.

Parameter	Description
Connections to host	How Zabbix server or proxy connects to Zabbix agent on a host: no encryption (default), using PSK (pre-shared key) or certificate.
Connections from host	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 "No encryption".
Issuer	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.
Subject	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.
PSK identity	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.

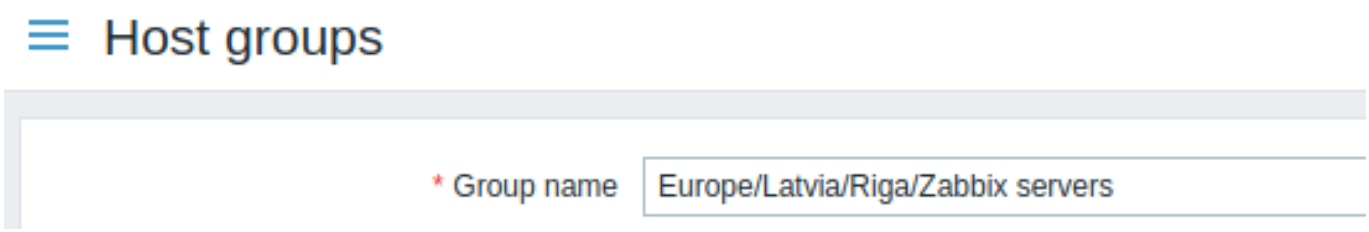
Parameter	Description
PSK	Pre-shared key (hex-string). Maximum length: 512 hex-digits (256-byte PSK) if Zabbix uses GnuTLS or OpenSSL library, 64 hex-digits (32-byte PSK) if Zabbix uses mbed TLS (PolarSSL) library. Example: 1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952

Creating a host group

Attention:
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 Group* in the upper right corner of the screen
- Enter parameters of the group in the form



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Group name	Enter a unique host group name. 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. Nested representation of host groups is supported since Zabbix 3.2.0.
Apply permissions and tag filters to all subgroups	Checkbox is available to Zabbix Super Admin users only and only when editing an existing host group. 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. This option is supported since Zabbix 3.4.0.

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[regexp,manager,short|full] - default is [all,all,full]

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*.

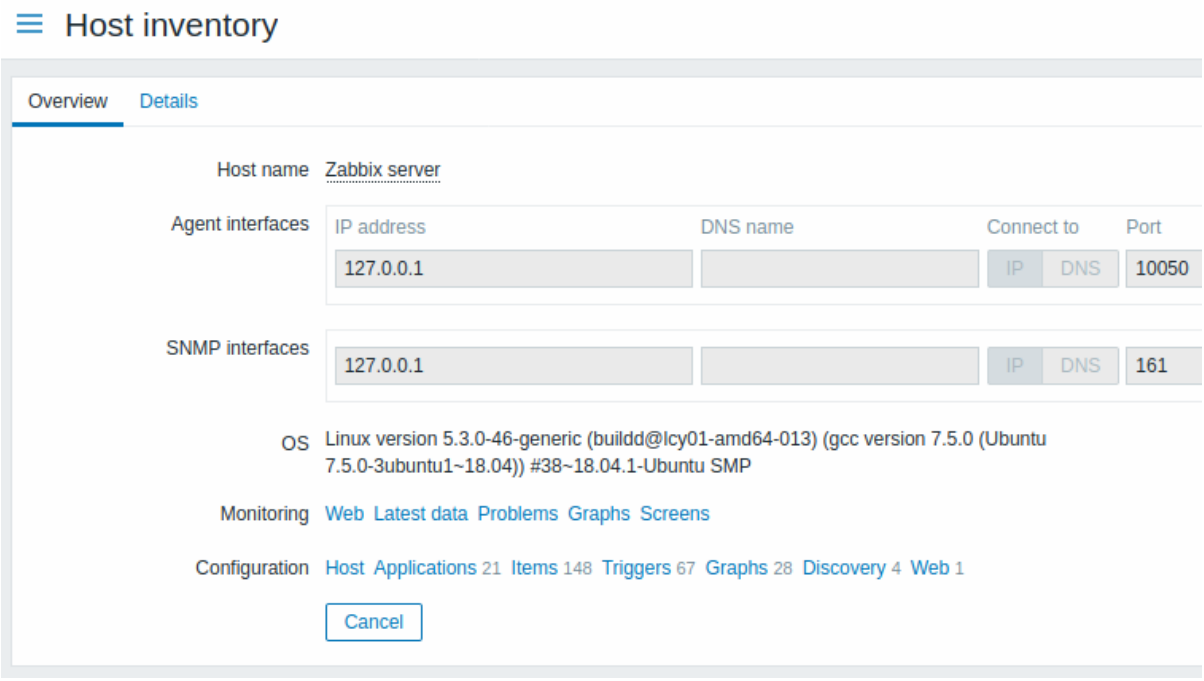
For hosts added by network discovery or autoregistration actions, it is possible to define a *Set host inventory mode* operation selecting manual or automatic mode. This operation overrides the *Default host inventory mode* setting.

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:

Parameter	Description
Host name	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.
Visible name	Visible name of the host (if defined).
Host (Agent, SNMP, JMX, IPMI) interfaces	This block provides details of the interfaces configured for the host.
OS	Operating system inventory field of the host (if defined).
Hardware	Host hardware inventory field (if defined).
Software	Host software inventory field (if defined).
Description	Host description.
Monitoring	Links to monitoring sections with data for this host: *Web*, *Latest data*, *Triggers*, *Problems*, *Graphs*, *Screens*.
Configuration	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.

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

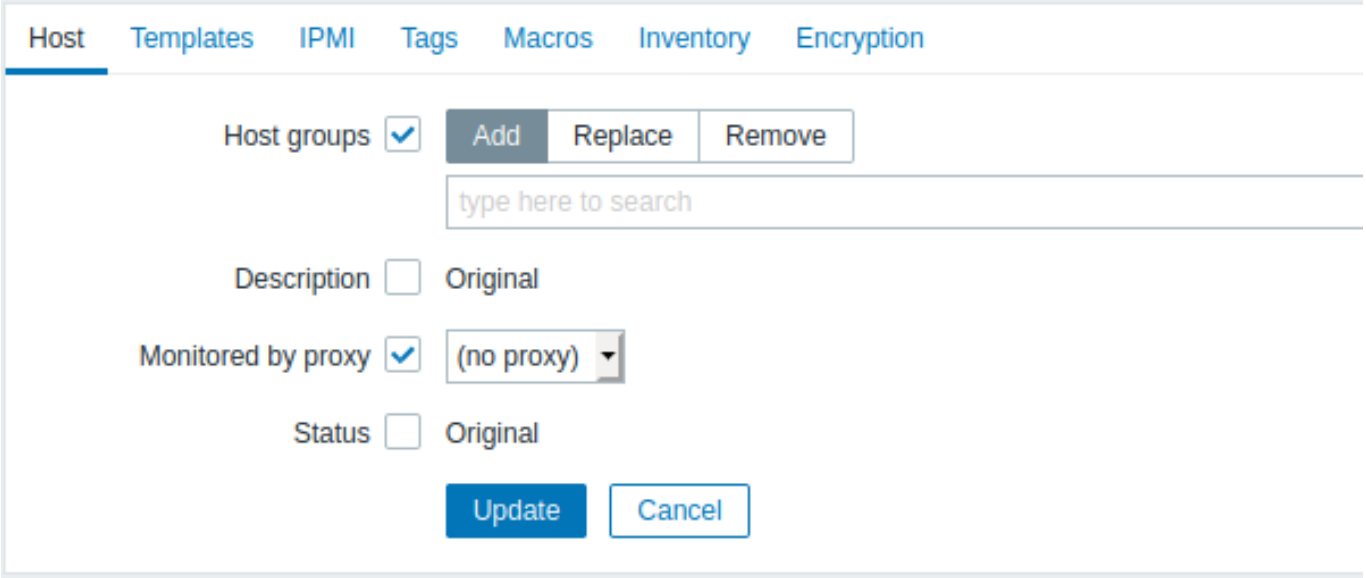
Overview

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:

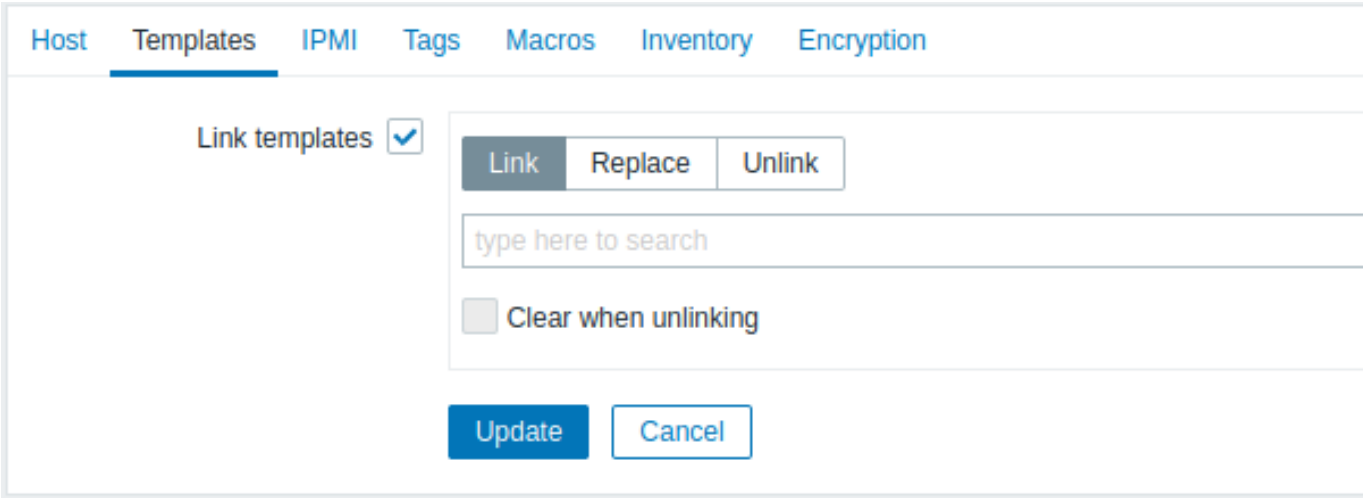
- Mark the checkboxes before the hosts you want to update in the host list
- Click on *Mass update* below the list
- Navigate to the tab with required attributes (*Host*, *Templates*, *IPMI*, *Inventory* or *Encryption*)
- Mark the checkboxes of any attribute to update and enter a new value for them



The following options are available when selecting the respective button for 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 it is indicated by *(new)* after the string. Just scroll down to select.

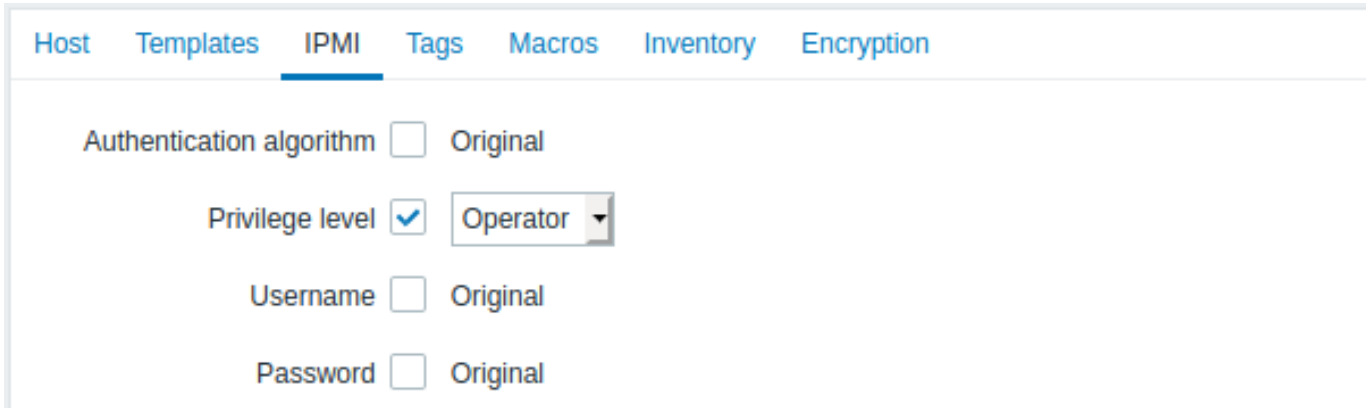


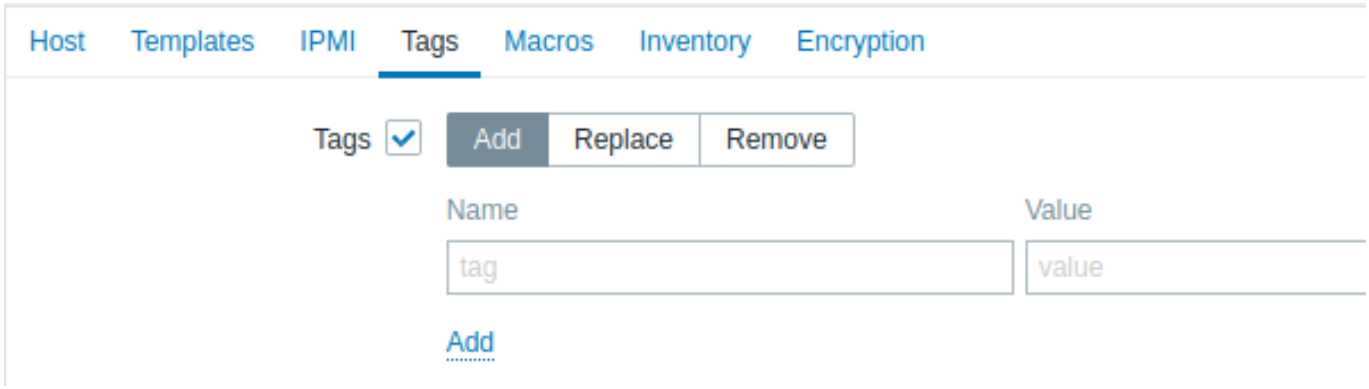
The following options are available when selecting the respective button for template linkage update:

- *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.

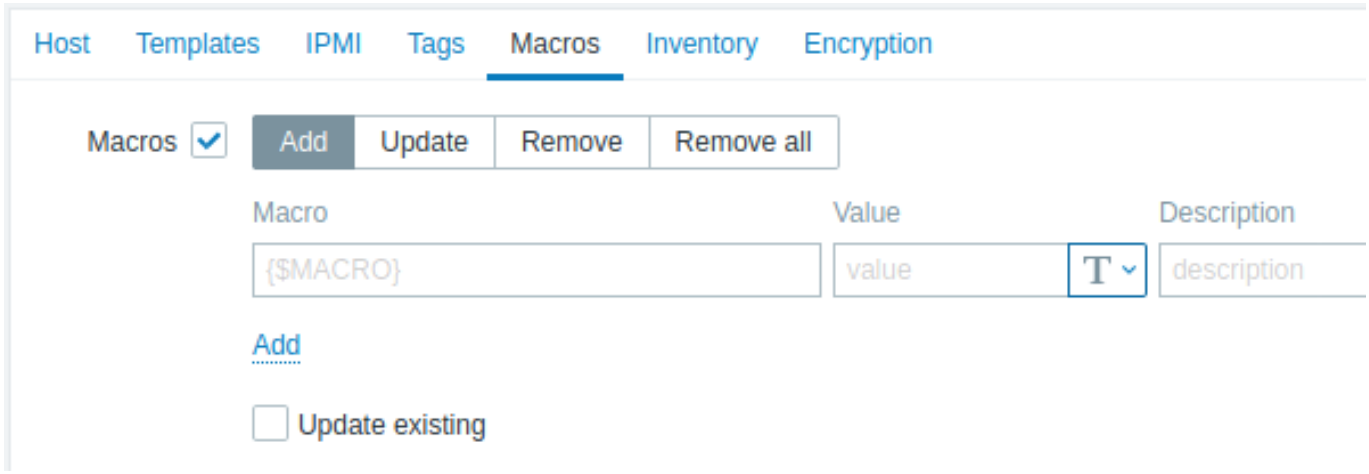
To specify the templates to link/unlink start typing the template name in the auto-complete field until a dropdown appears offering the matching templates. Just scroll down to select the required template.

The *Clear when unlinking* option will allow to not only unlink any previously linked templates, but also remove all elements inherited from them (items, triggers, etc.).



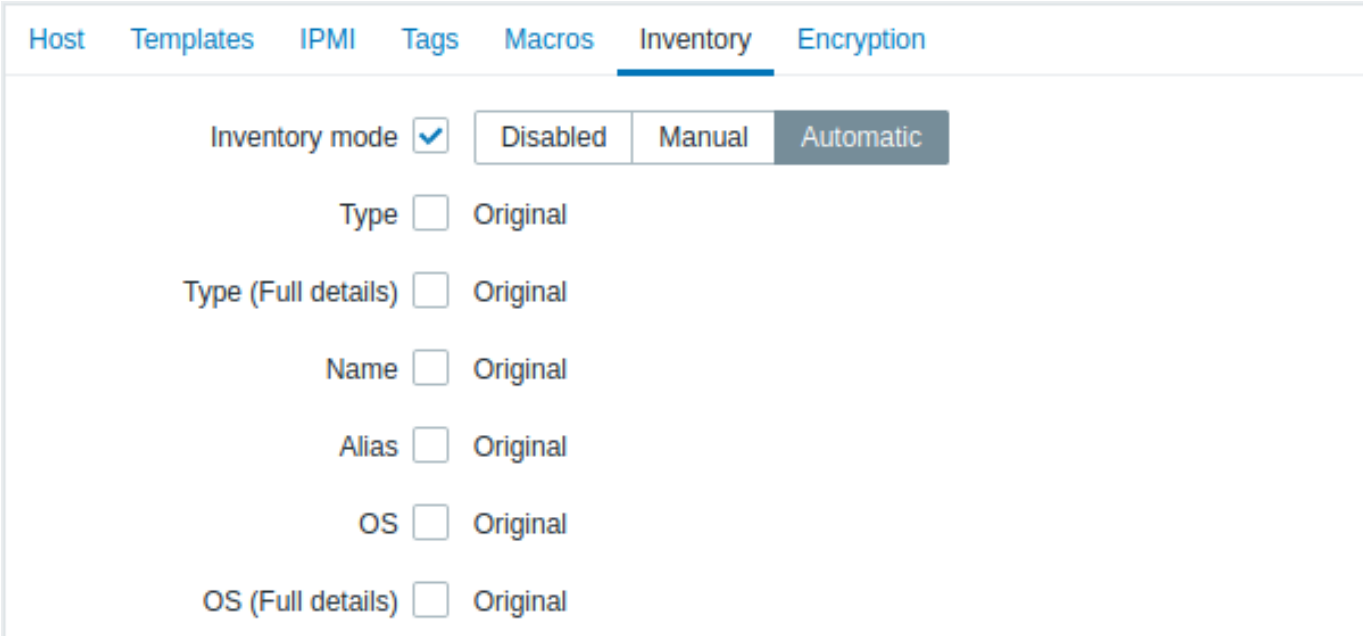


User macros, {INVENTORY.*} macros, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros are supported in tags. Note that tags with the same name but different values are not considered 'duplicates' and can be added to the same host.

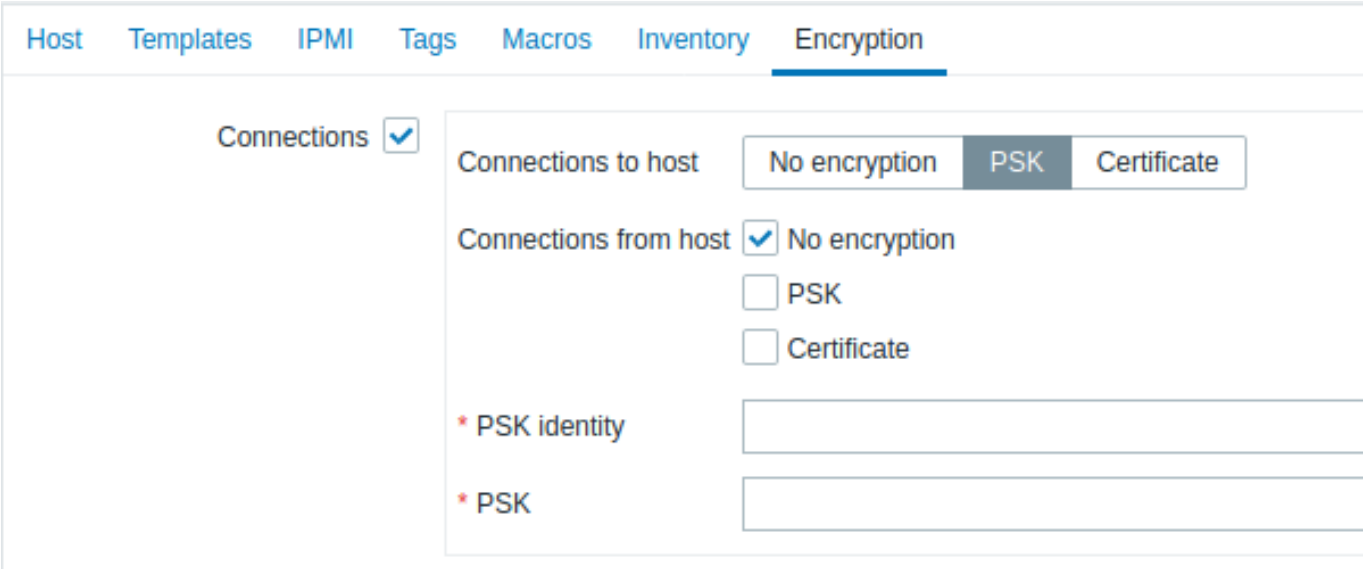


The following options are available when selecting the respective button for macros update:

- *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'.



When done with all required changes, click on *Update*. The attributes will be updated accordingly for all the selected hosts.

2 Items

Overview

An item is an individual metric.

Items are used for collecting data. Once you have configured a host, you must add items to get actual data. One way of quickly adding many items is to attach one of the predefined templates to a host. However, for optimized system performance, you may need to fine-tune the templates to have as many items and as frequent monitoring as necessary.

To specify what sort of data to collect from a host, use the item key. For example, an item with the key name **system.cpu.load** will collect processor load data, while an item with the key name **net.if.in** will collect incoming traffic information.

Additional parameters can be specified in square brackets after the key name. For example, system.cpu.load**[avg5]** will return the processor load average for the last 5 minutes, while net.if.in**[eth0]** will show incoming traffic in the interface "eth0".

Note:
See individual sections of item types for all supported item types and item keys.

Proceed to creating and configuring an item.

1 Creating an item

Overview

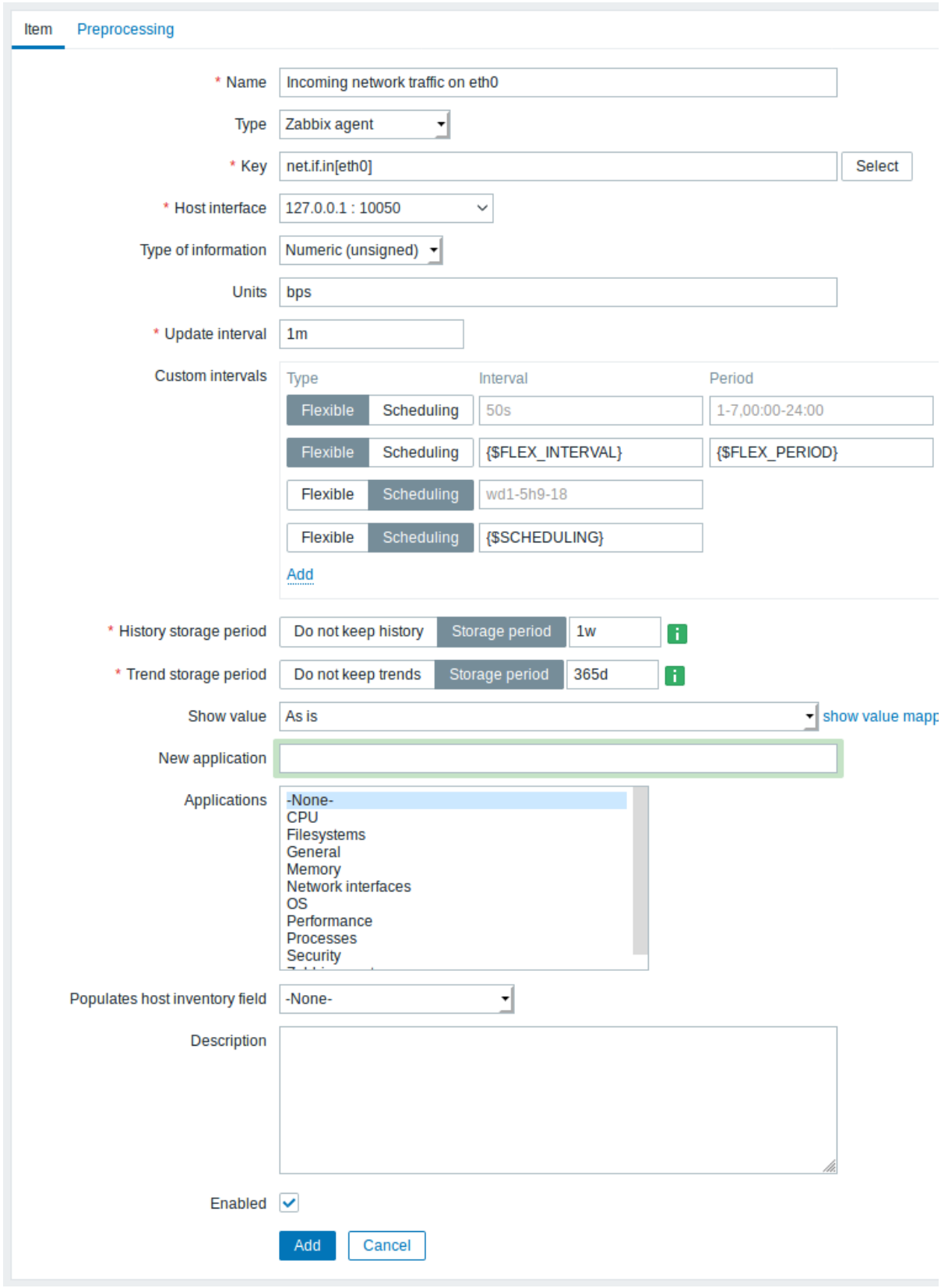
To create an item in Zabbix frontend, do the following:

- 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.

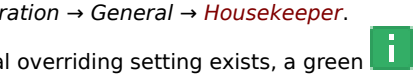


All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Item name. *Note* that the use of positional macros ($1,$2... $9 - referring to the first, second... ninth parameter of the item key) is now deprecated. For example: Free disk space on $1. If the item key is "vfs.fs.size[/,free]", the description will automatically change to "Free disk space on /"

Parameter	Description
Type	Item type. See individual item type sections.
Key	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)', 'Simple check' or 'Zabbix aggregate', the key value must be supported by Zabbix agent or Zabbix server. See also: the correct key format.
Host interface	Select the host interface. This field is available when editing an item on the host level.
Type of information	Type of data as stored in the database after performing conversions, if any. **Numeric (unsigned)** - 64-bit unsigned integer **Numeric (float)** - 64-bit floating point number 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). Receiving values in scientific notation is also supported. E.g., 1.23E+7, 1e308, 1.1E-4. **Character** - short text data **Log** - long text data with optional log related properties (timestamp, source, severity, logeventid) **Text** - long text data. See also text data limits.
Units	If a unit symbol is set, Zabbix will add postprocessing 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 **B** (byte), **Bps** (bytes per second) units, which are divided by 1024. Thus, if units are set to **B** or **Bps** Zabbix will display: 1 as 1B/1Bps 1024 as 1KB/1KBps 1536 as 1.5KB/1.5KBps Special processing is used if the following time-related units are used: **unixtime** - translated to "yyyy.mm.dd hh:mm:ss". To translate correctly, the received value must be a *Numeric (unsigned)* type of information. **uptime** - translated to "hh:mm:ss" or "N days, hh:mm:ss" For example, if you receive the value as 881764 (seconds), it will be displayed as "10 days, 04:56:04" **s** - translated to "yyy mmm ddd hhh mmm 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" 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). Will be translated to "< 1 ms" if the value is less than 0.001. *Note* that if a unit is prefixed with `!`, then no unit prefixes/processing is applied to item values. See unit conversion.

Parameter	Description
Update interval	Retrieve a new value for this item every N seconds. Maximum allowed update interval is 86400 seconds (1 day). Time suffixes are supported, e.g., 30s, 1m, 2h, 1d. User macros are supported. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. *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. *Note* that the first item poll after the item became active or after update interval change might occur earlier than the configured value. An existing passive item can be polled for value immediately by pushing the *Execute now* button.
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. Time suffixes are supported in the *Interval* field, e.g., 30s, 1m, 2h, 1d. User macros are supported. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. Scheduling is supported since Zabbix 3.0.0. *Note*: Not available for Zabbix agent active items.
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.

Parameter	Description
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.
Show value	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 be set to 0 in Unix time, representing January 1, 1970. For example, consider the following line from the Zabbix agent log file: ” 23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211).” It begins with six character positions for PID, followed by date, time, and the rest of the message. The log time format for this line would be ”ppppp:yyyyMMdd:hhmmss”. Note that ”p” and ”:” characters are placeholders and can be any characters except ”yMdhms”.
New application	Enter the name of a new application for the item.
Applications	Link item to one or more existing applications.
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. User macros are supported.
Enabled	Mark the checkbox to enable the item so it will be processed.

Note:
Item type specific fields are described on corresponding pages.

Note:
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.

Item value preprocessing

The **Preprocessing** tab allows to define transformation rules for the received values.

Testing

> **Attention:**
> To perform item testing, ensure that the system time on the server and the proxy is synchronized. In the case when the server time is behind, item testing may return an error message "The task has been expired." Having set different time zones on the server and the proxy, however, won't affect the testing result.

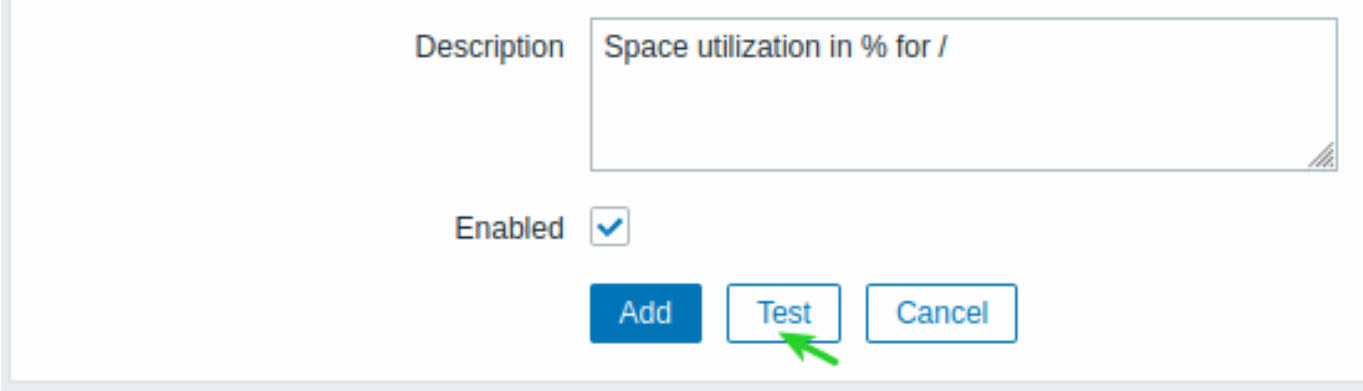
It is possible to test an item and, if configured correctly, get a real value in return. Testing can occur even before an item is saved.

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
- Zabbix aggregate
- Calculated items
- External checks
- Database monitor
- HTTP agent

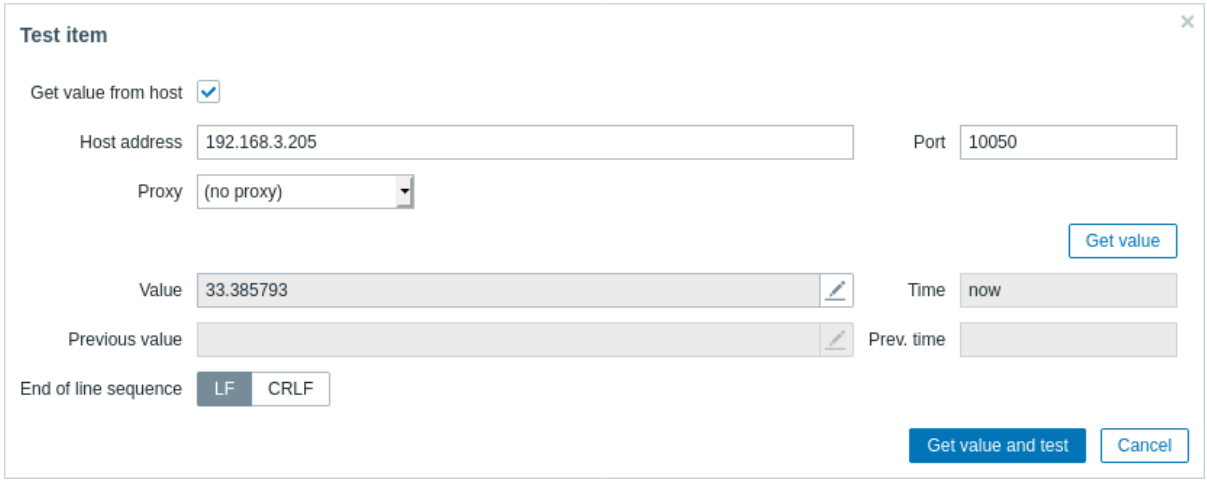
To test an item, click on the *Test* button at the bottom of the item configuration form. Note that the *Test* button will be disabled for items that cannot be tested (like active checks, excluded simple checks).



The item testing form has fields for the required host parameters (host address, port, proxy name/no proxy). 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
- The fields are disabled when not needed in the context of the item type (e.g., the host address field is disabled for calculated and aggregate items, the proxy field is 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 now	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	Test if item configuration is correct by getting a value.
Clear history and trends	Delete the item history and trends.

Delete	Delete the item.
Cancel	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:

Database	Type of information		
	Character	**Log**	**Text**
MySQL	255 characters	65536 bytes	65536 bytes
PostgreSQL	255 characters	65536 characters	65536 characters
Oracle	255 characters	65536 characters	65536 characters

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
```

Note:
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:

Database	*Limit in characters*	*Limit in bytes*
MySQL	65535	65535
Oracle Database	2048	4000
PostgreSQL	65535	not limited
SQLite (only Zabbix proxy)	65535	not limited

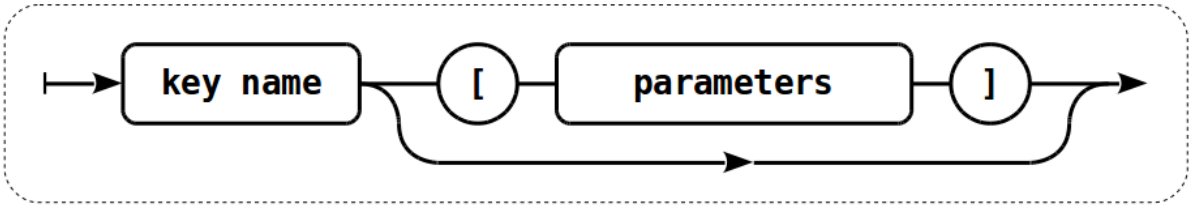
Unsupported items

An item can become unsupported if its value cannot be retrieved for some reason. Such items are still rechecked at a fixed interval, configurable in *Administration* → *General* → *Other parameters*.

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.

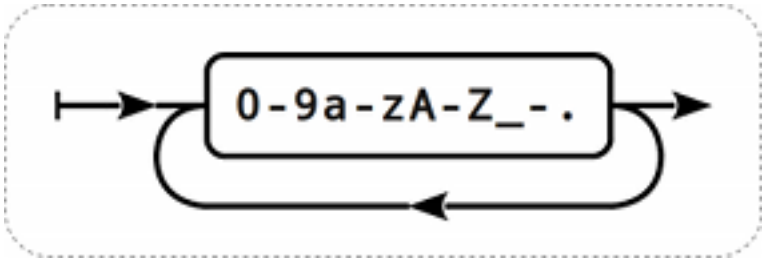
Key name

The key name itself has a limited range of allowed characters, which just follow each other. Allowed characters are:

`0-9a-zA-Z_-.`

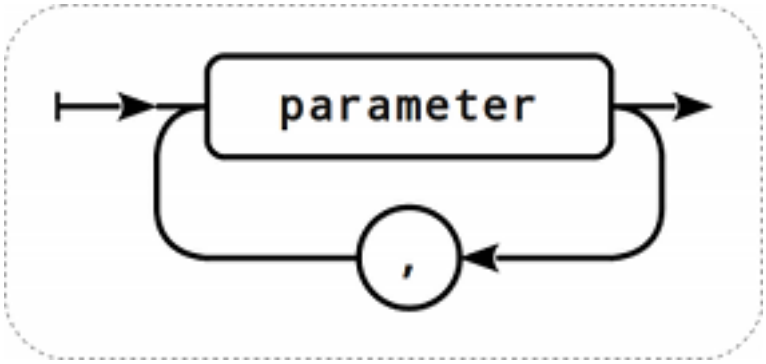
Which means:

- all numbers;
- all lowercase letters;
- all uppercase letters;
- underscore;
- dash;
- dot.

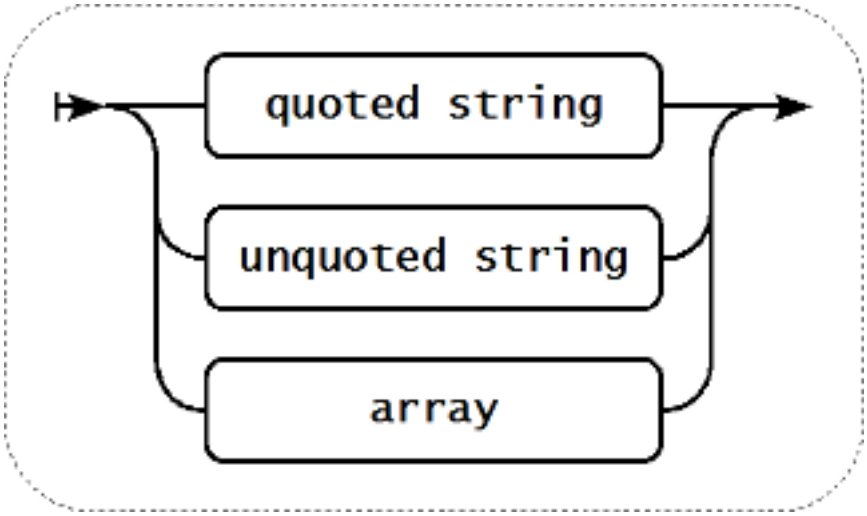


Key parameters

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 **icmpping[„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.

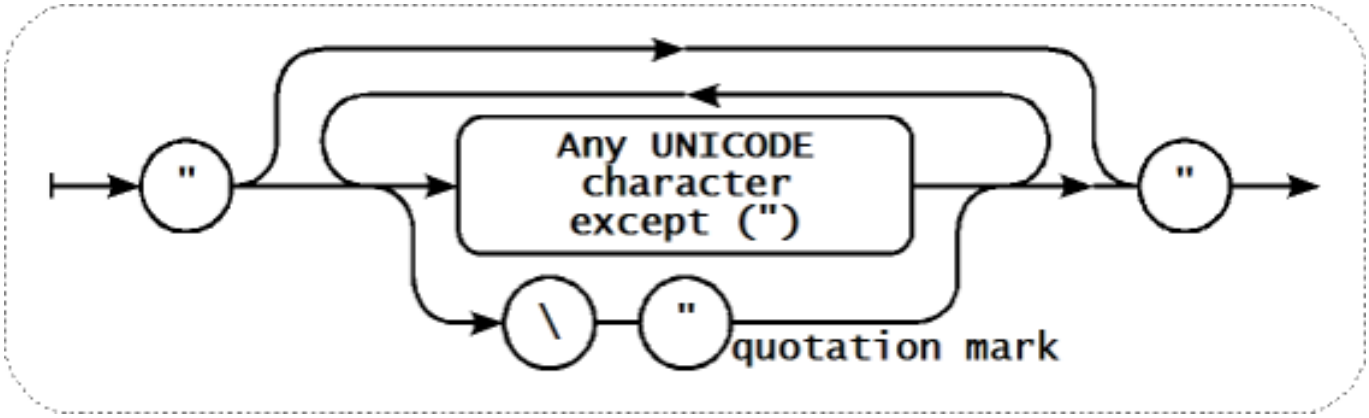
It is possible to include macros in the parameters. Those can be user macros or some of the built-in macros. To see what particular built-in macros are supported in item key parameters, search the page Supported macros for ”item key parameters”.

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.

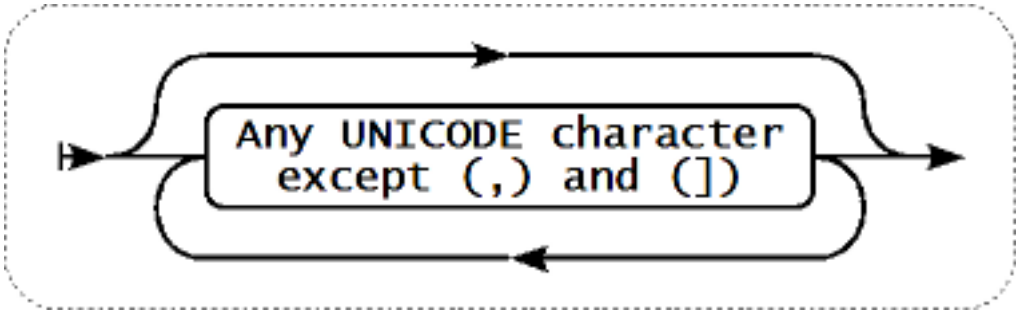
If the key parameter string contains quotation mark, this parameter has to be quoted and each quotation mark which is a part of the parameter string has to be escaped with a backslash (\) character.



Warning:
To quote item key parameters, use double quotes only. Single quotes are not supported.

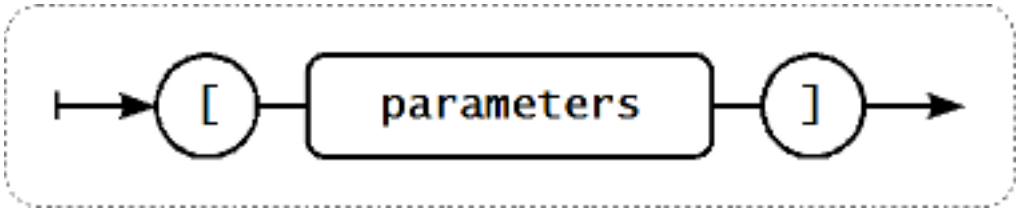
Parameter - unquoted string

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.



Attention:
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)

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:

Interval	Period	Description
10	1-5,09:00-18:00	Item will be checked every 10 seconds during working hours.
0	1-7,00:00-7:00	Item will not be checked during the night.
0	7-7,00:00-24:00	Item will not be checked on Sundays.
60	1-7,12:00-12:01	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.

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<filter>wd<filter>h<filter>m<filter>s<filter>` where:

- **md** - month days
- **wd** - week days
- **h** - hours
- **m** - minutes
- **s** – seconds

`<filter>` is used to specify values for its prefix (days, hours, minutes, seconds) and is defined as: `[<from>[-<to>]][/<step>][,<filter` where:

- `<from>` and `<to>` define the range of matching values (included). If `<to>` is omitted then the filter matches a `<from> - <from>` range. If `<from>` is also omitted then the filter matches all possible values.
- `<step>` defines the skips of the number value through the range. By default `<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 *<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:

Prefix	Description	*<from>*	*<to>*
md	Month days	1-31	1-31
wd	Week days	1-7	1-7
h	Hours	0-23	0-23
m	Minutes	0-59	0-59

Prefix	Description	*<from>*	*<to>*
s	Seconds	0-59	0-59

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:

Interval	Will be executed
m0-59	every minute
h9-17/2	every 2 hours starting with 9:00 (9:00, 11:00 ...)
m0,30 or m/30	hourly at hh:00 and hh:30
m0,5,10,15,20,25,30,35,40,45,50,55 or m/5	every five minutes
wd1-5h9	every Monday till Friday at 9:00
wd1-5h9-18	every Monday till Friday at 9:00,10:00,...,18:00
h9,10,11 or h9-11	every day at 9:00, 10:00 and 11:00
md1h9m30	every 1st day of each month at 9:30
md1wd1h9m30	every 1st day of each month at 9:30 if it is Monday
h9m/30	every day at 9:00, 9:30
h9m0-59/30	every day at 9:00, 9:30
h9,10m/30	every day at 9:00, 9:30, 10:00, 10:30
h9-10m30	every day at 9:30, 10:30
h9m10-40/30	every day at 9:10, 9:40
h9,10m10-40/30	every day at 9:10, 9:40, 10:10, 10:40
h9-10m10-40/30	every day at 9:10, 9:40, 10:10, 10:40
h9m10-40	every day at 9:10, 9:11, 9:12, ... 9:40
h9m10-40/1	every day at 9:10, 9:11, 9:12, ... 9:40
h9-12,15	every day at 9:00, 10:00, 11:00, 12:00, 15:00
h9-12,15m0	every day at 9:00, 10:00, 11:00, 12:00, 15:00
h9-12,15m0s30	every day at 9:00:30, 10:00:30, 11:00:30, 12:00:30, 15:00:30
h9-12s30	every day at 9:00:30, 9:01:30, 9:02:30 ... 12:58:30, 12:59:30
h9m/30;h10 (*API-specific syntax*)	every day at 9:00, 9:30, 10:00
h9m/30 h10 (*add this as another row in frontend*)	every day at 9:00, 9:30, 10:00

Custom intervals correlation

Keep in mind that when configuring custom intervals the value of *Update interval* is taken into account. The following table shows correlation between custom intervals and *Update interval*.

Update interval	Flexible	Scheduling	Result
0			*Error*
0	non-zero		Flexible
0		non-zero	Scheduling
non-zero			Update interval
non-zero	non-zero		Flexible
non-zero		non-zero	Update interval and Scheduling

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.

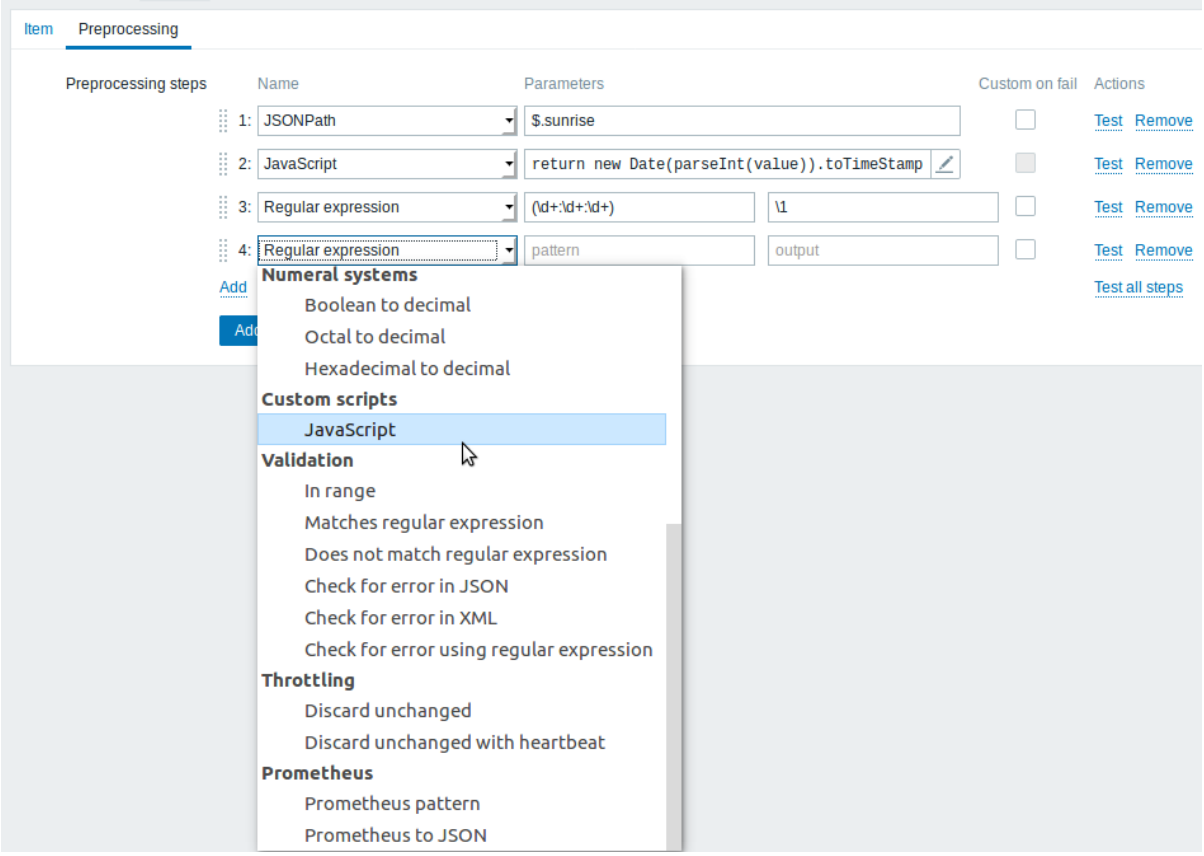
Transformations are executed in the order in which they are defined. Preprocessing is done by Zabbix server or proxy (if items are monitored by proxy).

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.



Attention:
An item will become unsupported if any of the preprocessing steps fail, unless *Custom on fail* error-handling (available for supported transformations) has been configured to discard the value or to set a specified value.

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.

Note:
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.

Type	Transformation	Description
Text		

Type	Transformation	Description
	Regular expression	Match the value to the <pattern> regular expression and replace value with <output>. 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: **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. Supported since 3.4.0. Please refer to regular expressions section for some existing examples. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
	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.
	Trim	Remove specified characters from the beginning and end of the value.
	Right trim	Remove specified characters from the end of the value.
	Left trim	Remove specified characters from the beginning of the value.
Structured data		

Type	Transformation	Description

	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></do`
		`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></do`
		Note that namespaces are not supported.
		Supported since 3.4.0.
		If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.

	JSON Path	Extract value or fragment from JSON data using JSONPath functionality. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
	CSV to JSON	Convert CSV file data into JSON format. For more information, see: CSV to JSON preprocessing. Supported since 4.4.0.
Arithmetic		

Type	Transformation	Description
	Custom multiplier	Multiply the value by the specified integer or floating-point value. Use this option to convert values received in KB, MBps, etc into B, Bps. Otherwise Zabbix cannot correctly set prefixes (K, M, G etc). *Note* that if the item type of information is *Numeric (unsigned)*, incoming values with a fractional part will be trimmed (i.e. '0.9' will become '0') before the custom multiplier is applied. Supported: scientific notation, for example, 1e+70 (since version 2.2); user macros and LLD macros (since version 4.0); strings that include macros, for example, `{#MACRO}e+10`, `{$MACRO1}e+{$MACRO2}` (since version 5.0.7) The macros must resolve to an integer or a floating-point number. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
Change		
	Simple change	Calculate the difference between the current and previous value. Evaluated as **value**-**prev_value**, where *value* - current value; *prev_value* - previously received value This setting can be useful to measure 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. Only one change operation per item is allowed. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.

Type	Transformation	Description
	Change per second	Calculate the value change (difference between the current and previous value) speed per second. Evaluated as (**value**-**prev_value**)/(**time**-**prev_time**), where *value* - current value; *prev_value* - previously received value; *time* - current timestamp; *prev_time* - timestamp of previous value. This setting is useful for collecting the 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, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
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. Following values are supported since 4.0.0: ok, master, err, slave. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.

Type	Transformation	Description
	Octal to decimal	Convert the value from octal format to decimal. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
	Hexadecimal to decimal	Convert the value from hexadecimal format to decimal. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
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, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
	Matches regular expression	Specify a regular expression that a value must match. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.

Type	Transformation	Description
	Does not match regular expression	Specify a regular expression that a value must not match. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
	Check for error in JSON	Check for an application-level error message located at JSONpath. 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, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
	Check for error in XML	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, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.

Type	Transformation	Description
	Check for error using a regular expression	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. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
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. Trigger 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.

Type	Transformation	Description
	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. Trigger 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.
Prometheus	*Prometheus pattern*	Use the following query to extract required data from Prometheus metrics. See Prometheus checks for more details.
	Prometheus to JSON	Convert required Prometheus metrics to JSON. See Prometheus checks for more details.

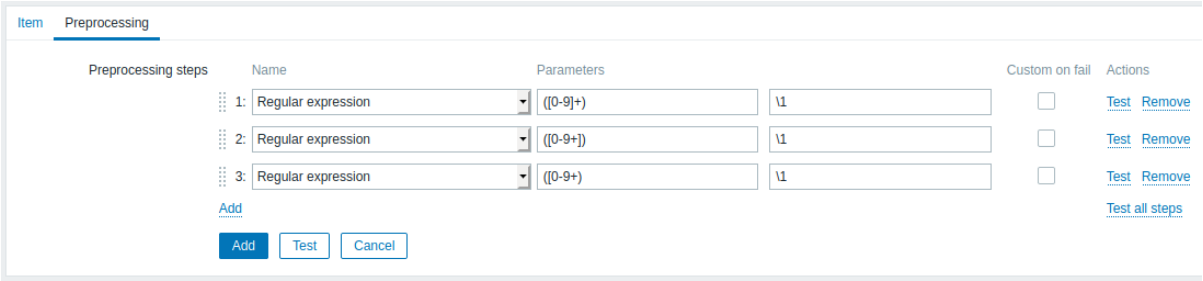
Attention:
For change and throttling preprocessing steps, a previous value is required to calculate/compare to the new value. Previous values are handled by the preprocessing manager and are reset when a change to the preprocessing steps configuration is made or when the Zabbix server/proxy is restarted. As a result of a previous value reset:

- 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.

Note:
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.

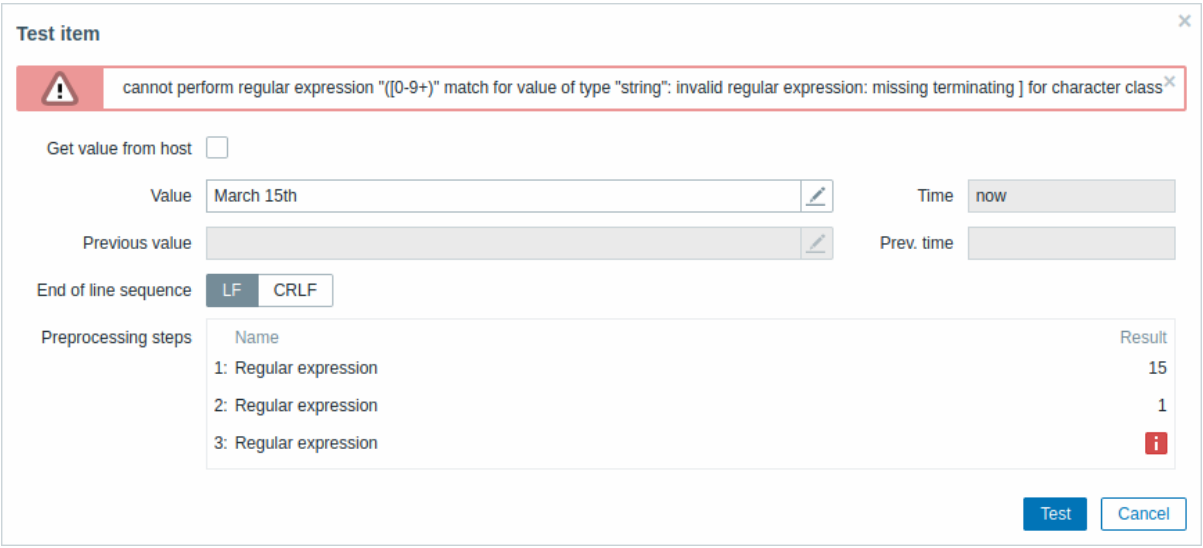


It is possible to test:

- 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



Parameter	Description
Get value from host	If you want to test a hypothetical value, leave this checkbox unmarked. See also: Testing real value.
Value	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.
Time	Time of the input value is displayed: `now` (read-only).
Previous value	Enter a previous input value to compare to. Only for *Change* and *Throttling* preprocessing steps.
Previous time	Enter the previous input value time to compare to. Only for *Change* and *Throttling* preprocessing steps. The default value is based on the 'Update interval' field value of the item (if '1m', then this field is filled with `now-1m`). If nothing is specified or the user has no access to the host, the default is `now-30s`.
Macros	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.
End of line sequence	Select the end of line sequence for multiline input values: **LF** - LF (line feed) sequence **CRLF** - CRLF (carriage-return line-feed) sequence.

Parameter	Description
Preprocessing steps	Preprocessing steps are listed; the testing result is displayed for each step after the *Test* button is clicked. If the step failed in testing, an error icon is displayed. The error description is displayed on mouseover. In case "Custom on fail" is specified for the step and that action is performed, a new line appears right after the preprocessing test step row, showing what action was done and what outcome it produced (error or value).
Result	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). The type of conversion to the value type of the item is also displayed, for example `Result converted to Numeric (unsigned)`.

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 a 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.

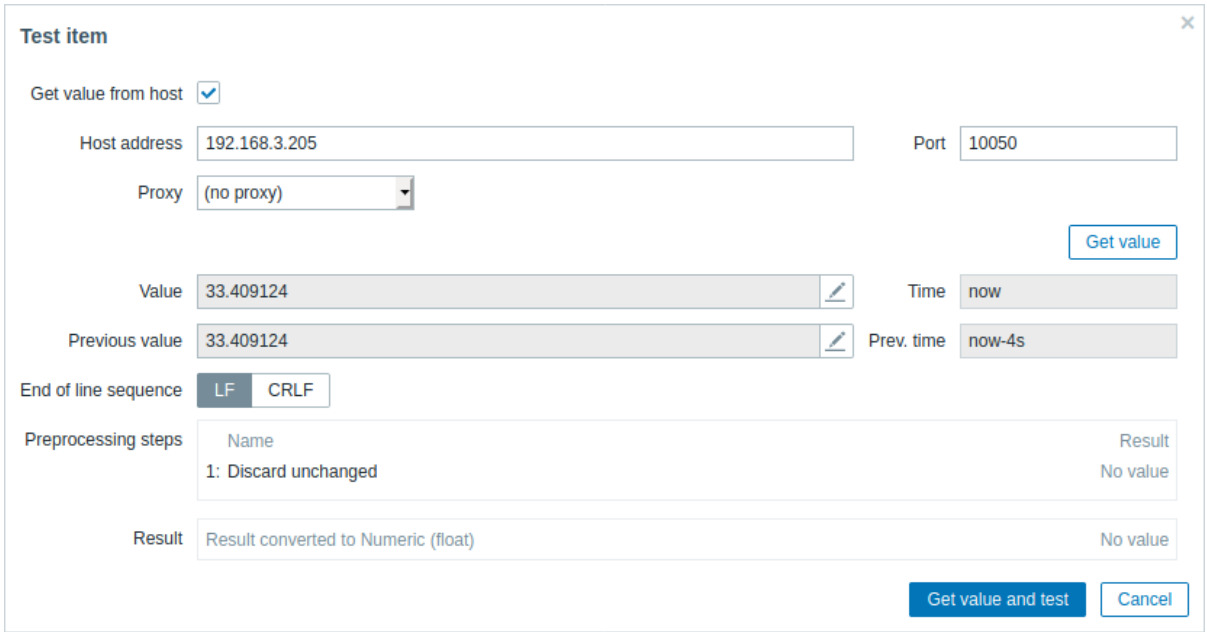
The testing is done by Zabbix server. The frontend sends a corresponding request to the server and waits for the result. The request contains the input value and preprocessing steps (with expanded user macros). For *Change* and *Throttling* steps, an optional previous value and time can be specified. 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). 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
  - The field is disabled when not needed in the context of the item type (e.g. the host address field is disabled for calculated and aggregate items, the proxy field is 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:

Parameter	Description
Get value from host	Mark this checkbox to get a real value from the host.
Host address	Enter the host address. This field is automatically filled by the address of the item host interface.
Port	Enter the host port. This field is automatically filled by the port of item host interface.
Proxy	Specify the proxy if the host is monitored by a proxy. This field is automatically filled by the proxy of the host (if any).

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
```

Attention:
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

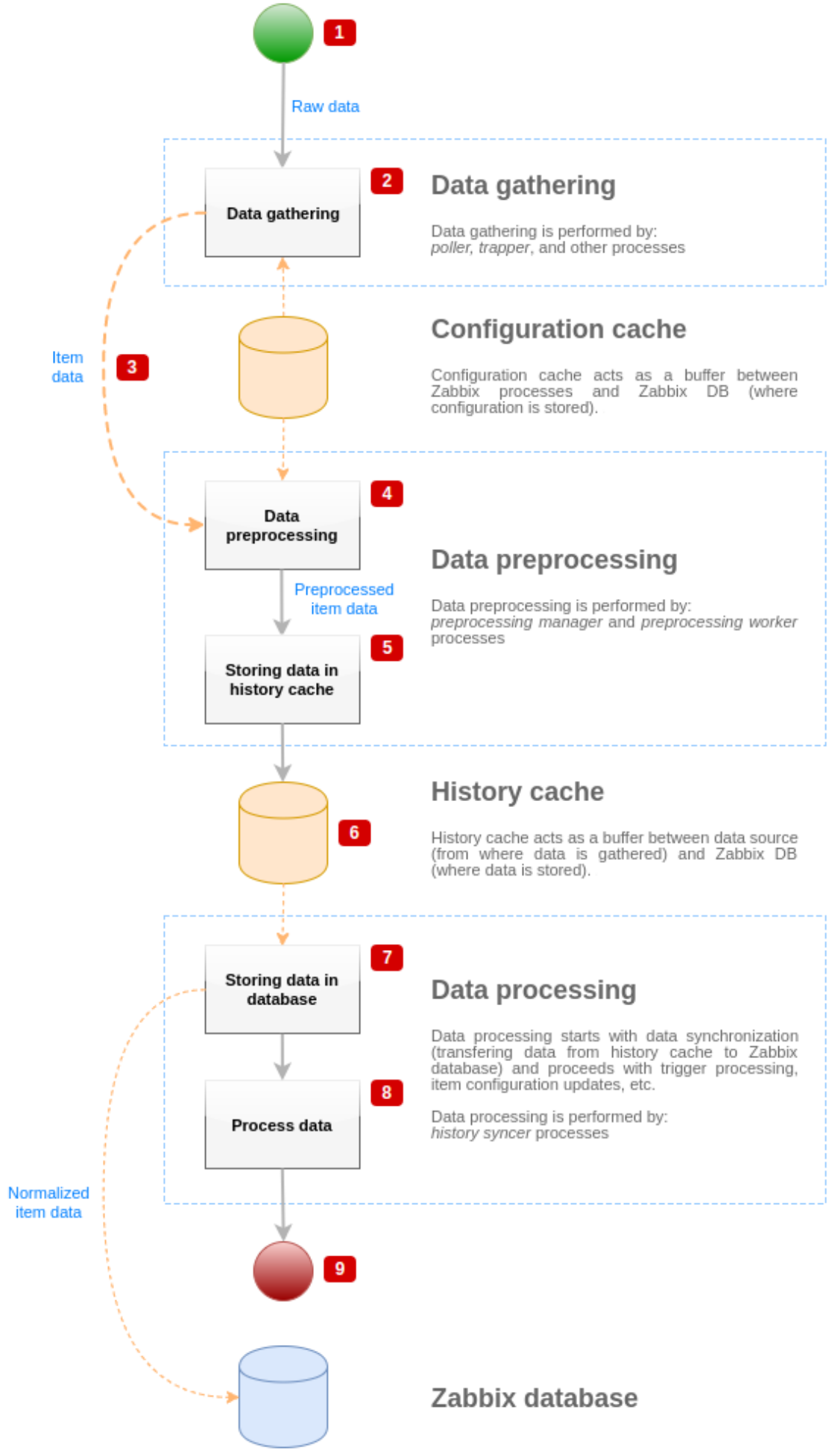
Overview

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
Data gathering
2
Data gathering
Data gathering is performed by:
poller, trapper, and other processes
Item data
3
Configuration cache
Configuration cache acts as a buffer between Zabbix processes and Zabbix DB (where configuration is stored).
Data preprocessing
4
Data preprocessing
Data preprocessing is performed by:
preprocessing manager and preprocessing worker processes
Preprocessed item data
Storing data in history cache
5
6
History cache
History cache acts as a buffer between data source (from where data is gathered) and Zabbix DB (where data is stored).
Storing data in database
7
Data processing
Data processing starts with data synchronization (transfering data from history cache to Zabbix database) and proceeds with trigger processing, item configuration updates, etc.
Data processing is performed by:
history syncer processes
Process data
8
Normalized item data
9
Zabbix database

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.

Note:
Item can change its state to NOT SUPPORTED while preprocessing is performed if any of preprocessing steps fail.

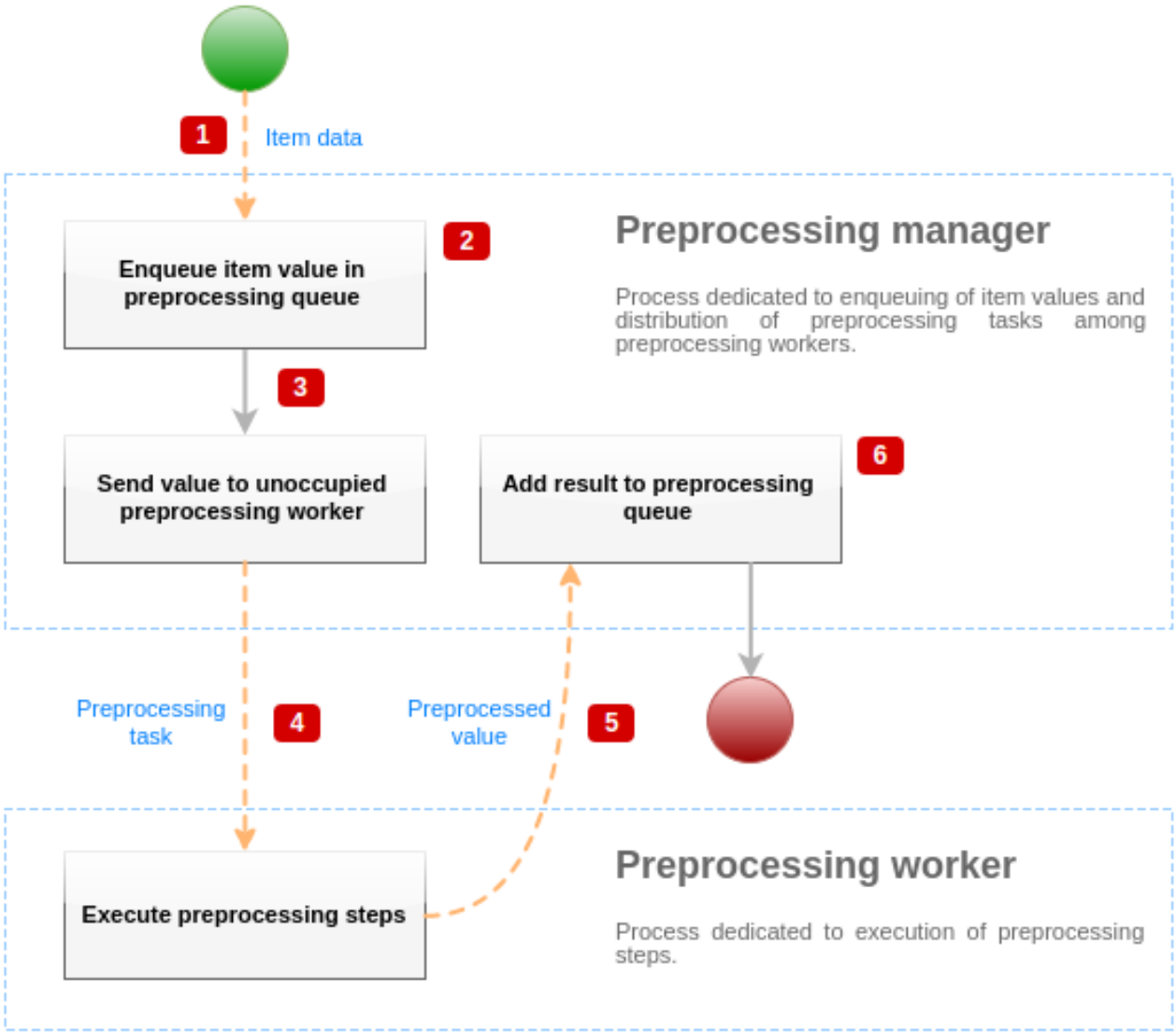
- History data from local data cache of preprocessing manager is being flushed into history cache.
- At this point data flow stops until the next synchronization of history cache (when history syncer process performs data synchronization).
- Synchronization process starts with data normalization storing data in Zabbix database. Data normalization performs conversions to desired item type (type defined in item configuration), including truncation of textual data based on predefined 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.

Note:
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.

Note:
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.

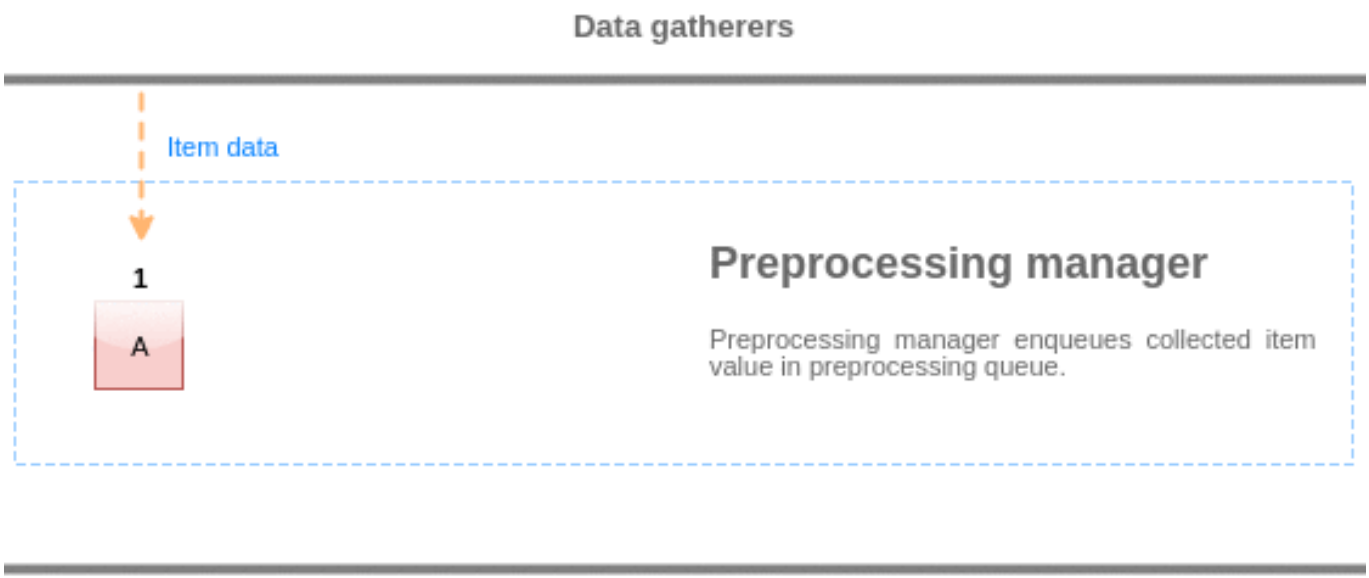
- Dependent item receives value that is not present in master item history. The use case is very similar to the previous one, except for the master item type. For example, if `CHAR` type is used for master item, then master item value will be truncated at the history synchronization phase, while dependent items will receive their value from the initial (not truncated) value of master item.

Preprocessing queue

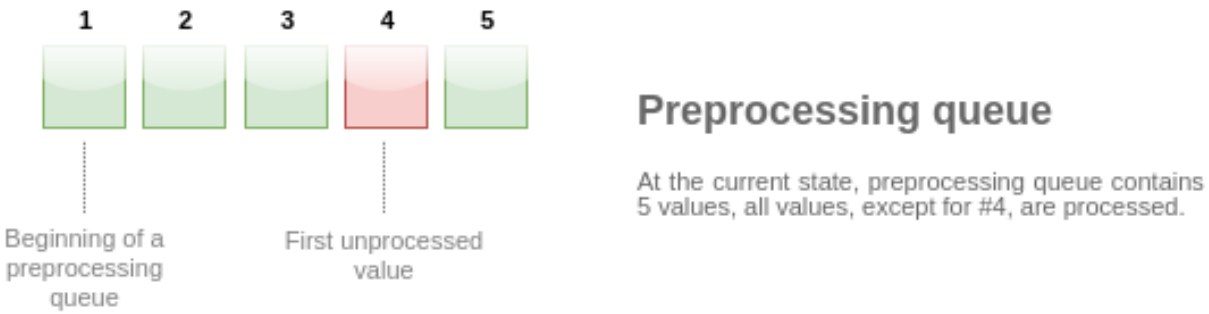
Preprocessing queue is a FIFO data structure that stores values preserving the order in which values are reviewed by preprocessing manager. There are multiple exceptions to FIFO logic:

- 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:



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:



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.

Warning:
Too many data gathering processes (pollers, unreachable 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:

Path	Description
`$.object.name`	Return the object.name contents.
`$.object['name']`	Return the object.name contents.
`$.object.['name']`	Return the object.name contents.
`$["object"]['name']`	Return the object.name contents.
`$.['object'].["name"]`	Return the object.name contents.
`$.object.history.length()`	Return the number of object.history array elements.
`$[?(@.name == 'Object')].price.first()`	Return the price field of the first object with name 'Object'.
`$[?(@.name == 'Object')].history.first().length()`	Return the number of history array elements of the first object with name 'Object'.
`$[?(@.price > 10)].length()`	Return the number of objects with price being greater than 10.

See also: Escaping special characters from LLD macro values in JSONPath.

Supported segments

Segment	Description
`<name>`	Match object property by name.
`*`	Match all object properties.
`['<name>']`	Match object property by name.
`['<name>', '<name>', ...]`	Match object property by any of the listed names.
`[<index>]`	Match array element by the index.
`[<number>, <number>, ...]`	Match array element by any of the listed indexes.
`[*]`	Match all object properties or array elements.
`[<start>:<end>]`	Match array elements by the defined range: **<start>** - 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. **<end>** - 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.

Segment	Description
`[?(<expression>)]`	Match objects/array elements by applying a filter expression.

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

The filter expression is an arithmetical expression in infix notation.

Supported operands:

Operand	Description	Example
`"<text>"` `'<text>'`	Text constant.	'value: \'1\'" "value: '1'"
`<number>`	Numeric constant supporting scientific notation.	123
`<jsonpath starting with $>`	Value referred to by the JSONPath from the input document root node; only definite paths are supported.	$.object.name
`<jsonpath starting with @>`	Value referred to by the JSONPath from the current object/element; only definite paths are supported.	@.name

Supported operators:

Operator	Type	Description	Result
-	binary	Subtraction.	Number.
+	binary	Addition.	Number.
/	binary	Division.	Number.
*	binary	Multiplication.	Number.
==	binary	Is equal to.	Boolean (1 or 0).
!=	binary	Is not equal to.	Boolean (1 or 0).
	binary	Is less than.	Boolean (1 or 0).
<=	binary	Is less than or equal to.	Boolean (1 or 0).
>	binary	Is greater than.	Boolean (1 or 0).
>=	binary	Is greater than or equal to.	Boolean (1 or 0).
=~	binary	Matches regular expression.	Boolean (1 or 0).
!	unary	Boolean not.	Boolean (1 or 0).
||	binary	Boolean or.	Boolean (1 or 0).
&&	binary	Boolean and.	Boolean (1 or 0).

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:

Function	Description	Input	Output
`avg`	Average value of numbers in input array.	Array of numbers.	Number.
`min`	Minimum value of numbers in input array.	Array of numbers.	Number.
`max`	Maximum value of numbers in input array.	Array of numbers.	Number.
`sum`	Sum of numbers in input array.	Array of numbers.	Number.
`length`	Number of elements in input array.	Array.	Number.

Function	Description	Input	Output
`first`	The first array element.	Array.	A JSON construct (object, array, value) depending on input array contents.

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 in 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

```
{
  "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"
  ]
}
```


JSONPath	Type	Result	Comments
`$.filters.price`	definite	10	
`$.filters.category`	definite	fiction	
`$.filters['no filters']`	definite	no ”filters”	
`$.filters`	definite	{ ”price”: 10, ”category”: ”fiction”, ”no filters”: ”no \”filters\”” }	
`$.books[1].title`	definite	Sword of Honour	
`$.books[-1].author`	definite	J. R. R. Tolkien	
`$.books.length()`	definite	4	
`$.tags[:]`	indefinite	[”a”, ”b”, ”c”, ”d”, ”e”]	

JSONPath	Type	Result	Comments
`$.tags[2:]`	indefinite	[”c”, ”d”, ”e”]	
`$.tags[:3]`	indefinite	[”a”, ”b”, ”c”]	
`$.tags[1:4]`	indefinite	[”b”, ”c”, ”d”]	
`$.tags[-2:]`	indefinite	[”d”, ”e”]	
`$.tags[:-3]`	indefinite	[”a”, ”b”]	
`$.tags[:-3].length()`	definite	2	
`$.books[0, 2].title`	indefinite	[”Sayings of the Century”, ”Moby Dick”]	
`$.books[1]['author', "title"]`	indefinite	[”Evelyn Waugh”, ”Sword of Honour”]	
`$..id`	indefinite	[1, 2, 3, 4]	
`$.services..price`	indefinite	[5, 154.99, 46, 24.5, 99.49]	
`$.books[?(@.id == 4 - 0.4 * 5)].title`	indefinite	[”Sword of Honour”]	This query shows that arithmetical operations can be used in queries. Of course this query can be simplified to `$.books[?(@.id == 2)].title`
`$.books[?(@.id == 2 || @.id == 4)].title`	indefinite	[”Sword of Honour”, ”The Lord of the Rings”]	
`$.books[?(!(@.id == 2))].title`	indefinite	[”Sayings of the Century”, ”Moby Dick”, ”The Lord of the Rings”]	
`$.books[?(@.id != 2)].title`	indefinite	[”Sayings of the Century”, ”Moby Dick”, ”The Lord of the Rings”]	
`$.books[?(@.title =~ " of ")].title`	indefinite	[”Sayings of the Century”, ”Sword of Honour”, ”The Lord of the Rings”]	
`$.books[?(@.price > 12.99)].title`	indefinite	[”The Lord of the Rings”]	
`$.books[?(@.author > "Herman Melville")].title`	indefinite	[”Sayings of the Century”, ”The Lord of the Rings”]	
`$.books[?(@.price > $.filters.price)].title`	indefinite	[”Sword of Honour”, ”The Lord of the Rings”]	
`$.books[?(@.category == $.filters.category)].title`	indefinite	[”Sword of Honour”,”Moby Dick”,”The Lord of the Rings”]	

JSONPath	Type	Result	Comments

`$..[?(@.id)]`	indefinite	[
		{
		”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
		}
		]	

`$.services..[?(@.price > 50)].description`	indefinite	’[”Printing and assembling book in A5 format”, ”Rebinding torn book”]	
`$..id.length()`	definite	4	
`$.books[?(@.id == 2)].title.first()`	definite	Sword of Honour	
`$..tags.first().length()`	definite	5	$..tags is indefinite path, so it returns an array of matched elements - [[”a”, ”b”, ”c”, ”d”, ”e”]], first() returns the first element - [”a”, ”b”, ”c”, ”d”, ”e”] and finally length() calculates its length - 5.
`$.books[*].price.min()`	definite	8.95	
`$..price.max()`	definite	154.99	
`$.books[?(@.category == "fiction")].price.avg()`	definite	14.99	
`$.books[?(@.category == $.filters.xyz)].title`	indefinite		A query without match returns NULL for definite and indefinite paths.
`$.services[?(@.active=="true")].servicegroup`	indefinite	[1000,1001]	Text constants must be used in boolean value comparisons.
`$.services[?(@.active=="false")].servicegroup`	indefinite	[1002]	Text constants must be used in boolean value comparisons.

JSONPath	Type	Result	Comments
`$.services[?(@.servicegroup==default")]~.first()`		restoration	

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:

JSONPath	LLD macro value	After substitution
$.[?(@.value == ”{#MACRO}”)]	special ”value”	$.[?(@.value == ”special \”value\””)]
	c:\temp	$.[?(@.value == ”c:\\temp”)]
	a\\b	$.[?(@.value == ”a\\\\b”)]

When used in the expression the macro that may have special characters should be enclosed in double quotes:

JSONPath	LLD macro value	After substitution	Result
$.[?(@.value == ”{#MACRO}”)]	special ”value”	$.[?(@.value == ”special \”value\””)]	OK
$.[?(@.value == {#MACRO})]		$.[?(@.value == special \”value\”)]	**Bad JSONPath expression**

When used in the path the macro that may have special characters should be enclosed in square brackets **and** double quotes:

JSONPath	LLD macro value	After substitution	Result
$.[”{#MACRO}”].value	c:\temp	$.[”c:\\temp”].value	OK
$.{#MACRO}.value		$.c:\\temp.value	**Bad JSONPath expression**

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 (10MB before Zabbix 5.0.9).

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.

> **Note:**
> 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.

Method	Description
`Log(loglevel, message)`	Writes <message> into Zabbix log using <loglevel> log level (see configuration file DebugLevel parameter).

Example:

```
Zabbix.Log(3, "this is a log entry written with 'Warning' log level")
```

> **Attention:**
> The total size of all logged messages is limited to 8 MB per script execution.

Method	Description
`sleep(delay)`	Delay JavaScript execution by `delay` milliseconds. This method is supported since Zabbix 5.0.20.

Example (delay execution by 15 seconds):

```
Zabbix.sleep(15000)
```

CurlHttpRequest

This object encapsulates cURL handle allowing to make simple HTTP requests. Errors are thrown as exceptions.

Attention:
The initialization of multiple `CurlHttpRequest` objects is limited to 10 per script execution.

Method	Description
`AddHeader(value)`	Adds HTTP header field. This field is used for all following requests until cleared with the `ClearHeader()` method. The total length of header fields that can be added to a single `CurlHttpRequest` object is limited to 128 Kbytes (special characters and header names included).
`ClearHeader()`	Clears HTTP header. If no header fields are set `CurlHttpRequest` will set Content-Type to application/json if the data being posted is json formatted and text/plain otherwise.
`GetHeaders()`	Returns object of received HTTP header fields. This method is available since Zabbix 5.0.4.
`Get(url, data)`	Sends HTTP GET request to the URL with optional *data* payload and returns the response.
`Put(url, data)`	Sends HTTP PUT request to the URL with optional *data* payload and returns the response.
`Post(url, data)`	Sends HTTP POST request to the URL with optional *data* payload and returns the response.
`Delete(url, data)`	Sends HTTP DELETE request to the URL with optional *data* payload and returns the response.
`Status()`	Returns the status code of the last HTTP request.
`SetProxy(proxy)`	Sets HTTP proxy to "proxy" value. If this parameter is empty then no proxy is used.

`SetHttpAuth(bitmask, username, password)`	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 | HTTPAUTH_BASIC, username, password)`
	`request.SetHttpAuth(HTTPAUTH_NONE)`
	This method is supported since Zabbix 5.0.9.

Example:

```
try {
    Zabbix.Log(4, 'jira webhook script value='+value);

    var result = {
        'tags': {
            'endpoint': 'jira'
        }
    },
    params = JSON.parse(value),
    req = new CurlHttpRequest(),
    fields = {},
    resp;

    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://jira.example.com/rest/api/2/issue/',
        JSON.stringify({"fields": fields})
```

```
    );

    if (req.Status() != 201) {
        throw 'Response code: '+req.Status();
    }

    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);
```

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);
}
catch (error) {
    return {'error.name' : error.name, 'error.message' : error.message}
}
```

- md5(string) - calculates the MD5 hash of a string; this function is supported since Zabbix 5.0.9
- sha256(string) - calculates the SHA256 hash of a string; this function is supported since Zabbix 5.0.9

5 CSV to JSON preprocessing

Overview

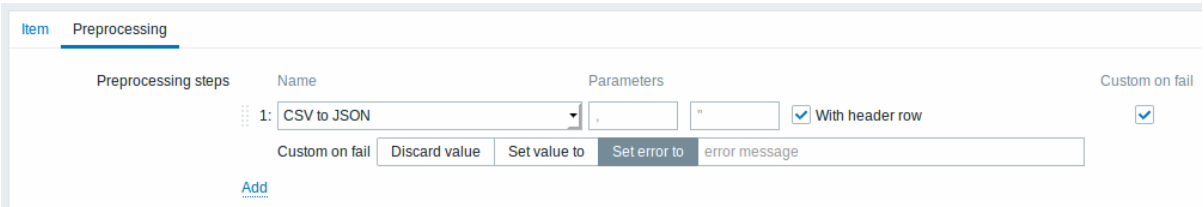
In this preprocessing step it is possible to convert CSV file data into JSON format. It's supported in:

- 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



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:

```
Nr,Item name,Key,Qty
1,active agent item,agent.hostname,33
"2","passive agent item","agent.version","44"
3,"active,passive agent items",agent.ping,55
```

Note:
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",
      "4":"33"
   },
   {
      "1":"2",
      "2":"passive agent item",
```

```
        "3":"agent.version",
        "4":"44"
    },
    {
        "1":"3",
        "2":"active,passive agent items",
        "3":"agent.ping",
        "4":"55"
    }
]
```

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 items 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
- Zabbix internal checks
- SSH checks
- Telnet checks
- External checks
- Aggregate checks
- Trapper items
- JMX monitoring
- ODBC checks
- Dependent items
- HTTP checks
- Prometheus checks

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).

> **Attention:**
> 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

These checks use the 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

Supported item keys

The table provides details on the item keys that you can use with Zabbix agent items.

See also:

- Items supported by platform
- Item keys supported by Zabbix agent 2
- Item keys specific for Windows agent
- Minimum permission level for Windows agent items

Mandatory and optional parameters

Parameters without angle brackets are mandatory. Parameters marked with angle brackets **< >** are optional.

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]
```

Key	Description	Return value	Parameters	Comments
agent.hostname	Agent host name.	String		Returns the actual value of the agent hostname from a configuration file.
agent.ping	Agent availability check.	Nothing - unavailable 1 - available		Use the **nodata()** trigger function to check for host unavailability.
agent.variant	Variant of Zabbix agent (Zabbix agent or Zabbix agent 2).	Integer		Example of returned value: 1 - Zabbix agent 2 - Zabbix agent 2 Supported since Zabbix 5.0.18.
agent.version	Version of Zabbix agent.	String		Example of returned value: 1.8.2
kernel.maxfiles				

Key		
	Maximum number of opened files supported by OS.	Integer
kernel.maxproc		
	Maximum number of processes supported by OS.	Integer

log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>,<persistent_dir>]

Key					
	Log file monitoring.	Log	**file** - full path and name of log file **regexp** - regular expression[4] describing the required 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 'MaxLinesPer-Second' in zab-bix_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 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: => log[/var/log/syslog] => log[/var/log/syslog,error] => log[/home/zabbix/logs/logfile„ *Using `output` parameter for extracting a number from log record:* => log[/app1/app.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,

Key
log.count[file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>,<persistent_dir>]

Key				
	Count of matched lines in log file monitoring.	Integer	**file** - full path and name of log file **regexp** - regular expression[4] describing the required pattern **encoding** - code page identifier **maxproclines** - maximum number of new lines per second the agent will analyze (cannot exceed 10000). Default value is 10*'MaxLines-PerSecond' in zabbix_agentd.conf. **mode** - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items). **maxdelay** - maximum delay in seconds. Type: float. Values: 0 - (default) never 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 Zabbix 4.4.7) - additional options: *mtime-noreread* - non-unique records, reread only if the file size changes	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. See also additional information on log monitoring. This item is not supported for Windows Event Log. Supported since Zabbix 3.2.0.

Key
logrt[file_regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>,<persistent_dir>]

Key				
	Log file monitoring with log rotation support.	Log	**file_regexp** - absolute path to file, with the file name specified using a regular expression[4]. Note that the regular expression applies only to the file name and does not need to match the entire name (e.g., /path/to/agent will match zabbix_agentd.log) **regexp** - regular expression[4] 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	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:

Key
logrt.count[file_regexp,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>,<persistent_dir>]

Key				
	Count of matched lines in log file monitoring with log rotation support.	Integer	**file_regexp** - absolute path to file, with the file name specified using a regular expression[4]. Note that the regular expression applies only to the file name and does not need to match the entire name (e.g., /path/to/agent will match zabbix_agentd.log) **regexp** - regular expression[4] describing the required content pattern **encoding** - code page identifier **maxproclines** - maximum number of new lines per second the agent will analyze (cannot exceed 10000). Default value is 10*'MaxLinesPerSecond' in zabbix_agentd.conf. **mode** - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items). **maxdelay** - maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; > 0.0 - ignore older lines in order to get the most recent lines analyzed	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. Log rotation is based on the last modification time of files. See also additional information on log monitoring. This item is not supported for Windows Event Log. Supported since Zabbix 3.2.0.

Key				
net.dns[<ip>,name,<type>,<timeout>,<count>,<protocol>]	Checks if DNS service is up.	0 - DNS is down (server did not respond or DNS resolution failed) 1 - DNS is up	**ip** - IP address of DNS server (leave empty for the default DNS server, ignored on Windows, unless using Zabbix agent 2 version 5.0.21 or newer) **name** - DNS name to query **type** - record type to be queried (default is *SOA*) **timeout** (ignored on Windows, unless using Zabbix agent 2 version 5.0.21 or newer) - timeout for the request in seconds (default is 1 second) **count** (ignored on Windows, unless using Zabbix agent 2 version 5.0.21 or newer) - number of tries for the request (default is 2) **protocol** (since version 3.0)- the protocol used to perform DNS queries: *udp* (default) or *tcp*	Example: => net.dns[8.8.8.8,example.com, The possible values for `type` are: *ANY*, *A*, *NS*, *CNAME*, *MB*, *MG*, *MR*, *PTR*, *MD*, *MF*, *MX*, *SOA*, *NULL*, *WKS* (not supported for Zabbix agent on Windows and, since version 5.0.21, Zabbix agent 2 on all OS), *HINFO*, *MINFO*, *TXT*, *SRV* Internationalized domain names are not supported, please use IDNA encoded names instead. SRV record type is supported since Zabbix agent versions 1.8.6 (Unix) and 2.0.0 (Windows). Naming before Zabbix 2.0 (still supported): *net.tcp.dns*
net.dns.record[<ip>,name,<type>,<timeout>,<count>,<protocol>]				

Key				
	Performs a DNS query.	Character string with the required type of information	**ip** - IP address of DNS server (leave empty for the default DNS server, ignored on Windows, unless using Zabbix agent 2 version 5.0.21 or newer) **name** - DNS name to query **type** - record type to be queried (default is *SOA*) **timeout** (ignored on Windows, unless using Zabbix agent 2 version 5.0.21 or newer) - timeout for the request in seconds (default is 1 second) **count** (ignored on Windows, unless using Zabbix agent 2 version 5.0.21 or newer) - number of tries for the request (default is 2) **protocol** (since version 3.0) - the protocol used to perform DNS queries: *udp* (default) or *tcp*	Example: => net.dns.record[8.8.8.8,exampl The possible values for `type` are: *ANY*, *A*, *NS*, *CNAME*, *MB*, *MG*, *MR*, *PTR*, *MD*, *MF*, *MX*, *SOA*, *NULL*, *WKS* (not supported for Zabbix agent on Windows and, since version 5.0.21, Zabbix agent 2 on all OS), *HINFO*, *MINFO*, *TXT*, *SRV* Internationalized domain names are not supported, please use IDNA encoded names instead. SRV record type is supported since Zabbix agent versions 1.8.6 (Unix) and 2.0.0 (Windows). Naming before Zabbix 2.0 (still supported): *net.tcp.dns.query*
net.if.collisions[if]	Number of out-of-window collisions.	Integer	**if** - network interface name	
net.if.discovery				

Key			
	List of network interfaces. Used for low-level discovery.	JSON object	Supported since Zabbix agent version 2.0. On FreeBSD, OpenBSD and NetBSD supported since Zabbix agent version 2.2. Some Windows versions (for example, Server 2008) might require the latest updates installed to support non-ASCII characters in interface names.
net.if.in[if,<mode>]			

Key				
	Incoming traffic statistics on network interface.	Integer	**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	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. The network interface GUID as the first parameter on Windows is supported since Zabbix 5.0.16. Examples: => net.if.in[eth0,errors] => net.if.in[eth0] You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. 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>]				

Key				
	Outgoing traffic statistics on network interface.	Integer	**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	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. The network interface GUID as the first parameter on Windows is supported since Zabbix 5.0.16. Examples: => net.if.out[eth0,errors] => net.if.out[eth0] You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. 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>]				

Key					
	Sum of incoming and outgoing traffic statistics on network interface.	Integer	**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	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. The network interface GUID as the first parameter on Windows is supported since Zabbix 5.0.16. Examples: => net.if.total[eth0,errors] => net.if.total[eth0] You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. 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 packets on your platform.

Key				
net.tcp.listen[port]	Checks if this TCP port is in LISTEN state.	0 - it is not in LISTEN state 1 - it is in LISTEN state	**port** - TCP port number	Example: => net.tcp.listen[80] On Linux supported since Zabbix agent version 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.
net.tcp.port[<ip>,port]	Checks if it is possible to make TCP connection to specified port.	0 - cannot connect 1 - can connect	**ip** - IP or DNS name (default is 127.0.0.1) **port** - port number	Example: => 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,<ip>, Note that these checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually). Old naming: *check_port[*]*

Key
net.tcp.service[service,<ip>,<port>]

Key				
	Checks if service is running and accepting TCP connections.	0 - service is down 1 - service is running	**service** - either of: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	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 (since Zabbix 5.0.3). 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. Old naming: *check_service[*]*

Key				
net.tcp.service.perf[service,<ip>,<port>]	Checks performance of TCP service.	0 - service is down seconds - the number of seconds spent while connecting to the service	**service** - either of: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	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>, for checks like these. Checking of LDAP and HTTPS on Windows is only supported by Zabbix agent 2 (since Zabbix 5.0.3). 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. Old naming: *check_service_perf[*]*
net.udp.listen[port]	Checks if this UDP port is in LISTEN state.	0 - it is not in LISTEN state 1 - it is in LISTEN state	**port** - UDP port number	Example: => net.udp.listen[68] On Linux supported since Zabbix agent version 1.8.4

Key				
net.udp.service[service,<ip>,<port>]	Checks if service is running and responding to UDP requests.	0 - service is down 1 - service is running	**service** - *ntp* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	Example: => net.udp.service[ntp„45] → can be used to test the availability of NTP service on UDP port 45. This item is supported since Zabbix 3.0.0, but *ntp* service was available for net.tcp.service[] item in prior versions.
net.udp.service.perf[service,<ip>,<port>]	Checks performance of UDP service.	0 - service is down seconds - the number of seconds spent waiting for response from the service	**service** - *ntp* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	Example: => net.udp.service.perf[ntp] → can be used to test response time from NTP service. This item is supported since Zabbix 3.0.0, but *ntp* service was available for net.tcp.service[] item in prior versions.
proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]				

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[4]) **mode** - data gathering mode: *avg1* (default), *avg5*, *avg15* **zone** - target zone: *current* (default), *all*. This parameter is supported on Solaris only.	Examples: => proc.cpu.util[,root] → CPU utilization of all processes running under the ”root” user => proc.cpu.util[zabbix_server,za → 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

Key
proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]

Key					
	Memory used by process in bytes.	Integer - with `mode` as *max*, *min*, *sum* Float - with `mode` as *avg*	**name** - process name (default is *all processes*) **user** - user name (default is *all users*) **mode** - possible values: *avg*, *max*, *min*, *sum* (default) **cmdline** - filter by command line (it is a regular expression[4]) **memtype** - type of memory used by process	Examples: => proc.mem[,root] → memory used by all processes running under the ”root” user => proc.mem[zabbix_server,zabb → memory used by all zabbix_server processes running under the zabbix user => proc.mem[,oracle,max,oracleZ → 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

Key
proc.num[<name>,<user>,<state>,<cmdline>,<zone>]

Key				
	The number of processes.	Integer	**name** - process name (default is *all processes*) **user** - user name (default is *all users*) **state** (*disk* and *trace* options since version 3.4.0) - 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[4]) **zone** - target zone: *current* (default), *all*. This parameter is supported on Solaris only.	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,oracle → 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). On Windows, only the `name` and `user` parameters are supported. 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

Key				
sensor[device,sensor,<mode>]	Hardware sensor reading.	Float	**device** - device name **sensor** - sensor name **mode** - possible values: *avg*, *max*, *min* (if this parameter is omitted, device and sensor are treated verbatim).	Reads /proc/sys/dev/sensors on Linux 2.4. Example: => sensor[w83781d-i2c-0-2d,temp1] Prior to Zabbix 1.8.4, the *sensor[temp1]* format was used. Reads /sys/class/hwmon on Linux 2.6+. See a more detailed description of sensor item on Linux. 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.boottime	System boot time.	Integer (Unix timestamp)		
system.cpu.discovery	List of detected CPUs/CPU cores. Used for low-level discovery.	JSON object		Supported on all platforms since 2.4.0.
system.cpu.intr	Device interrupts.	Integer		
system.cpu.load[<cpu>,<mode>]				

Key				
	CPU load.	Float	**cpu** - possible values: *all* (default), *percpu* (since version 2.0; total load divided by online CPU count) **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*	Example: => system.cpu.load[,avg5] Old naming: *system.cpu.loadX*
system.cpu.num[<type>]				
	Number of CPUs.	Integer	**type** - possible values: *online* (default), *max*	Example: => system.cpu.num
system.cpu.switches				
	Count of context switches.	Integer		Old naming: *system.tem[switches]*
system.cpu.util[<cpu>,<type>,<mode>,<logical_or_physical>]				

Key				
	CPU utilization percentage.	Float	**cpu** - *<CPU number>* or *all* (default) **type** - possible values: *user* (default), *idle*, *nice*, *system* (default for Windows), *iowait*, *interrupt*, *softirq*, *steal*, *guest* (on Linux kernels 2.6.24 and above), *guest_nice* (on Linux kernels 2.6.33 and above). See also platform-specific details for this parameter. **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15* **logical_or_physical** (since version 5.0.3; AIX only) - possible values: *logical* (default), *physical*.	On Windows, 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. Example: => system.cpu.util[0,user,avg5] Old naming: *system.cpu.idleX*, *system.cpu.niceX*, *system.cpu.systemX*, *system.cpu.userX*
system.hostname[<type>, <transform>]				

Key				
	System host name.	String	**type** (before version 5.0.18 supported on Windows only) - possible values: *netbios* (default on Windows), *host* (default on Linux), *shorthost* (since version 5.0.18; returns part of the hostname before the first dot, a full string for names without dots). **transform** (since version 5.0.18) - possible values: *none* (default), *lower* (convert to lowercase)	The value is acquired by either GetComputerName() (for **netbios**) or gethostname() (for **host**) functions on Windows and by taking `nodename` from the uname() system API output on other systems. Examples of returned values: *on Linux*: => system.hostname → linux-w7x1 => system.hostname → example.com => system.hostname[shorthost] → example *on Windows*: => system.hostname → WIN-SERV2008-I6 => system.hostname[host] → Win-Serv2008-I6LonG => system.hostname[host,lower] → win-serv2008-i6long See also a more detailed description.
system.hw.chassis[<info>]				

Key				
	Chassis information.	String	**info** - one of full (default), model, serial, type or vendor	Example: system.hw.chassis[full] Hewlett-Packard HP Pro 3010 Small Form Factor PC CZXXXXXXXX Desktop] This key depends on the availability of the SMBIOS 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 agent version 2.0.
system.hw.cpu[<cpu>,<info>]				

Key				
	CPU information.	String or integer	**cpu** - *<CPU number>* or *all* (default) **info** - possible values: *full* (default), *curfreq*, *maxfreq*, *model* or *vendor*	Example: => system.hw.cpu[0,vendor] → AuthenticAMD Gathers info from /proc/cpuinfo and /sys/devices/system/cpu/[cpur If a CPU number and *curfreq* or *maxfreq* is specified, a numeric value is returned (Hz). Supported since Zabbix agent version 2.0.
system.hw.devices[<type>]				
	Listing of PCI or USB devices.	Text	**type** - *pci* (default) or *usb*	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 agent version 2.0.
system.hw.macaddr[<interface>,<format>]				

Key				
	Listing of MAC addresses.	String	**interface** - *all* (default) or a regular expression[4] **format** - *full* (default) or *short*	Lists MAC addresses of the interfaces whose name matches the given `interface` regular expression[4] (*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 agent version 2.0.
system.localtime[<type>]	System time.	Integer - with `type` as *utc* String - with `type` as *local*	**type** (since version 2.0) - 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	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* screen element.
system.run[command,<mode>]				

Key				
	Run specified command on the host.	Text result of the command 1 - with `mode` as *nowait* (regardless of command result)	**command** - command for execution **mode** - possible values: *wait* - wait end of execution (default), *nowait* - do not wait	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: => system.run[ls -l /] → detailed file list of root directory. *Note*: system.run items are disabled by default. Learn how to enable them. 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>]				

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

Key				
				Comments This item is supported on AIX only, since Zabbix 1.8.1. Take note of the following limitations in these items: => system.stat[cpu,app] - supported only on AIX LPAR of type ”Shared” => system.stat[cpu,ec] - supported on AIX LPAR of type ”Shared” and ”Dedicated” (”Dedicated” always returns 100 (percent)) => system.stat[cpu,lbusy] - supported only on AIX LPAR of type ”Shared” => system.stat[cpu,pc] - supported on AIX LPAR of type ”Shared” and ”Dedicated” => system.stat[ent] - supported on AIX LPAR of type ”Shared” and ”Dedicated”
system.sw.arch	Software architecture information.	String		Example: => system.sw.arch → i686 Info is acquired from uname() function. Supported since Zabbix agent version 2.0.
system.sw.os[<info>]				

Key				
	Operating system information.	String	**info** - possible values: *full* (default), *short* or *name*	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 agent version 2.0.
system.sw.packages[<regexp>,<manager>,<format>]				

Key				
	Listing of installed packages.	Text	**regexp** - *all* (default) or a regular expression[4] **manager** - *all* (default) or a package manager **format** - *full* (default) or *short*	Lists (alphabetically) installed packages whose name matches the given `package` regular expression[4] (*all* lists them all). Example: => system.sw.packages[mini,dpkg,sh → 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 agent version 2.0.
system.swap.in[<device>,<type>]				

Key				
	Swap in (from device into memory) statistics.	Integer	**device** - device used for swapping (default is *all*) **type** - possible values: *count* (number of swapins), *sectors* (sectors swapped in), *pages* (pages swapped in). See also platform-specific details for this parameter.	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)
system.swap.out[<device>,<type>]				
	Swap out (from memory onto device) statistics.	Integer	**device** - device used for swapping (default is *all*) **type** - possible values: *count* (number of swapouts), *sectors* (sectors swapped out), *pages* (pages swapped out). See also platform-specific details for this parameter.	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>]				

Key				
	Swap space size in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**device** - device used for swapping (default is *all*) **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. See also platform-specific details for this parameter.	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*). 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(_Tota` key to obtain correct swap space percentage. Old naming: *system.swap.free*, *system.swap.total*
system.uname				

Key			
	Identification of the system.	String	Example of returned value (Unix): FreeBSD localhost 4.2-RELEASE FreeBSD 4.2-RELEASE #0: Mon Nov i386 Example of returned value (Windows): Windows ZABBIX-WIN 6.0.6001 Microsoft® Windows Server® 2008 Standard Service Pack 1 x86 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. On Windows since Zabbix 3.0 the value for this item is obtained from Win32_OperatingSystem and Win32_Processor WMI classes. Previously it was obtained from volatile Windows APIs and undocumented registry keys.

Key			
system.uptime			
	System uptime in seconds.	Integer	In item configuration, use **s** or **uptime** units to get readable values.
system.users.num			
	Number of users logged in.	Integer	**who** command is used on the agent side to obtain the value.
vfs.dev.discovery			
	List of block devices and their type. Used for low-level discovery.	JSON object	This item is supported on Linux platform only. Supported since Zabbix 4.4.0.
vfs.dev.read[<device>,<type>,<mode>]			

Key				
	Disk read statistics.	Integer - with `type` in *sectors*, *operations*, *bytes* Float - with `type` in *sps*, *ops*, *bps* *Note*: if using an update interval of three hours or more[2], will always return '0'	**device** - disk device (default is *all* [3]) **type** - possible values: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* Note that 'type' parameter support and defaults depend on the platform. See platform-specific details. *sps*, *ops*, *bps* stand for: sectors, operations, bytes per second, respectively. **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*. This parameter is supported only with `type` in: sps, ops, bps.	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: => vfs.dev.read[,operations] *sps*, *ops* and *bps* on supported platforms used to be limited to 8 devices (7 individual and one *all*). Since Zabbix 2.0.1 this limit is 1024 devices (1023 individual and one for *all*). Old naming: *io[*]*
vfs.dev.write[<device>,<type>,<mode>]				

Key					
		Disk write statistics.	Integer - with `type` in *sectors*, *operations*, *bytes* Float - with `type` in *sps*, *ops*, *bps* *Note*: if using an update interval of three hours or more[2], will always return '0'	**device** - disk device (default is *all* [3]) **type** - possible values: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* Note that 'type' parameter support and defaults depend on the platform. See platform-specific details. *sps*, *ops*, *bps* stand for: sectors, operations, bytes per second, respectively. **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*. This parameter is supported only with `type` in: sps, ops, bps.	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: => vfs.dev.write[,operations] *sps*, *ops* and *bps* on supported platforms used to be limited to 8 devices (7 individual and one *all*). Since Zabbix 2.0.1 this limit is 1024 (1023 individual and one for *all*). Old naming: *io[*]*
vfs.dir.count[dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age>,<max_age					

Key					
	Directory entry count.	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	Environment variables, e.g. %APP_HOME%, $HOME and %TEMP% are not supported. Pseudo-directories "." and ".." are never counted. Symbolic links are never followed for directory traversal. On Windows, directory symlinks are skipped. Both `regex_incl` and `regex_excl` are being applied to files and directories when calculating entry size, but are ignored when picking subdirectories to traverse (if regex_incl is "(?i)^.+\.zip$" 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.

Key
vfs.dir.size[dir,<regex_incl>,<regex_excl>,<mode>,<max_depth>,<regex_excl_dir>]

Key				
	Directory size (in bytes).	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) **mode** - possible values: *apparent* (default) - gets apparent file sizes rather than disk usage (acts as `du -sb dir`), *disk* - gets disk usage (acts as `du -s -B1 dir`). Unlike du command, vfs.dir.size item takes hidden files in account when calculating directory size (acts as `du -sb .[^.]* * ` within dir). **max_depth** - maximum depth of subdirectories to traverse. **-1** (default) - unlimited, **0** - no descending into subdirectories. **regex_excl_dir** - regular expression describing the	Only directories with at least read permission for *zabbix* user are calculated. On Windows any symlink is skipped and hard links are taken into account only once. 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 'log', excluding files containing '.old' The file size limit depends on large file support. Supported since Zabbix 3.4.0.

Key				
vfs.file.cksum[file]	File checksum, calculated by the UNIX cksum algorithm.	Integer	**file** - full path to file	Example: => vfs.file.cksum[/etc/passwd] Example of returned value: 1938292000 Old naming: *cksum* The file size limit depends on large file support.
vfs.file.contents[file,<encoding>]	Retrieving contents of a file.	Text	**file** - full path to file **encoding** - code page identifier	Returns an empty string if the file is empty or contains LF/CR characters only. Byte order mark (BOM) is excluded from the output. Example: => vfs.file.contents[/etc/passwd] This item is limited to files no larger than 64 Kbytes. Supported since Zabbix agent version 2.0.
vfs.file.exists[file,<types_incl>,<types_excl>]				

Key					
	Checks if file exists.	0 - not found 1 - file of the specified type exists	**file** - full path to file **types_incl** - list of file types to include, possible values: *file* (regular file, default (if types_excl is not set)), *dir* (directory), *sym* (symbolic link), *sock* (socket), *bdev* (block device), *cdev* (character device), *fifo* (FIFO), *dev* (synonymous with "bdev,cdev"), *all* (all mentioned types, default if types_excl is set). **types_excl** - list of file types to exclude, see types_incl for possible values (by default no types are excluded)	Multiple types must be separated with a comma and the entire set enclosed in quotes "". On Windows the double quotes have to be backslash '\' escaped and the whole item key enclosed in double quotes when using the command line utility for calling zabbix_get.exe or agent2. If the same type is in both <types_incl> and <types_excl>, files of this type are excluded. Examples: => vfs.file.exists[/tmp/application => vfs.file.exists[/tmp/application => vfs.file.exists[/tmp/application The file size limit depends on large file support. The `types_incl`, `types_excl` parameters are supported since Zabbix 5.0.2. Note that the item may turn unsupported in 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).

Key				
vfs.file.md5sum[file]	MD5 checksum of file.	Character string (MD5 hash of the file)	**file** - full path to file	Example: => vfs.file.md5sum[/usr/local/etc/ Example of returned value: b5052decb577e0fffd622d6dd The file size limit (64 MB) for this item was removed in version 1.8.6. The file size limit depends on large file support.
vfs.file.regexp[file,regexp,<encoding>,<start line>,<end line>,<output>]				

Key					
		Find string in a file.	The line containing the matched string, or as specified by the optional `output` parameter	**file** - full path to file **regexp** - regular expression[4] 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).	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,zal => vfs.file.regexp[/path/to/some/f 9]+)$”„3,5,\1] => vfs.file.regexp[/etc/passwd,”^ 9]+)”„„\1] → getting the ID of user *zabbix*
vfs.file.regmatch[file,regexp,<encoding>,<start line>,<end line>]					

Key				
	Find string in a file.	0 - match not found 1 - found	**file** - full path to file **regexp** - regular expression[4] 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).	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.
vfs.file.size[file]	File size (in bytes).	Integer	**file** - full path to file	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.
vfs.file.time[file,<mode>]	File time information.	Integer (Unix timestamp)	**file** - full path to the file **mode** - possible values: *modify* (default) - last time of modifying file content, *access* - last time of reading file, *change* - last time of changing file properties	Example: => vfs.file.time[/etc/passwd,modi The file size limit depends on large file support.
vfs.fs.discovery				

Key				
	List of mounted filesystems and their types. Used for low-level discovery.	JSON object		Supported since Zabbix agent version 2.0. {#FSDRIVETYPE} macro is supported on Windows since Zabbix agent version 3.0.
vfs.fs.get				
	List of mounted filesystems, their types, disk space and inode statistics. Can be used for low-level discovery.	JSON object		Since Zabbix 5.0.30, this item is capable of reporting file systems with the inode count equal to zero, which can be the case for file systems with dynamic inodes (e.g. btrfs).
vfs.fs.inode[fs,<mode>]				
	Number or percentage of inodes.	Integer - for number Float - for percentage	**fs** - filesystem **mode** - possible values: *total* (default), *free*, *used*, //pfree // (free, percentage), *pused* (used, percentage)	Since Zabbix 5.0.30, this item will not become unsupported in pfree/pused modes if the inode count equals to zero, which can be the case for file systems with dynamic inodes (e.g. btrfs). Instead the pfree/pused values for such file systems will be reported as ”100” and ”0” respectively. Example: => vfs.fs.inode[/,pfree]
vfs.fs.size[fs,<mode>]				

Key				
	Disk space in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**fs** - filesystem **mode** - possible values: *total* (default), *free*, *used*, *pfree* (free, percentage), *pused* (used, 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. Old naming: *vfs.fs.free[*]*, *vfs.fs.total[*]*, *vfs.fs.used[*]*, *vfs.fs.pfree[*]*, *vfs.fs.pused[*]*
vm.memory.size[<mode>]				
	Memory size in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**mode** - possible values: *total* (default), *active*, *anon*, *buffers*, *cached*, *exec*, *file*, *free*, *inactive*, *pinned*, *shared*, *slab*, *wired*, *used*, *pused* (used, percentage), *available*, *pavailable* (available, percentage) See also platform-specific support and additional details for this parameter.	This item accepts three categories of parameters: 1) *total* - total amount of memory; 2) platform-specific memory types: *active*, *anon*, *buffers*, *cached*, *exec*, *file*, *free*, *inactive*, *pinned*, *shared*, *slab*, *wired*; 3) user-level estimates on how much memory is used and available: *used*, *pused*, *available*, *pavailable*.
web.page.get[host,<path>,<port>]				

Key					
	Get content of web page.	Web page source as text (including headers)	**host** - hostname or URL (as `scheme://host:port/path`, where only *host* is mandatory). Allowed URL schemes: *http*, *https*[5]. 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` is only possible with cURL support [5]. 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: => web.page.get[www.example.c => web.page.get[https://www.exa => web.page.get[https://blog.exa => web.page.get[localhost:80] => web.page.get[”[::1]/server-status”]
web.page.perf[host,<path>,<port>]					

Key					
	Loading time of full web page (in seconds).	Float	**host** - hostname or URL (as `scheme://host:port/path`, where only *host* is mandatory). Allowed URL schemes: *http*, *https*[5]. 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` is only possible with cURL support [5]. 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: => web.page.perf[www.example.c => web.page.perf[https://www.ex
web.page.regexp[host,<path>,<port>,regexp,<length>,<output>]					

Key					
	Find string on a web page.	The matched string, or as specified by the optional `output` parameter	**host** - hostname or URL (as `scheme://host:port/path`, where only *host* is mandatory). Allowed URL schemes: *http*, *https*[5]. 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` is only possible with cURL support [5]. Punycode is supported in hostnames. **path** - path to HTML document (default is /) **port** - port number (default is 80 for HTTP) **regexp** - regular expression[4] 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	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. Content extraction using the `output` parameter takes place on the agent. The `output` parameter is supported from version 2.2. Example: => web.page.regexp[www.examp => web.page.regexp[https://www

Key				
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)	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)	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, sbd 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] Perl Compatible Regular Expression (PCRE) since Zabbix 3.4; POSIX-extended regular expression before that. See also: Regular expressions supported by location.

[5] SSL (HTTPS) is supported only if agent is compiled with cURL support. Otherwise the item will turn unsupported.

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-specific item keys

Item keys

The table provides details on the item keys that you can use with Zabbix Windows agent only.

See also: Minimum permission level for Windows agent items

Key	Description	Return value	Parameters	Comments
eventlog[name,<regexp>,<severity>,<source>,<eventid>,<maxlines>,<mode>]				

Key					
	Event log monitoring.	Log	**name** - name of event log **regexp** - regular expression describing the required pattern **severity** - regular expression describing severity (case-insensitive) This parameter accepts the following values: ”Information”, ”Warning”, ”Error”, ”Critical”, ”Verbose” (since Zabbix 2.2.0 running on Windows Vista or newer) **source** - regular expression describing source identifier (case-insensitive; regular expression is supported since Zabbix 2.2.0) **eventid** - regular expression describing the event identifier(s) **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** - possible values: *all* (default),	The item must be configured as an active check. Examples: => eventlog[Application] => eventlog[Security,,”Failure Audit”,,^(529|680)$] => eventlog[System,,”Warning\|Error”] => eventlog[System,,,^1$] => eventlog[System,,,@TWOSHORT] - here a custom regular expression named `TWOSHORT` is referenced (defined as a *Result is TRUE* type, the expression itself being `^1$\|^70$`). *Note* that the agent is unable to send in events from the ”Forwarded events” log. The `mode` parameter is supported since Zabbix 2.0.0. ”Windows Eventing 6.0” is supported since Zabbix 2.2.0. Note that selecting 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

Key			
net.if.list	Network interface list (includes interface type, status, IPv4 address, description).	Text	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.
perf_counter[counter,<interval>]			

Key				
	Value of any Windows performance counter.	Integer, float, string or text (depending on the request)	**counter** - path to the counter **interval** - 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 \System\Threads). It will not work as expected for counters that require more that one sample - like CPU utilization. Since 1.6, `interval` is used, so the check returns an average value for last "interval" seconds every time. See also: Windows performance counters.
perf_counter_en[counter,<interval>]	Value of any Windows performance counter in English.	Integer, float, string or text (depending on the request)	**counter** - path to the counter in English **interval** - 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: `HKEY_LOCAL_MACHINE\SOFT NT\CurrentVersion\Perfl` Supported since Zabbix agent versions 4.0.13 and 4.2.7.
perf_instance.discovery[object]				

Key				
	List of object instances of Windows performance counters. Used for low-level discovery.	JSON object	**object** - object name (localized)	Supported since Zabbix agent version 5.0.1.
perf_instance_en.discovery[object]				
	List of object instances of Windows performance counters, discovered using object names in English. Used for low-level discovery.	JSON object	**object** - object name (in English)	Supported since Zabbix agent version 5.0.1.
proc_info[process,<attribute>,<type>]				

Key					
	Various information about specific process(es).	Float	**process** - process name **attribute** - requested process attribute **type** - representation type (meaningful when more than one process with the same name exists)	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

Key				
service.discovery	List of Windows services. Used for low-level discovery.	JSON object		Supported since Zabbix agent version 3.0.
service.info[service,<param>]	Information about a service.	Integer - with `param` as *state*, *startup* String - with `param` as *displayname*, *path*, *user* Text - with `param` as *description* Specifically for *state*: 0 - running, 1 - paused, 2 - start pending, 3 - pause pending, 4 - continue pending, 5 - stop pending, 6 - stopped, 7 - unknown, 255 - no such service Specifically for *startup*: 0 - automatic, 1 - automatic delayed, 2 - manual, 3 - disabled, 4 - unknown, 5 - automatic trigger start, 6 - automatic delayed trigger start, 7 - manual trigger start	**service** - a real service name or its display name as seen in MMC Services snap-in **param** - *state* (default), *displayname*, *path*, *user*, *startup* or *description*	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 Items 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.
services[<type>,<state>,<exclude>]				

Key				
	Listing of services.	0 - if empty Text - list of services separated by a newline	**type** - *all* (default), *automatic*, *manual* or *disabled* **state** - *all* (default), *stopped*, *started*, *start_pending*, *stop_pending*, *running*, *continue_pending*, *pause_pending* or *paused* **exclude** - services to exclude from the result. Excluded services should be listed in double quotes, separated by comma, without spaces.	Examples: => services[,started] - list of started services => services[automatic, stopped] - list of stopped services, that should be run => services[automatic, stopped, "service1,service2,service3"] - list of stopped services, that should be run, excluding services with names service1, service2 and service3 The `exclude` parameter is supported since Zabbix 1.8.1.
wmi.get[<namespace>,<query>]	Execute WMI query and return the first selected object.	Integer, float, string or text (depending on the request)	**namespace** - WMI namespace **query** - WMI query returning a single object	WMI queries are performed with WQL. Example: => wmi.get[root\cimv2,select status from Win32_DiskDrive where Name like '%PHYSICALDRIVE0%'] - returns the status of the first physical disk This key is supported since Zabbix 2.2.0.
wmi.getall[<namespace>,<query>]				

Key				
	Execute WMI query and return the whole response. Can be used for low-level discovery.	JSON object	**namespace** - WMI namespace **query** - WMI query	WMI queries are performed with WQL. Example: => wmi.getall[root\cimv2,select * from Win32_DiskDrive where Name like '%PHYSICALDRIVE%'] - returns status information of physical disks JSONPath preprocessing can be used to point to more specific values in the returned JSON. This key is supported since Zabbix 4.4.0.
vm.vmemory.size[<type>]				

Key				
	Virtual memory size in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**type** - possible values: *available* (available virtual memory), *pavailable* (available virtual memory, in percent), *pused* (used virtual memory, in percent), *total* (total virtual memory, default), *used* (used virtual memory)	Example: => vm.vmemory.size[pavailable] → available virtual memory, in percentage Monitoring of virtual memory statistics is based on: * Total virtual memory on Windows (total physical + page file size); * The maximum amount of memory Zabbix agent can commit; * The current committed memory limit for the system or Zabbix agent, whichever is smaller. This key is supported since Zabbix 3.0.7 and 3.2.3.

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

Item keys

The table provides details on the item keys that you can use with Zabbix agent 2 only.

See also: Plugins supplied out-of-the-box

Note:
Parameters without angle brackets are mandatory. Parameters marked with angle brackets < > are optional.

Key	Description	Return value	Parameters	Comments
ceph.df.details[connString,<user>,<apikey>]	Cluster's data usage and distribution among pools.	JSON object	**connString** - URI or session name. **user, password** - Ceph login credentials.	This item is supported for the Ceph plugin.
ceph.osd.stats[connString,<user>,<apikey>]	Aggregated and per OSD statistics.	JSON object	**connString** - URI or session name. **user, password** - Ceph login credentials.	This item is supported for the Ceph plugin.
ceph.osd.discovery[connString,<user>,<apikey>]	List of discovered OSDs. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - Ceph login credentials.	This item is supported for the Ceph plugin.
ceph.osd.dump[connString,<user>,<apikey>]	Usage thresholds and statuses of OSDs.	JSON object	**connString** - URI or session name. **user, password** - Ceph login credentials.	This item is supported for the Ceph plugin.
ceph.ping[connString,<user>,<apikey>]	Tests whether a connection to Ceph can be established.	0 - connection is broken (if there is any error presented including AUTH and configuration issues) 1 - connection is successful.	**connString** - URI or session name. **user, password** - Ceph login credentials.	This item is supported for the Ceph plugin.
ceph.pool.discovery[connString,<user>,<apikey>]	List of discovered pools. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - Ceph login credentials.	This item is supported for the Ceph plugin.
ceph.status[connString,<user>,<apikey>]				

Key				
	Overall cluster's status.	JSON object	**connString** - URI or session name. **user, password** - Ceph login credentials.	This item is supported for the Ceph plugin.
docker.container_info[<ID>,<info>]				
	Low-level information about a container.	An output of the ContainerInspect API call serialized as JSON	**ID** - ID or name of the container. **info** - the amount of information returned. Supported values: *short* (default) or *full*.	This item is supported since Zabbix 5.0.0 for the Docker plugin. The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.
docker.container_stats[<ID>]				
	Container resource usage statistics.	An output of the ContainerStats API call and CPU usage percentage serialized as JSON	**ID** - ID or name of the container.	This item is supported for the Docker plugin. The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.
docker.containers				
	A list of containers.	An output of the ContainerList API call serialized as JSON	-	This item is supported for the Docker plugin. The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.
docker.containers.discovery[<options>]				
	A list of containers. Used for low-level discovery.	JSON object	**options** - specifies whether all or only running containers should be discovered. Supported values: *true* - return all containers; *false* - return only running containers (default).	This item is supported for the Docker plugin. The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.
docker.data_usage				
	Information about current data usage.	An output of the System-DataUsage API call serialized as JSON	-	This item is supported for the Docker plugin. The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.
docker.images				

Key				
	A list of images.	An output of the ImageList API call serialized as JSON	-	This item is supported for the Docker plugin. The Agent2 user (’zabbix’) must be added to the ’docker’ group for sufficient privileges. Otherwise the check will fail.
docker.images.discovery				
	A list of images. Used for low-level discovery.	JSON object	-	This item is supported for the Docker plugin. The Agent2 user (’zabbix’) must be added to the ’docker’ group for sufficient privileges. Otherwise the check will fail.
docker.info				
	System information.	An output of the SystemInfo API call serialized as JSON	-	This item is supported for the Docker plugin. The Agent2 user (’zabbix’) must be added to the ’docker’ group for sufficient privileges. Otherwise the check will fail.
docker.ping				
	Test if a Docker daemon is alive or not.	1 - connection is alive 0 - connection is broken	-	This item is supported for the Docker plugin. The Agent2 user (’zabbix’) must be added to the ’docker’ group for sufficient privileges. Otherwise the check will fail.
memcached.ping[connString,<user>,<password>]				
	Test if 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)	**connString** - URI or session name.	This item is supported for the Memcached plugin.
memcached.stats[connString,<user>,<password>,<type>]				
	Gets the output of the STATS command.	JSON - output is serialized as JSON	**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).	This item is supported for the Memcached plugin.
mongodb.collection.stats[connString,<user>,<password>,<database>,collection]				

Key				
	Returns a variety of storage statistics for a given collection.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials. **database** - database name (default: admin). **collection** — collection name.	This item is supported for the MongoDB plugin.
mongodb.collections.discovery[connString,<user>,<password>]				
	Returns a list of discovered collections. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.	This item is supported for the MongoDB plugin.
mongodb.collections.usage[connString,<user>,<password>]				
	Returns usage statistics for collections.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.	This item is supported for the MongoDB plugin.
mongodb.connpool.stats[connString,<user>,<password>]				
	Returns information regarding the open outgoing connections from the current database instance to other members of the sharded cluster or replica set.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.	This item is supported for the MongoDB plugin.
mongodb.db.stats[connString,<user>,<password>,<database>]				
	Returns statistics reflecting a given database system state.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials. **database** - database name (default: admin).	This item is supported for the MongoDB plugin.
mongodb.db.discovery[connString,<user>,<password>]				
	Returns a list of discovered databases. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.	This item is supported for the MongoDB plugin.
mongodb.jumbo_chunks.count[connString,<user>,<password>]				
	Returns count of jumbo chunks.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.	This item is supported for the MongoDB plugin.
mongodb.oplog.stats[connString,<user>,<password>]				
	Returns a status of the replica set, using data polled from the oplog.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.	This item is supported for the MongoDB plugin.
mongodb.ping[connString,<user>,<password>]				
	Tests if 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).	**connString** - URI or session name. **user, password** - MongoDB login credentials.	This item is supported for the MongoDB plugin.
mongodb.rs.config[connString,<user>,<password>]				

Key				
	Returns a current configuration of the replica set.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.	This item is supported for the MongoDB plugin.
mongodb.rs.status[connString,<user>,<password>]				
	Returns a replica set status from the point of view of the member where the method is run.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.	This item is supported for the MongoDB plugin.
mongodb.server.status[connString,<user>,<password>]				
	Returns database state.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.	This item is supported for the MongoDB plugin.
mongodb.sh.discovery[connString,<user>,<password>]				
	Returns a list of discovered shards present in the cluster.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.	This item is supported for the MongoDB plugin.
mysql.db.discovery[connString,<username>,<password>]				
	List of MySQL databases. Used for low-level discovery.	Result of the "show databases" SQL query in LLD JSON format.	**connString** - URI or session name. **username, password** - MySQL login credentials.	This item is supported for the MySQL plugin.
mysql.db.size[connString,<username>,<password>,dbName]				
	Database size in bytes.	Result of the "select coalesce(sum(data_length + index_length),0) as size from information_schema.tables where table_schema=?" SQL query for specific database in bytes.	**connString** - URI or session name. **username, password** - MySQL login credentials. **dbName** - Database name.	This item is supported for the MySQL plugin.
mysql.get_status_variables[connString,<username>,<password>]				
	Values of global status variables.	Result of the "show global status" SQL query in JSON format.	**connString** - URI or session name. **username, password** - MySQL login credentials.	This item is supported for the MySQL plugin.
mysql.ping[connString,<username>,<password>]				
	Test if 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).	**connString** - URI or session name. **username, password** - MySQL login credentials.	This item is supported for the MySQL plugin.
mysql.replication.discovery[connString,<username>,<password>]				

Key				
	List of MySQL replications. Used for low-level discovery.	Result of the "show slave status" SQL query in LLD JSON format.	**connString** - URI or session name. **username, password** - MySQL login credentials.	This item is supported for the MySQL plugin.
mysql.replication.get_slave_status[connString,<username>,<password>,<masterHost>]				
	Replication status.	Result of the "show slave status" SQL query in JSON format.	**connString** - URI or session name. **username, password** - MySQL login credentials. **masterHost** - Replication master host name.	This item is supported for the MySQL plugin.
mysql.version[connString,<username>,<password>]				
	MySQL version.	String with MySQL instance version.	**connString** - URI or session name. **username, password** - MySQL login credentials.	This item is supported for the MySQL plugin.
oracle.diskgroups.stats[connString,<user>,<password>,<service>,<diskgroup>]				
	Automatic Storage Management (ASM) disk groups statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **diskgroup** - name of the ASM disk group to query.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.diskgroups.discovery[connString,<user>,<password>,<service>]				
	List of ASM disk groups. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.archive.info[connString,<user>,<password>,<service>,<destination>]				
	Archive logs statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **destination** - name of the destination to query.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.cdb.info[connString,<user>,<password>,<service>,<database>]				

Key				
	Container Databases (CDBs) information.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **database** - name of the database to query.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.custom.query[connString,<user>,<password>,<service>,queryName,<args...>]				
	Result of a custom query.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **queryName** - name of a custom query (must be equal to a name of an sql file without an extension). **args...** - one or several comma-separated arguments to pass to a query.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.datafiles.stats[connString,<user>,<password>,<service>]				
	Data files statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.db.discovery[connString,<user>,<password>,<service>]				
	List of databases. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.fra.stats[connString,<user>,<password>,<service>]				

Key				
	Fast Recovery Area (FRA) statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.instance.info[connString,<user>,<password>,<service>]				
	Instance statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.pdb.info[connString,<user>,<password>,<service>,<database>]				
	Pluggable Databases (PDBs) information.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **database** - name of the database to query.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.pdb.discovery[connString,<user>,<password>,<service>]				
	List of PDBs. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.pga.stats[connString,<user>,<password>,<service>]				
	Program Global Area (PGA) statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.ping[connString,<user>,<password>,<service>]				

Key				
	Tests whether a connection to Oracle can be established.	0 - connection is broken (if there is any error presented including AUTH and configuration issues) 1 - connection is successful.	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.proc.stats[connString,<user>,<password>,<service>]				
	Processes statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.redolog.info[connString,<user>,<password>,<service>]				
	Log file information from the control file.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.sga.stats[connString,<user>,<password>,<service>]				
	System Global Area (SGA) statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.sessions.stats[connString,<user>,<password>,<service>,<lockMaxTime>]				
	Sessions statistics.	JSON object	**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.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.sys.metrics[connString,<user>,<password>,<service>,<duration>]				

Key				
	A set of system metric values.	JSON object	**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.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.sys.params[connString,<user>,<password>,<service>]				
	A set of system parameter values.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.ts.stats[connString,<user>,<password>,<service>,<tablespace>,<type>]				
	Tablespaces statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **tablespace** - name of the tablespace to query. Default (if left empty and `type` is set): - ”TEMP” (if `type` is set to ”TEMPORARY”); - ”USERS” (if `type` is set to ”PERMANENT”). **type** - type of the tablespace to query. Default (if `tablespace` is set): ”PERMANENT”.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.ts.discovery[connString,<user>,<password>,<service>]				
	List of tablespaces. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
oracle.user.info[connString,<user>,<password>,<service>,<username>]				

Key				
	List of tablespaces. Used for low-level discovery.	JSON object	**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.	This item is supported for the Oracle plugin. **user** parameter allows to append one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).
pgsql.autovacuum.count[uri,<username>,<password>,<dbName>]				
	The number of autovacuum workers.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.archive[uri,<username>,<password>,<dbName>]				
	Information about archived files.	JSON format	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.bgwriter[uri,<username>,<password>,<dbName>]				
	Combined number of checkpoints for the database cluster, broken down by checkpoint type.	JSON format	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
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.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.connections[uri,<username>,<password>,<dbName>]				
	Connections by type.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.custom.query[uri,<username>,<password>,queryName[,args...]]				
	Returns result of a custom query.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **queryName** - name of a custom query, must match SQL file name without an extension. **args**(optional) - arguments to pass to a query.	
pgsql.dbstat[uri,<username>,<password>,dbName]				
	Collects statistics per database. Used for low-level discovery.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
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.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.db.age[uri,<username>,<password>,dbName]				
	Age of the oldest FrozenXID of the database. Used for low-level discovery.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.db.bloating_tables[uri,<username>,<password>,<dbName>]				

Key				
	The number of bloating tables per database. Used for low-level discovery.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.db.discovery[uri,<username>,<password>,<dbName>]				
	List of the PostgreSQL databases. Used for low-level discovery.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.db.size[uri,<username>,<password>,dbName]				
	Database size in bytes. Used for low-level discovery.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.locks[uri,<username>,<password>,<dbName>]				
	Information about granted locks per database. Used for low-level discovery.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.oldest.xid[uri,<username>,<password>,<dbName>]				
	Age of the oldest XID.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.ping[uri,<username>,<password>,<dbName>]				
	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).	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.queries[uri,<username>,<password>,<dbName>,timePeriod]				
	Measures query execution time.	JSON object	**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).	This item is supported since Zabbix 5.0.22 for the PostgreSQL plugin.
pgsql.replication.count[uri,<username>,<password>]				
	The number of standby servers.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.replication.process[uri,<username>,<password>]				
	Flush lag, write lag and replay lag per each sender process.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	
pgsql.replication.process.discovery[uri,<username>,<password>]				
	Replication process name discovery.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	
pgsql.replication.recovery_role[uri,<username>,<password>]				
	Recovery status.	0 - master mode 1 - recovery is still in progress (standby mode)	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.

Key				
pgsql.replication.status[uri,<username>,<password>]				
	The status of replication.	0 - streaming is down 1 - streaming is up 2 - master mode	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.replication_lag.b[uri,<username>,<password>]				
	Replication lag in bytes.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.replication_lag.sec[uri,<username>,<password>]				
	Replication lag in seconds.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.uptime[uri,<username>,<password>,<dbName>]				
	PostgreSQL uptime in milliseconds.	Float	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
pgsql.wal.stat[uri,<username>,<password>,<dbName>]				
	WAL statistics.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	This item is supported since Zabbix 5.0.1 for the PostgreSQL plugin.
redis.config[connString,<password>,<pattern>]				
	Gets the configuration parameters of a Redis instance that match the pattern.	JSON - if a glob-style pattern was used single value - if a pattern did not contain any wildcard character	**connString** - URI or session name. **password** - Redis password. **pattern** - glob-style pattern (* by default).	This item is supported since Zabbix 4.4.5 for the Redis plugin.
redis.info[connString,<password>,<section>]				
	Gets the output of the INFO command.	JSON - output is serialized as JSON	**connString** - URI or session name. **password** - Redis password. **section** - section of information (*default* by default).	This item is supported since Zabbix 4.4.5 for the Redis plugin.
redis.ping[connString,<password>]				
	Test if 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)	**connString** - URI or session name. **password** - Redis password.	This item is supported since Zabbix 4.4.5 for the Redis plugin.
redis.slowlog.count[connString,<password>]				
	The number of slow log entries since Redis was started.	Integer	**connString** - URI or session name. **password** - Redis password.	This item is supported since Zabbix 4.4.5 for the Redis plugin.
smart.attribute.discovery				

Key				
	Returns a list of S.M.A.R.T. device attributes.	JSON object		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: {"{#NAME}": "/dev/sda cciss,2"} This item is supported for the Smart plugin since Zabbix 5.0.9.
smart.disk.discovery				
	Returns a list of S.M.A.R.T. devices.	JSON object		The following macros and their values are returned: {#NAME}, {#DISKTYPE}, {#MODEL}, {#SN}, {#PATH}, {#ATTRIBUTES}, {#RAIDTYPE}. HDD, SSD and NVME drive types are supported. If a drive does not belong to a RAID, {#RAIDTYPE} will be empty. {#NAME} will have an add-on in case of RAID, e.g: {"{#NAME}": "/dev/sda cciss,2"} This item is supported for the Smart plugin since Zabbix 5.0.9.
smart.disk.get[<path>,<raid_type>]				
	Returns all available properties of S.M.A.R.T. devices.	JSON object	**path** (since Zabbix 5.0.23) - disk path, the {#PATH} macro may be used as a value **raid_type** (since Zabbix 5.0.23) - 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/sda cciss,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. This item is supported for the Smart plugin since Zabbix 5.0.9.

Key				
systemd.unit.discovery[<type>]	List of systemd units and their details. Used for low-level discovery.	JSON object	**type** - possible values: *all*, *automount*, *device*, *mount*, *path*, *service* (default), *socket*, *swap*, *target*	This item is supported for the Systemd plugin on Linux platform only.
systemd.unit.get[unit name,<interface>]	Returns all properties of a systemd unit.	JSON object	**unit name** - unit name (you may want to use the {#UNIT.NAME} macro in item prototype to discover the name) **interface** - unit interface type, possible values: *Unit* (default), *Service*, *Socket*, *Device*, *Mount*, *Automount*, *Swap*, *Target*, *Path*	This item is supported for the Systemd plugin on Linux platform only. LoadState, ActiveState and UnitFileState for Unit interface are returned as text and integer: `"ActiveState":{"state":1,"text"` This item is supported since Zabbix 5.0.7.
systemd.unit.info[unit name,<property>,<interface>]	Systemd unit information.	String	**unit name** - unit name (you may want to use the {#UNIT.NAME} macro in item prototype to discover the name) **property** - unit property (e.g. ActiveState (default), LoadState, Description) **interface** - unit interface type (e.g. Unit (default), Socket, Service)	This item allows to retrieve a specific property from specific type of interface as described in dbus API. This item is supported for the Systemd plugin on Linux platform only. Examples: => sys-temd.unit.info[”{#UNIT.NAME}”] - collect active state (active, reloading, inactive, failed, activating, deactivating) info on discovered systemd units => sys-temd.unit.info[”{#UNIT.NAME}”,LoadStat - collect load state info on discovered systemd units => sys-temd.unit.info[mysqld.service,Id] - retrieve service technical name (mysqld.service) => sys-temd.unit.info[mysqld.service,Description - retrieve service description (MySQL Server) => sys-temd.unit.info[mysqld.service,ActiveEnter - retrieve the last time the service entered the active state (1562565036283903) => sys-temd.unit.info[dbus.socket,NConnections, - collect the number of connections from this socket unit

Key				
web.certificate.get[hostname,<port>,<address>]	Validates certificates and returns certificate details.	JSON object	**hostname** - can be either IP or DNS. May contain the URL scheme (*https* only), path (it will be ignored), and port. If a port is provided in both the first and the second parameters, their values must match. If address (the 3rd parameter) is specified, the hostname is only used for SNI and hostname verification. **port** - port number (default is 443 for HTTPS). **address** - can be either IP or DNS. If specified, it will be used for connection, and hostname (the 1st parameter) will be used for SNI, and host verification. In case, the 1st parameter is an IP and the 3rd parameter is DNS, the 1st parameter will be used for connection, and the 3rd parameter will be used for SNI and host verification.	This item turns unsupported if the resource specified in `host` does not exist or is unavailable or if TLS handshake fails with any error except an invalid certificate. Currently, AIA (Authority Information Access) X.509 extension, CRLs and OCSP (including OCSP stapling), Certificate Transparency, and custom CA trust store are not supported. This item is supported since Zabbix 5.0.15

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 by specifying the `--with-net-snmp` flag.

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.

Warning:
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.

Warning:
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 <host IP> 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
```

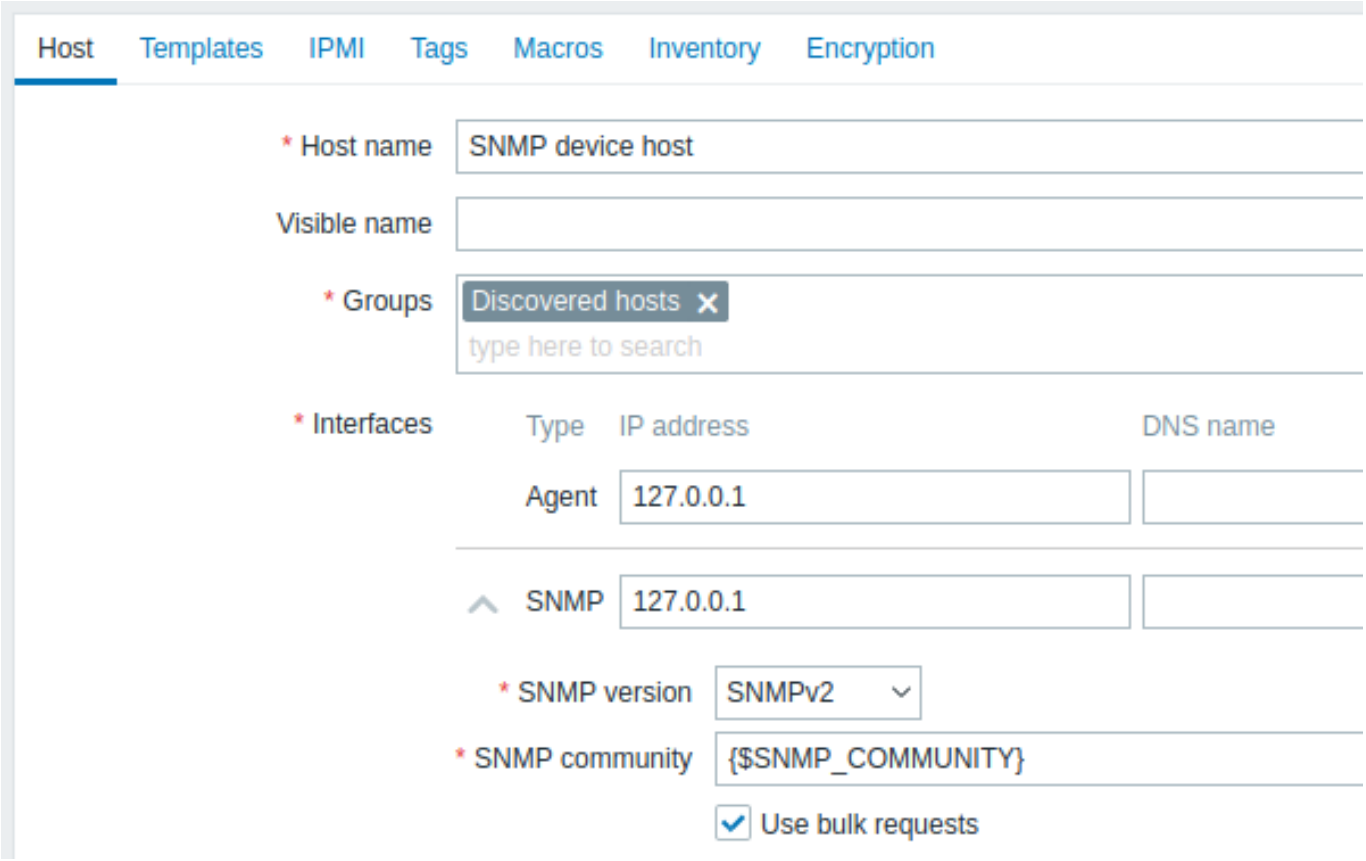
Again, the last number in the OID is the port number.

Note:
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_UINTEGER, 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	Select authentication protocol - *MD5* or *SHA*.
Authentication passphrase	Enter authentication passphrase. User macros are resolved in this field.
Privacy protocol	Select privacy protocol - *DES* or *AES*.
Privacy passphrase	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;

- (since Zabbix 5.0.31) SNMP interface availability will switch to red (unavailable).

Warning:
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.

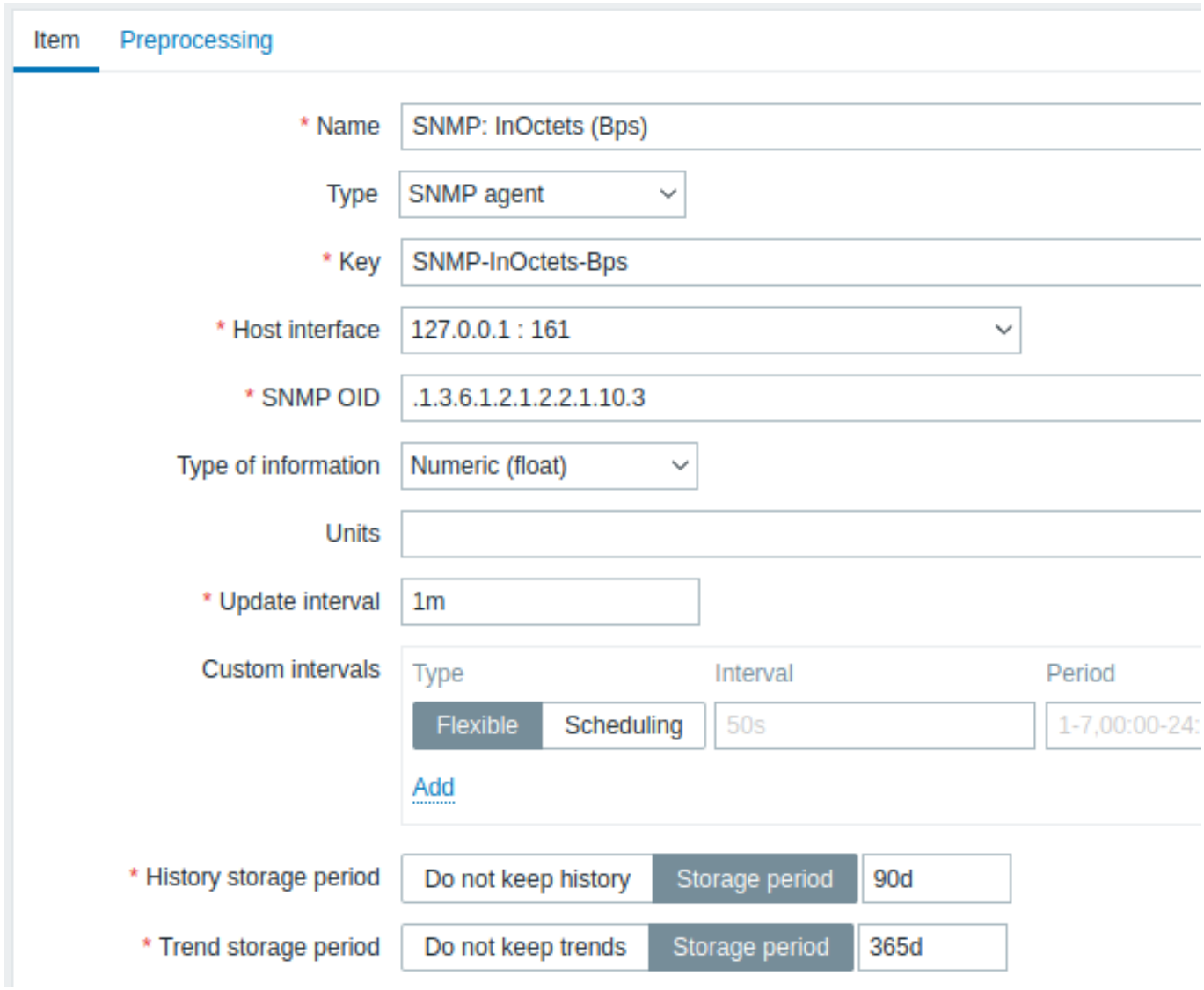
You can use one of the provided SNMP templates (*Template SNMP Device* and others) that will automatically add a set of items. However, the template may not be compatible with the host. Click on *Add* to save the host.

Step 3

Create an item for monitoring.

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:

Parameter	Description
OID	1.2.3.45.6.7.8.0 (or .1.2.3.45.6.7.8.0)
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:

Parameter	Description
OID	MIB::sysUpTime.0
Key	router.uptime
Value type	Float
Units	uptime
Preprocessing step: Custom multiplier	0.01

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>"]

Parameter	Description
OID of data	Main OID to use for data retrieval on the item.
index	Method of processing. Currently one method is supported: **index** – search for index and append it to the data OID
base OID of index	This OID will be looked up to get the index value corresponding to the string.
string to search for	The string to use for an exact match with a value when doing lookup. Case sensitive.

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 **1.3.6.1.2.1.2.2.1.1**, **ifIndex.0** is translated to **1.3.6.1.2.1.2.2.1.1.0**.

The table contains list of the special OIDs.

Special OID	Identifier	Description
ifIndex	1.3.6.1.2.1.2.2.1.1	A unique value for each interface.
ifDescr	1.3.6.1.2.1.2.2.1.2	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.
ifType	1.3.6.1.2.1.2.2.1.3	The type of interface, distinguished according to the physical/link protocol(s) immediately 'below' the network layer in the protocol stack.
ifMtu	1.3.6.1.2.1.2.2.1.4	The size of the largest datagram which can be sent / received on the interface, specified in octets.
ifSpeed	1.3.6.1.2.1.2.2.1.5	An estimate of the interface's current bandwidth in bits per second.
ifPhysAddress	1.3.6.1.2.1.2.2.1.6	The interface's address at the protocol layer immediately 'below' the network layer in the protocol stack.
ifAdminStatus	1.3.6.1.2.1.2.2.1.7	The current administrative state of the interface.
ifOperStatus	1.3.6.1.2.1.2.2.1.8	The current operational state of the interface.
ifInOctets	1.3.6.1.2.1.2.2.1.10	The total number of octets received on the interface, including framing characters.
ifInUcastPkts	1.3.6.1.2.1.2.2.1.11	The number of subnetwork-unicast packets delivered to a higher-layer protocol.
ifInNUcastPkts	1.3.6.1.2.1.2.2.1.12	The number of non-unicast (i.e., subnetwork- broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol.
ifInDiscards	1.3.6.1.2.1.2.2.1.13	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.
ifInErrors	1.3.6.1.2.1.2.2.1.14	The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol.

Special OID	Identifier	Description
ifInUnknownProtos	1.3.6.1.2.1.2.2.1.15	The number of packets received via the interface which were discarded because of an unknown or unsupported protocol.
ifOutOctets	1.3.6.1.2.1.2.2.1.16	The total number of octets transmitted out of the interface, including framing characters.
ifOutUcastPkts	1.3.6.1.2.1.2.2.1.17	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.
ifOutNUcastPkts	1.3.6.1.2.1.2.2.1.18	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.
ifOutDiscards	1.3.6.1.2.1.2.2.1.19	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.
ifOutErrors	1.3.6.1.2.1.2.2.1.20	The number of outbound packets that could not be transmitted because of errors.
ifOutQLen	1.3.6.1.2.1.2.2.1.21	The length of the output packet queue (in packets).

3 MIB files

Introduction

MIB stands for the Management Information Base. MIB files allow to use textual representation of an OID (Object Identifier). It is possible to use raw OIDs when monitoring SNMP devices with Zabbix, but if you feel more comfortable using textual representation, you need to install MIB files.

For example,

`ifHCOutOctets`

is textual representation of the OID

`1.3.6.1.2.1.31.1.1.1.10`

Installing MIB files

On Debian-based systems:

```
# apt install snmp-mibs-downloader
# download-mibs
```

On RedHat-based systems:

```
# yum install net-snmp-libs
```

Enabling MIB files

On RedHat-based systems, MIB files should be enabled by default. On Debian-based systems, you have to edit the file `/etc/snmp/snmp.conf` and comment out the line that says `mibs :`

```
# 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
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.:

```
# systemctl restart zabbix-server
```

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:

```
# wget ftp://ftp.cisco.com/pub/mibs/v2/CISCO-SMI.my -P /tmp
# 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/lo
# 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.

Attention:
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 a 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:

Key		
Description	Return value	Comments
snmptrap[regexp] Catches all SNMP traps that match the regular expression specified in **regexp**. If regexp is unspecified, catches any trap.	SNMP trap	This item can be set only for SNMP interfaces. This item is supported since Zabbix **2.0.0.** *Note*: Starting with Zabbix 2.0.5, user macros and global regular expressions are supported in the parameter of this item key.
snmptrap.fallback Catches all SNMP traps that were not caught by any of the snmptrap[] items for that interface.	SNMP trap	This item can be set only for SNMP interfaces. This item is supported since Zabbix **2.0.0.**

Note:
Multiline 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.

Note:
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):

```
StartSNMPTrapper=1
SNMPTrapperFile=[TRAP FILE]
```

Warning:
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.

Note:
For the best performance, SNMPTT should be configured as a daemon using **snmptthandler-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
```

Warning:
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.

Attention:
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:


```
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:

- add the perl script to snmptrapd configuration file (snmptrapd.conf), e.g.:

```
perl do "[FULL PATH TO PERL RECEIVER SCRIPT]";
```

- configure the receiver, e.g:

```
$SNMPTrapperFile = '[TRAP FILE]';
$DateTimeFormat = '[DATE TIME FORMAT]';
```

Note:
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
```

Warning:
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
```

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 define 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 snmptt
```

3. **snmptt.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 $aA $ar
```

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' 6 33 '55' .1.3.6.1.6.3.1.1.5.3 s "tests
```

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
```

> **Note:**
> 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

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:

```
# StartIPMIPollers=0
```

Uncomment it and set poller count to, say, 3, so that it reads:

```
StartIPMIPollers=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.

Item key ▲	Description	Return value	Comments
ipmi.get	IPMI-sensor related information.	JSON object	This item can be used for the discovery of IPMI sensors. Supported since Zabbix 5.0.0.

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 rescan 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:

```
$ 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:
8358:20130318:111122.170 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:15 id:'CPU Therm Trip' read
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'System Event Log' re
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'PhysicalSecurity' re
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'IPMI Watchdog' readi
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'Power Unit Stat' rea
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'P1 Therm Ctrl %' rea
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'P1 Therm Margin' rea
```

```
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:13 id:'System Fan 2' readin
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:13 id:'System Fan 3' readin
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'P1 Mem Margin' readi
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'Front Panel Temp' re
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:15 id:'Baseboard Temp' read
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:9 id:'BB +5.0V' reading_typ
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'BB +3.3V STBY' readi
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:9 id:'BB +3.3V' reading_typ
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'BB +1.5V P1 DDR3' re
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'BB +1.1V P1 Vccp' re
8358:20130318:111122.174 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'BB +1.05V PCH' readi
```

To decode IPMI sensor types and states, a copy of IPMI 2.0 specifications is available (please note that no further updates to the IPMI specification are planned).

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 sometimes are 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 "PowerOff/Power Down". In other words if the least significant bit is 1, then server is powered off. To test this bit a function **band** with mask 1 can be used. The trigger expression could be like

```
{www.example.com:Power Unit Stat.band(#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/notsupported, 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]
net.tcp.service[http]
net.tcp.service.perf[http,,8080]
```

`net.udp.service.perf[ntp]`

> **Note:**
> *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

Key	Description	Return value	Parameters	Comments
icmpping[<target>,<packets>,<interval>,<size>,<timeout>]	Host accessibility by ICMP ping.	0 - ICMP ping fails 1 - ICMP ping successful	**target** - host IP or DNS name **packets** - number of packets **interval** - time between successive packets in milliseconds **size** - packet size in bytes **timeout** - timeout in milliseconds	Example: => icmpping[,4] → if at least one packet of the four is returned, the item will return 1. See also: table of default values.
icmppingloss[<target>,<packets>,<interval>,<size>,<timeout>]	Percentage of lost packets.	Float.	**target** - host IP or DNS name **packets** - number of packets **interval** - time between successive packets in milliseconds **size** - packet size in bytes **timeout** - timeout in milliseconds	See also: table of default values.
icmppingsec[<target>,<packets>,<interval>,<size>,<timeout>,<mode>]				

Key				
	ICMP ping response time (in seconds).	Float.	**target** - host IP or DNS name **packets** - number of packets **interval** - time between successive packets in milliseconds **size** - packet size in bytes **timeout** - timeout in milliseconds **mode** - possible values: *min*, *max*, *avg* (default)	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.
net.tcp.service[service,<ip>,<port>]				

Key				
	Checks if service is running and accepting TCP connections.	0 - service is down 1 - service is running	**service** - possible values: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details) **ip** - IP address or DNS name (by default host IP/DNS is used) **port** - port number (by default standard service port number is used).	Example: => 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,<ip>,port] for checks like these. *https* and *telnet* services are supported since Zabbix 2.0.
net.tcp.service.perf[service,<ip>,<port>]				

Key				
	Checks performance of TCP service.	Float. 0.000000 - service is down seconds - the number of seconds spent while connecting to the service	**service** - possible values: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details) **ip** - IP address or DNS name (by default, host IP/DNS is used) **port** - port number (by default standard service port number is used).	Example: => net.tcp.service.perf[ssh] → can be used to test the speed of initial response from SSH server. 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,<ip>, for checks like these. *https* and *telnet* services are supported since Zabbix 2.0. Called tcp_perf before Zabbix 2.0.
net.udp.service[service,<ip>,<port>]				
	Checks if service is running and responding to UDP requests.	0 - service is down 1 - service is running	**service** - possible values: *ntp* (see details) **ip** - IP address or DNS name (by default host IP/DNS is used) **port** - port number (by default standard service port number is used).	Example: => net.udp.service[ntp,,45] → can be used to test the availability of NTP service on UDP port 45. This item is supported since Zabbix 3.0, but *ntp* service was available for net.tcp.service[] item in prior versions.
net.udp.service.perf[service,<ip>,<port>]				

Key				
	Checks performance of UDP service.	Float. 0.000000 - service is down seconds - the number of seconds spent waiting for response from the service	**service** - possible values: *ntp* (see details) **ip** - IP address or DNS name (by default, host IP/DNS is used) **port** - port number (by default standard service port number is used).	Example: => net.udp.service.perf[ntp] → can be used to test response time from NTP service. This item is supported since Zabbix 3.0, but *ntp* service was available for net.tcp.service[] item in prior versions.

Attention:
For SourceIP support in LDAP simple checks (e.g. `net.tcp.service[ldap]`), OpenLDAP version 2.6.1 or above is required. SourceIP is supported in LDAP simple checks since Zabbix 5.0.21.

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:

Parameter	Unit	Description	Fping's flag	Defaults set by		Allowed limits by Zabbix	
				fping	**Zabbix**	**min**	**max**
packets	number	number of request packets to a target	-C		3	1	10000

Parameter	Unit	Description	Fping's flag	Defaults set by	Allowed limits by Zabbix	
interval	milliseconds	time to wait between successive packets	-p	1000	20	unlimited
size	bytes	packet size in bytes 56 bytes on x86, 68 bytes on x86_64	-b	56 or 68	24	65507
timeout	milliseconds	**fping v3.x** - timeout to wait after last packet sent, affected by *-C* flag **fping v4.x** - individual timeout for each packet	-t	**fping v3.x** - 500 **fping v4.x** - inherited from *-p* flag, but not more than 2000	50	unlimited

In addition Zabbix uses fping options *-i interval ms* (do not mix up with the item parameter *interval* 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.

Since Zabbix 5.0.4 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:
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.

1 VMware monitoring item keys

Item keys

The table provides details on the simple checks that can be used to monitor VMware environments.

Parameters without angle brackets are mandatory. Parameters marked with angle brackets **< >** are optional.

Key	Description	Return value	Parameters	Comments
vmware.cluster.discovery[url]	Discovery of VMware clusters.	JSON object	**url** - VMware service URL	
vmware.cluster.status[url,name]	VMware cluster status.	Integer: 0 - gray; 1 - green; 2 - yellow; 3 - red	**url** - VMware service URL **name** - VMware cluster name	
vmware.datastore.discovery[url]				

Key				
	Discovery of VMware datastores.	JSON object	**url** - VMware service URL	
vmware.datastore.hv.list[url,datastore]				
	List of datastore hypervisors.	JSON object	**url** - VMware service URL **datastore** - datastore name	
vmware.datastore.read[url,datastore,<mode>]				
	Amount of time for a read operation from the datastore (milliseconds).	Integer [2]	**url** - VMware service URL **datastore** - datastore name **mode** - latency (average value, default), maxlatency (maximum value)	
vmware.datastore.size[url,datastore,<mode>]				
	VMware datastore space in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**url** - VMware service URL **datastore** - datastore name **mode** - possible values: total (default), free, pfree (free, percentage), uncommitted	
vmware.datastore.write[url,datastore,<mode>]				
	Amount of time for a write operation to the datastore (milliseconds).	Integer [2]	**url** - VMware service URL **datastore** - datastore name **mode** - latency (average value, default), maxlatency (maximum value)	
vmware.dc.discovery[url]				
	Discovery of VMware datacenters.	JSON object	**url** - VMware service URL	This item is supported since Zabbix 5.0.3.
vmware.eventlog[url,<mode>]				

Key				
	VMware event log.	Log	**url** - VMware service URL **mode** - *all* (default), *skip* - skip processing of older data	There must be only one vmware.eventlog[] item key per URL. See also: example of filtering VMware event log records.
vmware.fullname[url]				
	VMware service full name.	String	**url** - VMware service URL	
vmware.hv.cluster.name[url,uuid]				
	VMware hypervisor cluster name.	String	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.cpu.usage[url,uuid]				
	VMware hypervisor processor usage (Hz).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.datacenter.name[url,uuid]				
	VMware hypervisor datacenter name.	String	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.datastore.discovery[url,uuid]				
	Discovery of VMware hypervisor datastores.	JSON object	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.datastore.list[url,uuid]				
	List of VMware hypervisor datastores.	JSON object	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.datastore.read[url,uuid,datastore,<mode>]				
	Average amount of time for a read operation from the datastore (milliseconds).	Integer [2]	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier **datastore** - datastore name **mode** - latency (default)	

Key				
vmware.hv.datastore.size[url,uuid,datastore,<mode>]				
	VMware datastore space in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier **datastore** - datastore name **mode** - possible values: total (default), free, pfree (free, percentage), uncommitted	Available since Zabbix versions 3.0.6, 3.2.2
vmware.hv.datastore.write[url,uuid,datastore,<mode>]				
	Average amount of time for a write operation to the datastore (milliseconds).	Integer [2]	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier **datastore** - datastore name **mode** - latency (default)	
vmware.hv.discovery[url]				
	Discovery of VMware hypervisors.	JSON object	**url** - VMware service URL	
vmware.hv.fullname[url,uuid]				
	VMware hypervisor name.	String	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.hw.cpu.freq[url,uuid]				
	VMware hypervisor processor frequency (Hz).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.hw.cpu.model[url,uuid]				
	VMware hypervisor processor model.	String	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.hw.cpu.num[url,uuid]				
	Number of processor cores on VMware hypervisor.	Integer	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	

Key				
vmware.hv.hw.cpu.threads[url,uuid]	Number of processor threads on VMware hypervisor.	Integer	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.hw.memory[url,uuid]	VMware hypervisor total memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.hw.model[url,uuid]	VMware hypervisor model.	String	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.hw.uuid[url,uuid]	VMware hypervisor BIOS UUID.	String	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.hw.vendor[url,uuid]	VMware hypervisor vendor name.	String	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.maintenance[url,uuid]	VMware hypervisor maintenance status.	Integer	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	Returns '0' - not in maintenance or '1' - in maintenance This item is supported since Zabbix 5.0.18.
vmware.hv.memory.size.ballooned[url,uuid]	VMware hypervisor ballooned memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.memory.used[url,uuid]	VMware hypervisor used memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.network.in[url,uuid,<mode>]				

Key				
	VMware hypervisor network input statistics (bytes per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier **mode** - bps (default)	
vmware.hv.network.out[url,uuid,<mode>]				
	VMware hypervisor network output statistics (bytes per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier **mode** - bps (default)	
vmware.hv.perfcounter[url,uuid,path,<instance>]				
	VMware hypervisor performance counter value.	Integer [2]	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier **path** - performance counter path [1] **instance** - performance counter instance. Use empty instance for aggregate values (default)	Available since Zabbix versions 2.2.9, 2.4.4
vmware.hv.sensor.health.state[url,uuid]				
	VMware hypervisor health state rollup sensor.	Integer: 0 - gray; 1 - green; 2 - yellow; 3 - red	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	The item might not work in the VMware vSphere 6.5 and newer, because VMware has deprecated the *VMware Rollup Health State* sensor.
vmware.hv.sensors.get[url,uuid]				
	VMware hypervisor HW vendor state sensors.	JSON	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	This item is supported since Zabbix 5.0.18.
vmware.hv.status[url,uuid]				
	VMware hypervisor status.	Integer: 0 - gray; 1 - green; 2 - yellow; 3 - red	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	Uses host system overall status property since Zabbix 2.2.16, 3.0.6, 3.2.2

Key				
vmware.hv.uptime[url,uuid]	VMware hypervisor uptime (seconds).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.version[url,uuid]	VMware hypervisor version.	String	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.hv.vm.num[url,uuid]	Number of virtual machines on VMware hypervisor.	Integer	**url** - VMware service URL **uuid** - VMware hypervisor global unique identifier	
vmware.version[url]	VMware service version.	String	**url** - VMware service URL	
vmware.vm.cluster.name[url,uuid]	VMware virtual machine name.	String	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier	
vmware.vm.cpu.num[url,uuid]	Number of processors on VMware virtual machine.	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier	
vmware.vm.cpu.ready[url,uuid]	Time (in milliseconds) 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 (%).	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier	Available since Zabbix version 3.0.0
vmware.vm.cpu.usage[url,uuid]				

Key			
	VMware virtual machine processor usage (Hz).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier
vmware.vm.datacenter.name[url,uuid]			
	VMware virtual machine datacenter name.	String	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier
vmware.vm.discovery[url]			
	Discovery of VMware virtual machines.	JSON object	**url** - VMware service URL
vmware.vm.hv.name[url,uuid]			
	VMware virtual machine hypervisor name.	String	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier
vmware.vm.memory.size[url,uuid]			
	VMware virtual machine total memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier
vmware.vm.memory.size.ballooned[url,uuid]			
	VMware virtual machine ballooned memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier
vmware.vm.memory.size.compressed[url,uuid]			
	VMware virtual machine compressed memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier
vmware.vm.memory.size.private[url,uuid]			
	VMware virtual machine private memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier
vmware.vm.memory.size.shared[url,uuid]			
	VMware virtual machine shared memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier
vmware.vm.memory.size.swapped[url,uuid]			

Key			
	VMware virtual machine swapped memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier
vmware.vm.memory.size.usage.guest[url,uuid]			
	VMware virtual machine guest memory usage (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier
vmware.vm.memory.size.usage.host[url,uuid]			
	VMware virtual machine host memory usage (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier
vmware.vm.net.if.discovery[url,uuid]			
	Discovery of VMware virtual machine network interfaces.	JSON object	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier
vmware.vm.net.if.in[url,uuid,instance,<mode>]			
	VMware virtual machine network interface input statistics (bytes/packets per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier **instance** - network interface instance **mode** - bps (default)/pps - bytes/packets per second
vmware.vm.net.if.out[url,uuid,instance,<mode>]			
	VMware virtual machine network interface output statistics (bytes/packets per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier **instance** - network interface instance **mode** - bps (default)/pps - bytes/packets per second
vmware.vm.perfcounter[url,uuid,path,<instance>]			

Key				
	VMware virtual machine performance counter value.	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier **path** - performance counter path [1] **instance** - performance counter instance. Use empty instance for aggregate values (default)	Available since Zabbix versions 2.2.9, 2.4.4
vmware.vm.powerstate[url,uuid]				
	VMware virtual machine power state.	Integer: 0 - poweredOff; 1 - poweredOn; 2 - suspended	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier	
vmware.vm.storage.committed[url,uuid]				
	VMware virtual machine committed storage space (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier	
vmware.vm.storage.uncommitted[url,uuid]				
	VMware virtual machine uncommitted storage space (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier	
vmware.vm.storage.unshared[url,uuid]				
	VMware virtual machine unshared storage space (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier	
vmware.vm.uptime[url,uuid]				
	VMware virtual machine uptime (seconds).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier	
vmware.vm.vfs.dev.discovery[url,uuid]				
	Discovery of VMware virtual machine disk devices.	JSON object	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier	
vmware.vm.vfs.dev.read[url,uuid,instance,<mode>]				

Key				
	VMware virtual machine disk device read statistics (bytes/operations per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier **instance** - disk device instance **mode** - bps (default)/ops - bytes/operations per second	
vmware.vm.vfs.dev.write[url,uuid,instance,<mode>]				
	VMware virtual machine disk device write statistics (bytes/operations per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier **instance** - disk device instance **mode** - bps (default)/ops - bytes/operations per second	
vmware.vm.vfs.fs.discovery[url,uuid]				
	Discovery of VMware virtual machine file systems.	JSON object	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier	VMware Tools must be installed on the guest virtual machine.
vmware.vm.vfs.fs.size[url,uuid,fsname,<mode>]				
	VMware virtual machine file system statistics (bytes/percentages).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine global unique identifier **fsname** - file system name **mode** - total/free/used/pfree/pused	VMware Tools must be installed on the guest virtual machine.

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.

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

Attention:
The size limit of a monitored log file depends on large file support.

Configuration

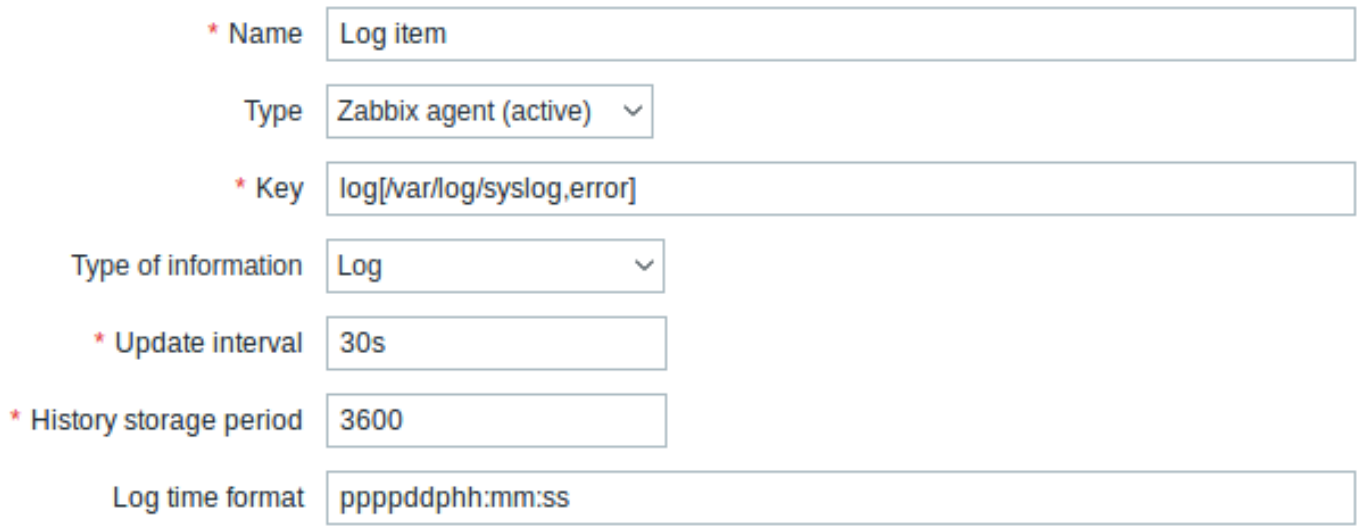
Verify agent parameters

Make sure that in the agent configuration file:

- 'Hostname' parameter matches the host name in the frontend
- Servers in the 'ServerActive' parameter are specified for the processing of active checks

Item configuration

Configure a log monitoring item.



All mandatory input fields are marked with a red asterisk.

Specifically for log monitoring items you enter:

Type	Select **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	Select: For log[] or logrt[] items - `Log`; For log.count[] or logrt.count[] items - `Numeric (unsigned)`. If optionally using the `output` parameter, you may 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. 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 be set to 0 in Unix time, representing January 1, 1970. For example, consider the following line from the Zabbix agent log file: " 23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211)." It begins with six character positions for PID, followed by date, time, and the rest of the message. The log time format for this line would be "ppppp:yyyyMMdd:hhmmss". Note that "p" and ":" characters are placeholders and can be any character except "yMdhms".

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 MD5 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 MD5 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 receives 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 agent BufferSize 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 or DebugLevel=5.
- Searching for a question mark using a regular expression, e.g. `\?` may result in false positives if the text file contains NUL symbols, as those are replaced with "?" by Zabbix to continue processing the line until the newline character.

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 subgroup 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:

```
Fr Feb 07 2014 11:07:36.6690 */ Thread Id 1400 (GLEWF) large result
buffer allocation - /Length: 437136/Entries: 5948/Client Ver: >=10/RPC
ID: 41726453/User: AUser/Form: CFG:ServiceLevelAgreement
```

The reason why Zabbix will return only the number is because 'output' here is defined by **\1** referring to the first and only subgroup of interest: **([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.

Warning:
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 10*'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.

> **Warning:**
> 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„„„„„/home/zabbix50/agent_private] 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 you can create calculations on the basis of other items.

Thus, calculated items are a way of creating virtual data sources. The values will be periodically calculated based on an arithmetical expression. All calculations are done by the Zabbix server - nothing related to calculated items is performed on Zabbix agents or proxies.

The resulting data will be stored in the Zabbix database as for any other item - this means storing both history and trend values for fast graph generation. Calculated items may be used in trigger expressions, referenced by macros or other entities same as any other item type. Comparison to strings is allowed in calculated items since Zabbix 5.0.

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 formula in any way.

The correct syntax of a simple formula is:

`func(<key>|<hostname:key>,<parameter1>,<parameter2>,...)`

Where:

ARGUMENT	DEFINITION
func	One of the functions supported in trigger expressions: last, min, max, avg, count, etc
key	The key of another item whose data you want to use. It may be defined as **key** or **hostname:key**. *Note:* Putting the whole key in double quotes ("...") is strongly recommended to avoid incorrect parsing because of spaces or commas within the key. If there are also quoted parameters within the key, those double quotes must be escaped by using the backslash (\). See **Example 5** below.
parameter(s)	Function parameter(s), if required.

Note:
All items that are referenced from the calculated item formula must exist and be collecting data (exceptions in functions and unsupported items). Also, if you change the item key of a referenced item, you have to manually update any formulas using that key.

Attention:
User macros in the formula will be expanded if used to reference a function 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. Note that the syntax is slightly different, however 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.

Note:
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)*.

A calculated item may become unsupported in several cases:

1. referenced item(s)
    - is not found
    - is disabled
    - belongs to a disabled host
    - is not supported (see exceptions in functions and unsupported items, Expressions with unsupported items and unknown values and Operators)
2. no data to calculate a function
3. division by zero
4. incorrect syntax used

Support for calculated items was introduced in Zabbix 1.8.1.
Starting from Zabbix 3.2 calculated items in some cases may involve unsupported items as described in functions and unsupported items, Expressions with unsupported items and unknown values and Operators.

Usage examples

Example 1

Calculating percentage of free disk space on '/'.

Use of function **last**:

```
100*last("vfs.fs.size[/,free]")/last("vfs.fs.size[/,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**:

```
avg("Zabbix Server:zabbix[wcache,values]",600)
```

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:

```
last("net.if.in[eth0,bytes]")+last("net.if.out[eth0,bytes]")
```

Example 4

Calculating percentage of incoming traffic.

More complex expression:

```
100*last("net.if.in[eth0,bytes]")/(last("net.if.in[eth0,bytes]")+last("net.if.out[eth0,bytes]"))
```

Example 5

Using aggregated items correctly within a calculated item.

Take note of how double quotes are escaped within the quoted key:

```
last("grpsum[\"video\",\"net.if.out[eth0,bytes]\",\"last\"]") / last("grpsum[\"video\",\"nginx_stat.sh[act
```

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.

Note:
Internal checks are processed by Zabbix pollers.

Performance

Using some internal items may negatively affect performance. These items are:

- zabbix[host,,items]
- zabbix[host,,items_unsupported]
- zabbix[hosts]
- zabbix[items]
- zabbix[items_unsupported]
- zabbix[queue,,]
- zabbix[requiredperformance]
- zabbix[stats,,,queue,,]
- zabbix[triggers]

The System information and Queue frontend sections are also affected.

Supported checks

- Parameters without angle brackets are mandatory and must be used *as is* (for example, ”host” and ”available” in zabbix[host,<type>,available]).
- Parameters with angle brackets **< >** must be replaced with a valid value. If a parameter has a default value, it can be omitted.
- Values for items and item parameters labeled ”not supported on proxy” can only be retrieved if the host is monitored by server. Conversely, values ”not supported on server” can only be retrieved if the host is monitored by proxy.

Key	Description	Return value	Comments
zabbix[boottime]	Startup time of Zabbix server or Zabbix proxy process in seconds.	Integer.	
zabbix[history]			

Key			
	Number of values stored in the HISTORY table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! *(not supported on proxy)*
zabbix[history_log]			
	Number of values stored in the HISTORY_LOG table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! *(not supported on proxy)*
zabbix[history_str]			
	Number of values stored in the HISTORY_STR table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! *(not supported on proxy)*
zabbix[history_text]			
	Number of values stored in the HISTORY_TEXT table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! *(not supported on proxy)*
zabbix[history_uint]			
	Number of values stored in the HISTORY_UINT table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! This item is supported since Zabbix 1.8.3. *(not supported on proxy)*
zabbix[host„items]			
	Number of enabled items (supported and not supported) on the host.	Integer.	This item is supported since Zabbix 3.0.0.

Key			
zabbix[host„items_unsupported]	Number of enabled unsupported items on the host.	Integer.	This item is supported since Zabbix 3.0.0.
zabbix[host„maintenance]	Current mainte-nance status of a host.	0 - host in normal state, 1 - host in mainte-nance with data collection, 2 - host in mainte-nance without data collection.	This item is always processed by Zabbix server regardless of host location (on server or proxy). The proxy will not receive this item with configu-ration data. The second parameter must be empty and is reserved for future use.
zabbix[host,discovery,interfaces]	Details of all configured interfaces of the host in Zabbix frontend.	JSON object.	This item can be used in low-level discovery. This item is supported since Zabbix 3.4.0. *(not supported on proxy)*
zabbix[host,<type>,available]			

Key			
	Availability of a particular type of checks on the host. The value of this item corresponds to availability icons in the host list.	0 - not available, 1 - available, 2 - unknown.	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.
zabbix[hosts]	Number of monitored hosts.	Integer.	
zabbix[items]	Number of enabled items (supported and not supported).	Integer.	
zabbix[items_unsupported]	Number of not supported items.	Integer.	
zabbix[java„<param>]	Information about Zabbix Java gateway.	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”.	Valid values for **param** are: *ping*, *version* Second parameter must be empty and is reserved for future use.

Key			
zabbix[lld_queue]			
	Count of values enqueued in the low-level discovery processing queue.	Integer.	This item can be used to monitor the low-level discovery processing queue length. This item is supported since Zabbix 4.2.0.
zabbix[preprocessing_queue]			
	Count of values enqueued in the prepro-cessing queue.	Integer.	This item can be used to monitor the prepro-cessing queue length. This item is supported since Zabbix 3.4.0.
zabbix[process,<type>,<mode>,<state>]			

Key				
	Time a particular Zabbix process or a group of processes (identified by <type> and <mode>) spent in <state> in percentage. It is calculated for the last minute only. If <mode> is Zabbix process number that is not running (for example, with 5 pollers running <mode> is specified to be 6), such an item will turn into unsupported state. Minimum and maximum refers to the usage percentage for a single process. So if in a group of 3 pollers usage percentages per process were 2, 18 and 66, min would return 2 and max would return 66. Processes report what they are doing in shared memory and the self-monitoring process summarizes that data		Percentage of time. Float.	Supported **types** of server processes: *alert manager*, *alert syncer*, *alerter*, *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* Supported **types** of proxy processes: *configuration syncer*, *data sender*, *discoverer*, *heartbeat sender*, *history syncer*, *housekeeper*, *http poller*, *icmp pinger*, *ipmi manager*, *ipmi poller*, *java poller*, *poller*, *preprocessing manager*, *preprocessing worker*,

Key			
zabbix[proxy,<name>,<param>]	Information about Zabbix proxy.	Integer.	**name**: proxy name Valid values for **param** are: *lastaccess* - timestamp of last heart beat 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: => zabbix[proxy,”Germany”,lastacc `fuzzytime()` trigger 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]			

Key			
	Number of values in the proxy history table waiting to be sent to the server.	Integer.	*(not supported on server)*
zabbix[queue,<from>,<to>]			
	Number of monitored items in the queue which are delayed at least by <from> seconds but less than by <to> seconds.	Integer.	**from** - default: 6 seconds **to** - default: infinity Time-unit symbols (s,m,h,d,w) are supported for these parameters.
zabbix[rcache,<cache>,<mode>]			
	Availability statistics of Zabbix configuration cache.	Integer (for size); float (for percentage).	**cache**: *buffer* 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 *pused* mode is supported since Zabbix 4.0.0.
zabbix[requiredperformance]			
	Required performance of Zabbix server or Zabbix proxy, in new values per second expected.	Float.	Approximately correlates with "Required server performance, new values per second" in *Reports* → *System information*.
zabbix[stats,<ip>,<port>]			

Key				
	Remote Zabbix server or proxy internal metrics.		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) 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. Supported since 4.2.0.
zabbix[stats,<ip>,<port>,queue,<from>,<to>]				

Key			
	Remote Zabbix server or proxy internal queue metrics (see `zabbix[queue,<from>,<to>]`).	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) **from** - delayed by at least (default is 6 seconds) **to** - delayed by at most (default is infinity) Note that the stats request will only be accepted from the addresses listed in the 'StatsAllowedIP' server/proxy parameter on the target instance. Supported since 4.2.0.
zabbix[trends]	Number of values stored in the TRENDS table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! *(not supported on proxy)*
zabbix[trends_uint]			

Key			
	Number of values stored in the TRENDS_UINT table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! This item is supported since Zabbix 1.8.3. *(not supported on proxy)*
zabbix[triggers]			
	Number of enabled triggers in Zabbix database, with all items enabled on enabled hosts.	Integer.	*(not supported on proxy)*
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>]			

Key				
		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.
	Cache	**Mode**			
	values	all *(default)*	Total number of values processed by Zabbix server or Zabbix proxy, except unsupported items.	Integer.	Counter. You may use this key with the *Change per second* preprocess-ing step in order to get values per second statistics.
		float	Number of processed float values.	Integer.	Counter.
		uint	Number of processed unsigned integer values.	Integer.	Counter.
		str	Number of processed charac-ter/string values.	Integer.	Counter.
		log	Number of processed log values.	Integer.	Counter.

Key					
		text	Number of processed text values.	Integer.	Counter.
		not supported	Number of times item processing resulted in item becoming unsupported or keeping that state.	Integer.	Counter.
	history	pfree *(default)*	Percentage of free history buffer.	Float.	History cache is used to store item values. A low number indicates performance problems on the database side.
		free	Size of free history buffer.	Integer.	
		total	Total size of history buffer.	Integer.	
		used	Size of used history buffer.	Integer.	
		pused	Percentage of used history buffer.	Float.	*pused* mode is supported since Zabbix 4.0.0.
	index	pfree *(default)*	Percentage of free history index buffer.	Float.	History index cache is used to index values stored in history cache. *Index* cache is supported since Zabbix 3.0.0.
		free	Size of free history index history buffer.	Integer.	
		total	Total size of history index history buffer.	Integer.	
		used	Size of used history index history buffer.	Integer.	

Key					
		pused	Percentage of used history index buffer.	Float.	*pused* mode is supported since Zabbix 4.0.0.
	trend	pfree *(default)*	Percentage of free trend cache.	Float.	Trend cache stores aggregate for the current hour for all items that receive data. *(not supported on proxy)*
		free	Size of free trend buffer.	Integer.	*(not supported on proxy)*
		total	Total size of trend buffer.	Integer.	*(not supported on proxy)*
		used	Size of used trend buffer.	Integer.	*(not supported on proxy)*
		pused	Percentage of used trend buffer.	Float.	*(not supported on proxy)* *pused* mode is supported since Zabbix 4.0.0.

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 (libssh or libssh2). See also: Requirements.

Attention:
Starting with RHEL/CentOS 8, only libssh is supported. For other distributions, libssh is suggested over libssh2.

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 libssh or libssh2 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 the full path to the folder where the public and private keys will be located:

```
SSHKeyLocation=/home/zabbix/.ssh
```

Save the file and restart Zabbix server afterwards.

The path */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 an ssh-keygen command inside the home directory.

Usually installation packages of Zabbix server from different OS distributions create the *zabbix* user account with a home directory elsewhere, for example, */var/lib/zabbix* (as for system accounts).

Before generating the keys, you could reallocate the home directory to */home/zabbix*, so that it corresponds with the `SSHKeyLocation` Zabbix server configuration parameter mentioned above.

> **Note:**
> The following steps can be skipped if *zabbix* account has been added manually according to the installation section. In such a case the home directory for the *zabbix* account is most likely already */home/zabbix*.

To change the home directory of the *zabbix* user account, all working processes which are using it have to be stopped:

```
systemctl stop zabbix-agent
systemctl stop zabbix-server
```

To change the home directory location with an attempt to move it (if it exists) the following command should be executed:

```
usermod -m -d /home/zabbix zabbix
```

It is also possible that a home directory did not exist in the old location (e.g., in the CentOS), so it should be created at the new location. A safe attempt to do that is:

```
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:

```
chown zabbix:zabbix /home/zabbix
chmod 700 /home/zabbix
```

Previously stopped processes can now be started again:

```
systemctl start zabbix-agent
systemctl start zabbix-server
```

Now, the steps to generate the public and private keys can be performed with the following commands (for better readability, command prompts are commented out):

```
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):
/home/zabbix/.ssh/id_rsa
##### Enter passphrase (empty for no passphrase):
<Leave empty>
##### Enter same passphrase again:
<Leave empty>
##### 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         |
##### | .      o        |
##### |+      . S       |
##### |.+    o =        |
##### |E .    * =       |
##### |=o . ..* .       |
##### |... oo.o+        |
##### +-----------------+
```

Note:
The public and private keys (*id_rsa.pub* and *id_rsa*) have been generated by default in the */home/zabbix/.ssh* directory, which corresponds to the Zabbix server `SSHKeyLocation` configuration parameter.

Attention:
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 commands, the **public** key file can be installed on a remote host *10.10.10.10*, so that the SSH checks can be performed with a *root* account (for better readability, command prompts are commented out):

```
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.
##### RSA key fingerprint is 38:ba:f2:a4:b5:d9:8f:52:00:09:f7:1f:75:cc:0b:46.
##### 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:
<Enter root password>
##### Now try logging into the machine, with "ssh 'root@10.10.10.10'",
##### and check to make sure that only the key(s) you wanted were added.
```

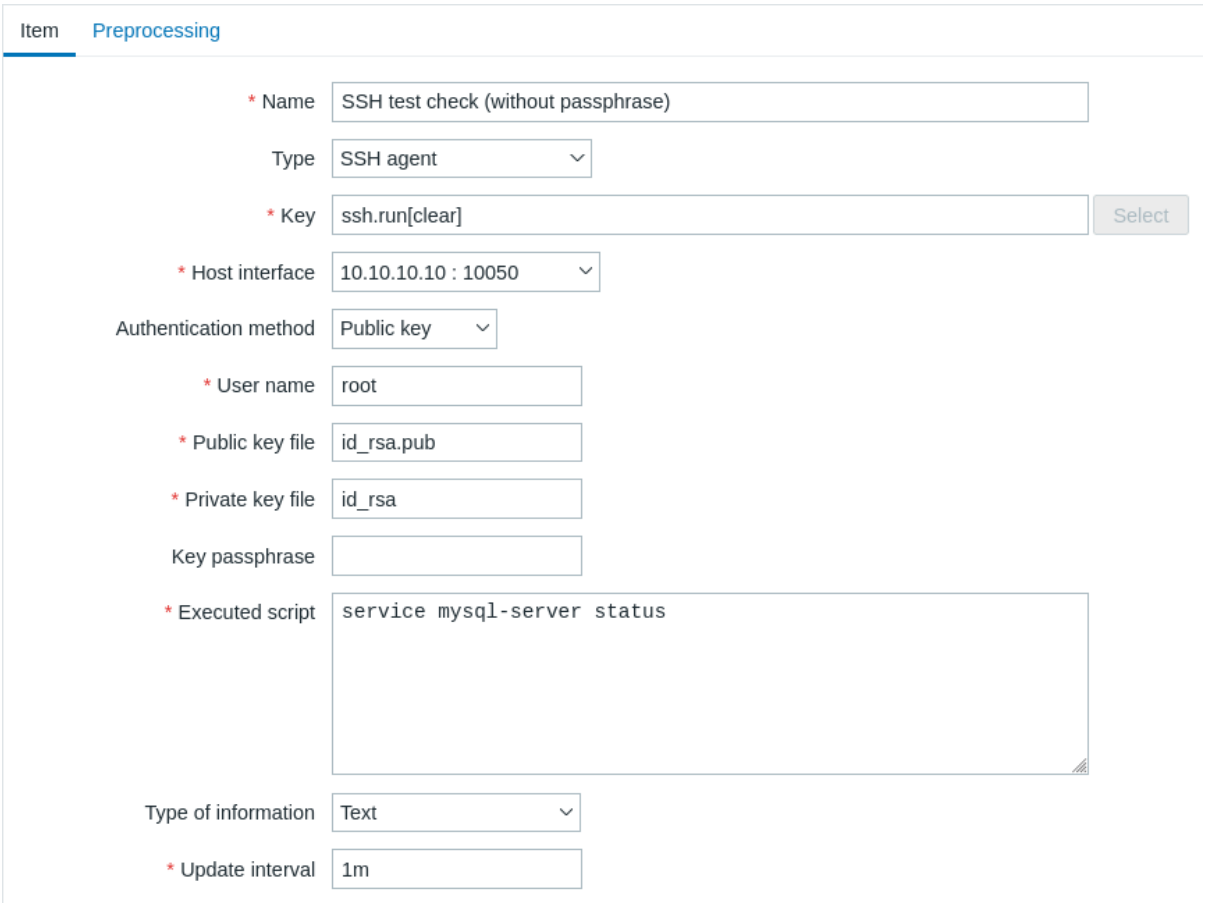
Now it is possible to check the SSH login using the default private key (*/home/zabbix/.ssh/id_rsa*) for the *zabbix* user account:

```
sudo -u zabbix ssh root@10.10.10.10
```

If the login is successful, then the configuration part in the shell is finished and the 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 will also be formatted as multilined.



All mandatory input fields are marked with a red asterisk.

The fields that require specific information for SSH items are:

Parameter	Description	Comments
Type	Select **SSH agent** here.	
Key	Unique (per host) item key in format **ssh.run[<unique short description>,<ip>,<port>,<encoding>]**	<unique short description> is required and should be unique for all SSH items per host. Default port is 22, not the port specified in the interface to which this item is assigned.
Authentication method	One of the "Password" or "Public key".	
User name	User name to authenticate on remote host. Required.	
Public key file	File name of public key if *Authentication method* is "Public key". Required.	Example: *id_rsa.pub* - default public key file name generated by a command ssh-keygen.
Private key file	File name of private key if *Authentication method* is "Public key". Required.	Example: *id_rsa* - default private key file name.
Password or *Key passphrase*	Password to authenticate or Passphrase **if** it was used for the private key	Leave the *Key passphrase* field empty if passphrase was not used. See also known issues regarding passphrase usage.
Executed script	Executed shell command(s) using SSH remote session.	Examples: *date +%s* *service mysql-server status* *ps auxww | grep httpd | wc -l*

Attention:
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 multilined.

Supported characters that the shell prompt can end with:

- $
- #
-
- %

Note:
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.

Key	Description	Comments
telnet.run[<unique short description>,<ip>,<port>,<encoding>]	Run a command on a remote device using telnet connection	

Attention:
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:

ARGUMENT	DEFINITION
script	Name of a shell script or a binary.
parameter(s)	Optional command line parameters.

If you don't want to pass any parameters to the script you may use:

```
script[] or
script
```

Zabbix server or proxy will search the directory specified for external scripts and execute the command (see `ExternalScripts` parameter in Zabbix server/proxy configuration file). The command will be executed under the same user as Zabbix server/proxy, so any access permissions or environment variables should be handled in a wrapper script, if necessary. Permissions on the command should also allow that user to execute it. Only commands in the specified directory are available for execution.

Warning:
Do not overuse external checks, as each script requires starting a fork process by Zabbix server/proxy, and running many scripts can significantly decrease Zabbix performance.

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 server/proxy 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).

Attention:
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/proxy 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 Aggregate checks

Overview

In aggregate checks Zabbix server collects aggregate information from items by doing direct database queries.

Aggregate checks do not require any agent running on the host being monitored.

Syntax

The syntax of the aggregate item key is:

```
groupfunc["host group","item key",itemfunc,timeperiod]
```

Supported group functions (groupfunc) are:

Group function	Description
grpavg	Average value
grpmax	Maximum value
grpmin	Minimum value
grpsum	Sum of values

Multiple host groups may be included by inserting a comma-delimited array. Specifying a parent host group will include the parent group and all nested host groups with their items.

All items that are referenced from the aggregate item key must exist and be collecting data. Only enabled items on enabled hosts are included in the calculations.

Attention:
The key of the aggregate item must be updated manually, if the item key of a referenced item is changed.

Supported item functions (itemfunc) are:

Item function	Description
avg	Average value
count	Number of values
last	Last value
max	Maximum value
min	Minimum value
sum	Sum of values

The **timeperiod** parameter specifies a time period of latest collected values. Supported unit symbols can be used in this parameter for convenience, for example '5m' (minutes) instead of '300' (seconds) or '1d' (day) instead of '86400' (seconds).

Warning:
An amount of values (prefixed with **#**) is not supported in the timeperiod.

Timeperiod is ignored by the server if the third parameter (item function) is *last* and can thus be omitted:

```
groupfunc["host group","item key",last]
```

Note:
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)*.

User macros and low-level discovery macros are supported in:

- item key parameters
- function parameters
- expression constants

An aggregate item may become unsupported if:

- none of the referenced items is found (which may happen if the item key is incorrect, none of the items exist or all included groups are incorrect)

- no data to calculate a function

Usage examples

Examples of keys for aggregate checks:

Example 1

Total disk space of host group 'MySQL Servers'.

```
grpsum["MySQL Servers","vfs.fs.size[/,total]",last]
```

Example 2

Average processor load of host group 'MySQL Servers'.

```
grpavg["MySQL Servers","system.cpu.load[,avg1]",last]
```

Example 3

5-minute average of the number of queries per second for host group 'MySQL Servers'.

```
grpavg["MySQL Servers",mysql.qps,avg,5m]
```

Example 4

Average CPU load on all hosts in multiple host groups.

```
grpavg[["Servers A","Servers B","Servers C"],system.cpu.load,last]
```

13 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:

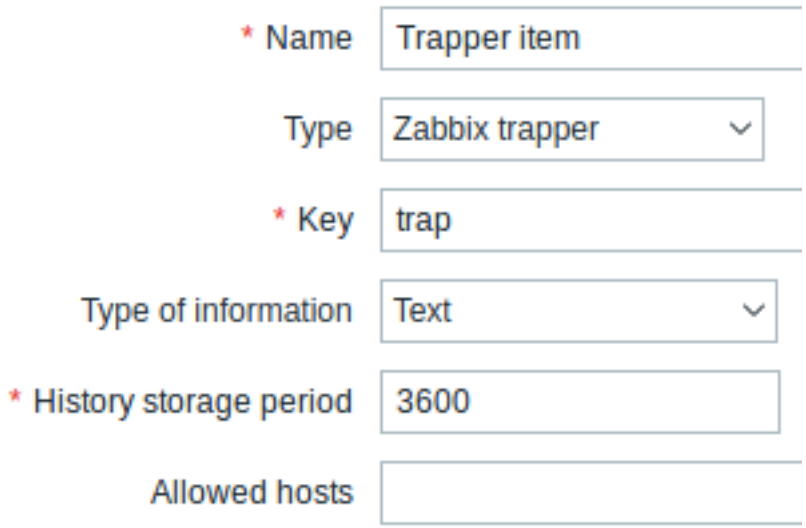
- have a trapper item set up in Zabbix
- send in the data into Zabbix

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



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 DNS names. 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: 127.0.0.1, 192.168.1.0/24, 192.168.3.1-255, 192.168.1-10.1-255, ::1,2001:db8::/32, mysqlserver1, zabbix.example.com, {HOST.HOST} 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.

Note:
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 in 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)

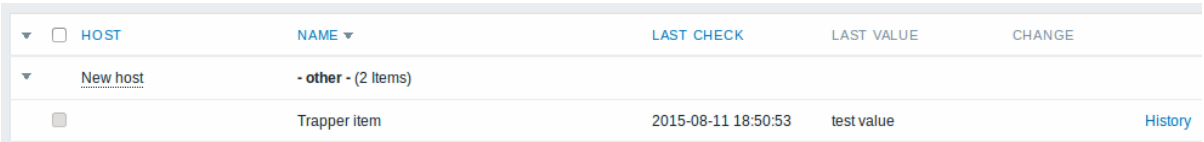
-k - to specify the key of the item we just defined

-o - to specify the actual value to send

Attention:
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*:



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.

14 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.

> **Warning:**
> 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 \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.authenticate=false \
-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 \
-Djava.rmi.server.hostname=192.168.3.14 \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.authenticate=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.ssl=true \
-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.

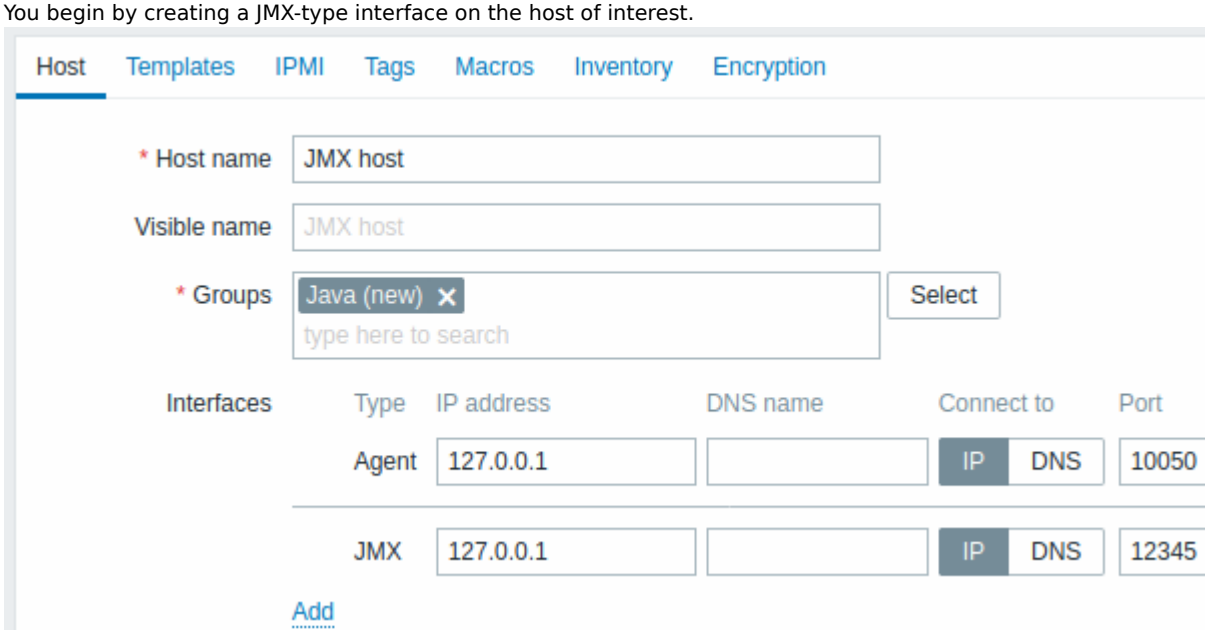
See Monitoring and Management Using JMX for a detailed description.

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.

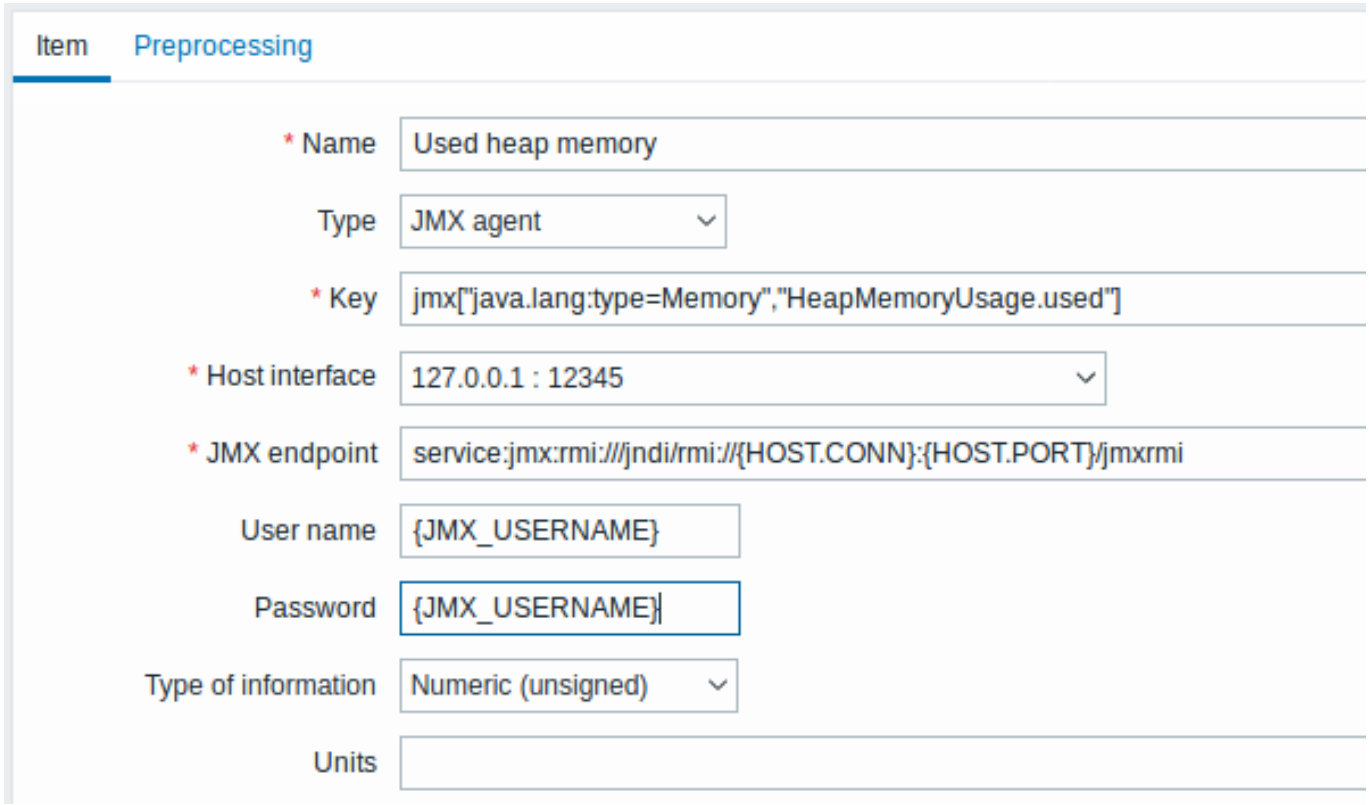


All mandatory input fields are marked with a red asterisk.

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"]`.



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 two parameters: **object name** - the object name of an MBean; **attribute name** - an MBean attribute name with optional composite data field names separated by dots. 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 the object name is ”com.example:Type=Hello”, the attribute name is ”weight”, and the returned value type should probably 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
```

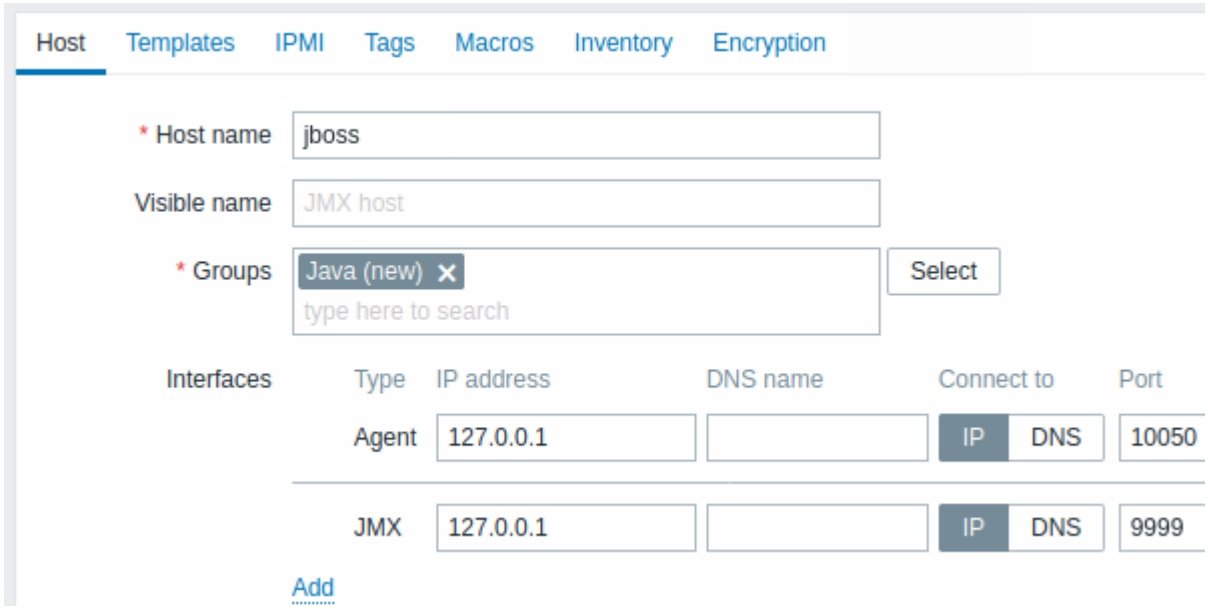
And in the `zabbix_java/settings.sh` configuration file (or `zabbix_java_gateway.conf`):

```
START_POLLERS=5
```

Check that JBoss listens to its standard management port:

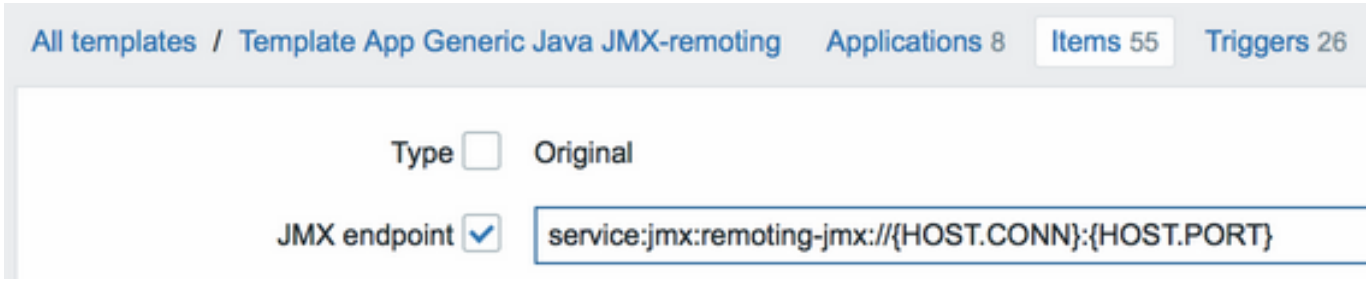
```
$ netstat -natp | grep 9999
tcp        0      0 127.0.0.1:9999          0.0.0.0:*               LISTEN      10148/java
```

Now let's create a host with JMX interface 127.0.0.1:9999 in Zabbix.



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:

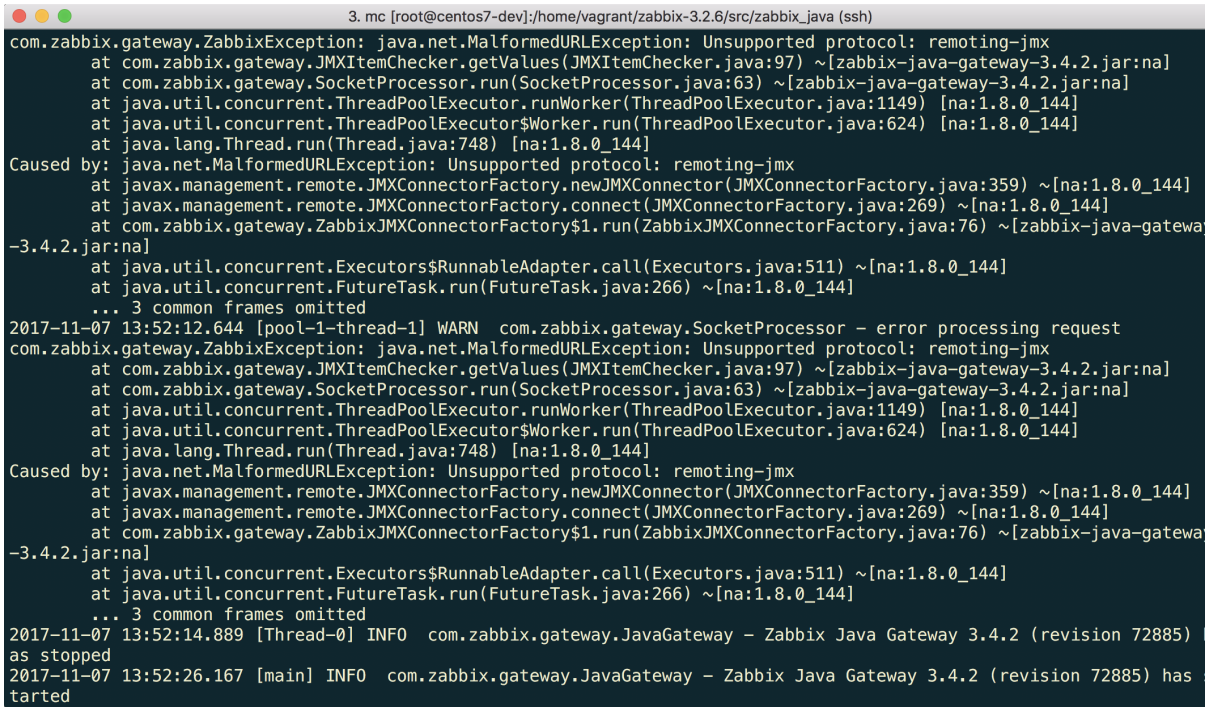
`service:jmx:remoting-jmx://{HOST.CONN}:{HOST.PORT}`



Let's update the configuration cache:

```
$ /usr/local/sbin/zabbix_server -R config_cache_reload
```

Note that you may encounter an error first.



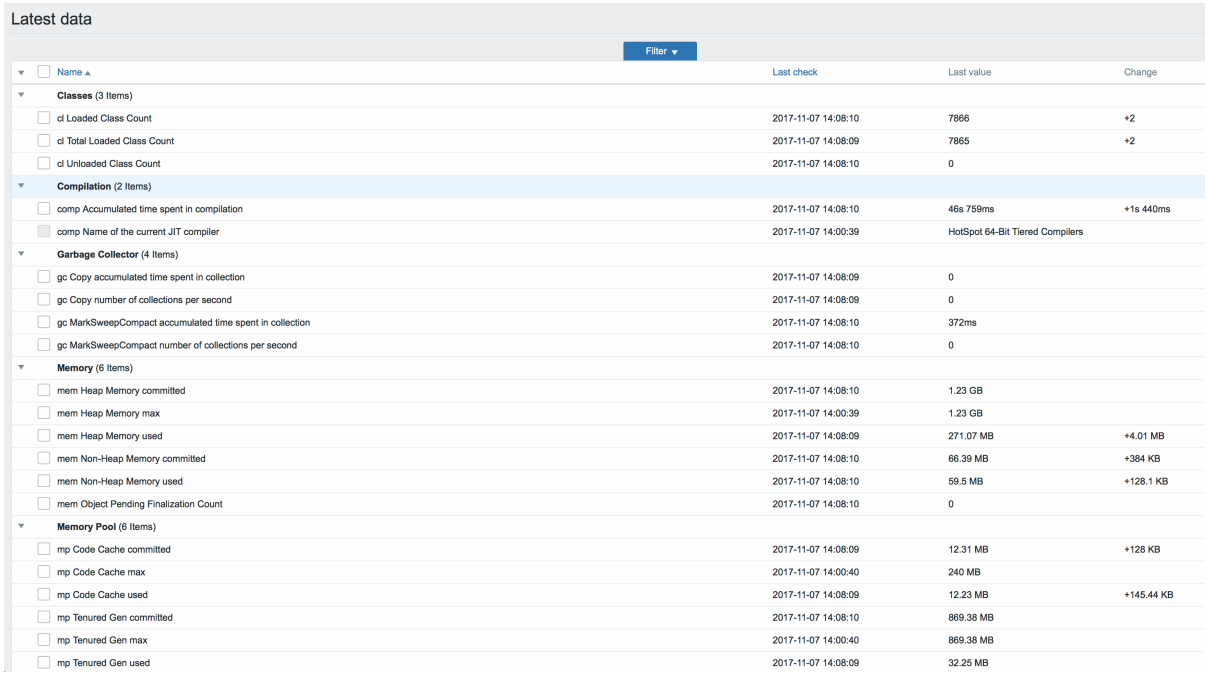
"Unsupported protocol: remoting-jmx" means that Java gateway does not know how to work with the specified protocol. That can be fixed by creating a `~/needed_modules.txt` file with the following content:

```
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/l
```

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:



15 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.

Attention:
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.

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
```

Note:
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. 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:

Attribute	Description
mysql	Database driver name.
Description	Database driver description.
Driver	Database driver library location.

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:

Attribute	Description
test	Data source name (DSN).
Description	Data source description.
Driver	Database driver name - as specified in odbcinst.ini
Server	Database server IP/DNS.
User	Database user for connection.
Password	Database user password.
Port	Database connection port.
Database	Database name.

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

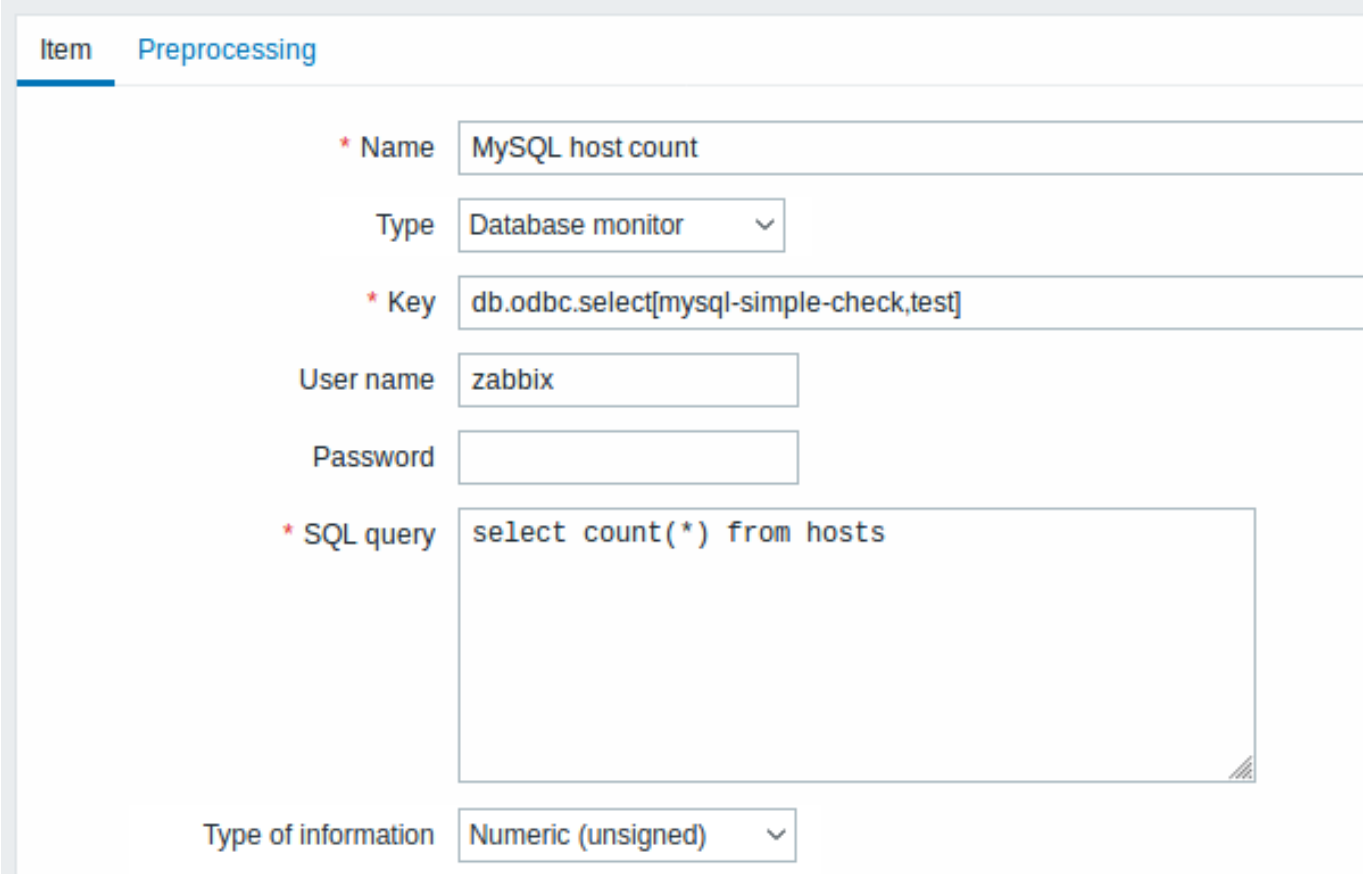
To enable ODBC support, Zabbix should be compiled with the following flag:

```
--with-unixodbc[=ARG]   use odbc driver against unixODBC package
```

Note:
See more about Zabbix installation from the source code.

Item configuration in Zabbix frontend

Configure a database monitoring item.



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.
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

- 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, an error message might look like this:

```
Cannot execute ODBC query: [SQL_ERROR]:[42601][7][ERROR: syntax error at or near ";"; Error while executin
```

- "`Cannot execute ODBC query`" - Zabbix message
- "`[SQL_ERROR]`" - ODBC return code
- "`[42601]`" - SQLState

- ”[7]” - Native error code
- ”[ERROR: syntax error at or near ";"; Error while executing the query]” - Native error message

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

Installation

```
*** 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:

```
[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:

```
[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:

```
[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.

2 Recommended UnixODBC settings for PostgreSQL

Installation

- ** Red Hat Enterprise Linux/CentOS**:

```
# yum install postgresql-odbc
```

- **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

```
[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 e
# This entry alters the default thread serialization level.
Threading   = 2
```

odbc.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
# Whether the datasource will allow updates.
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.
# It also speeds the update process since every single column does not need to be specified in the where c
RowVersioning = No
# Show SystemTables
# The driver will treat system tables as regular tables in SQLTables. This is good for Access so you can s
ShowSystemTables = No
# If true, the driver automatically uses declare cursor/fetch to handle SELECT statements and keeps 100 ro
Fetch = Yes
# Bools as Char
# Bools are mapped to SQL_CHAR, otherwise to SQL_BIT.
BoolsAsChar = Yes
# SSL mode
```

```
SSLmode = Require
# 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
```

16 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.

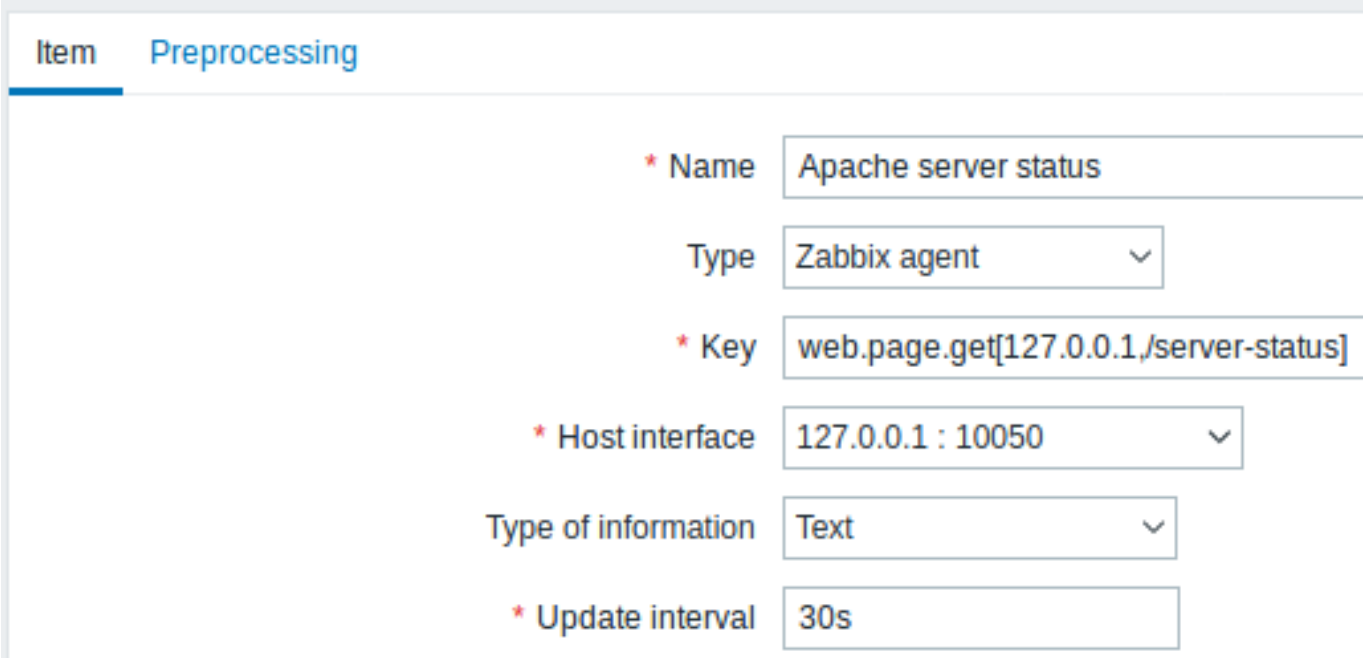
Limitations

- 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:

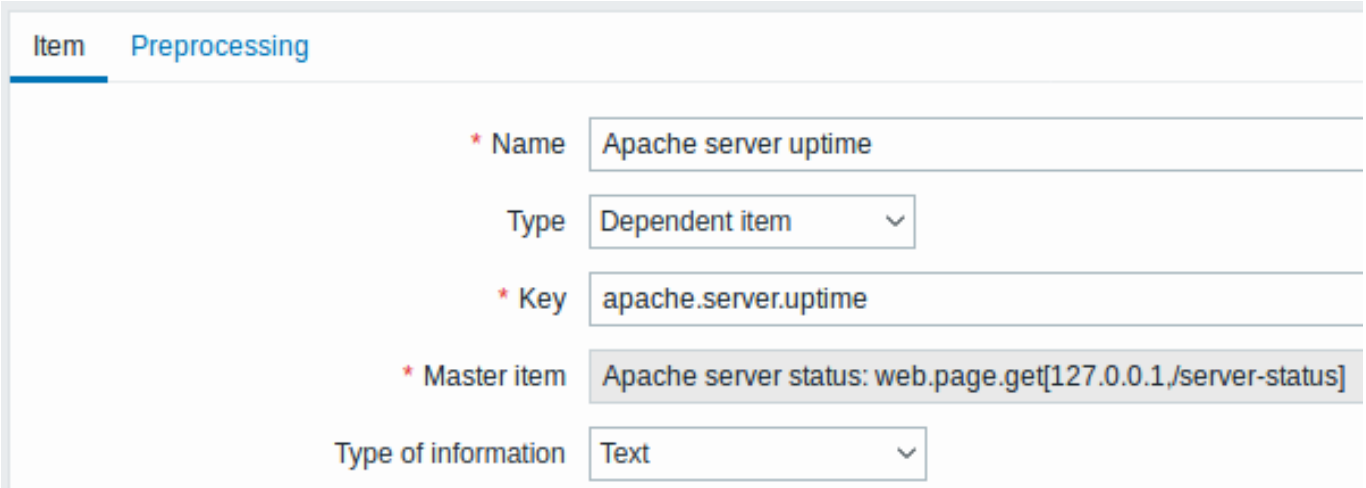
- Go to: *Configuration* → *Hosts*
- Click on *Items* in the row of the host
- Click on *Create item*
- Enter parameters of the item in the form



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**.

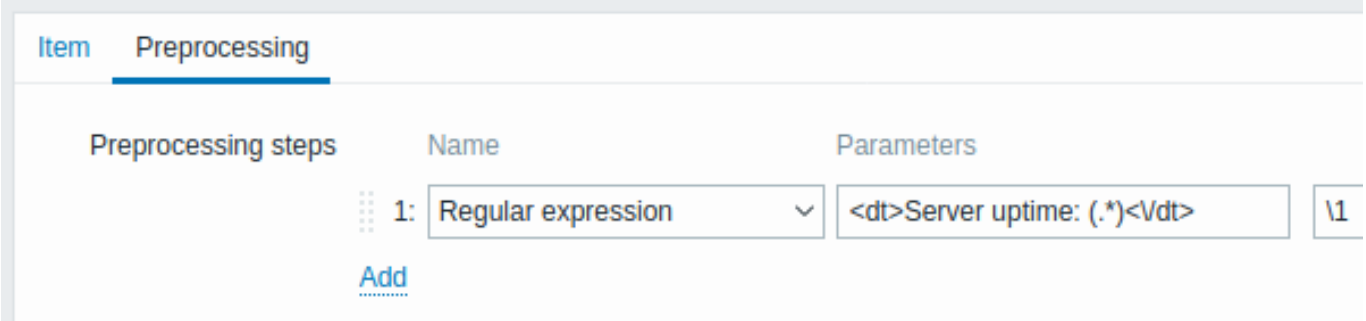


All mandatory input fields are marked with a red asterisk.

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.

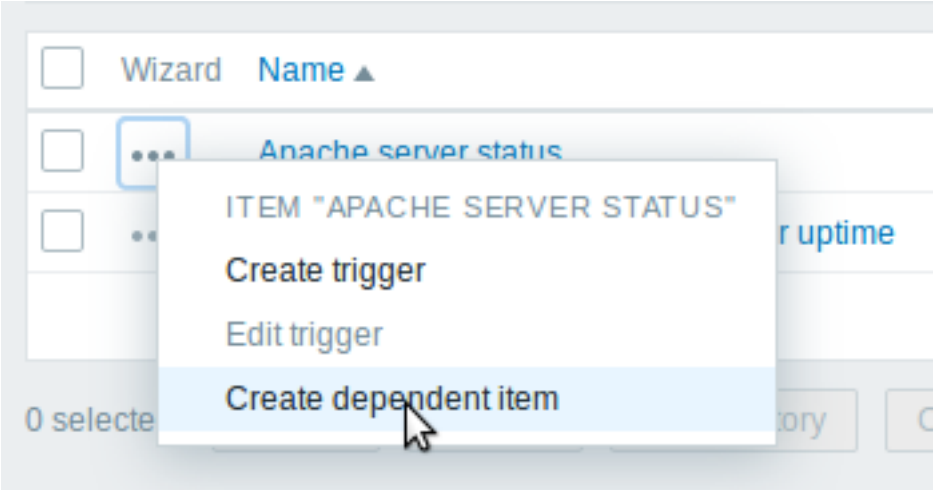
You may use item value preprocessing to extract the required part of the master item value.



Without preprocessing, the dependent item value will be exactly the same as the master item value.

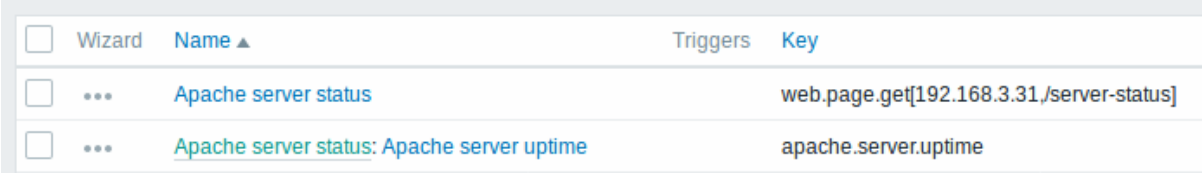
Click on *Add* to save the dependent item.

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.

17 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.

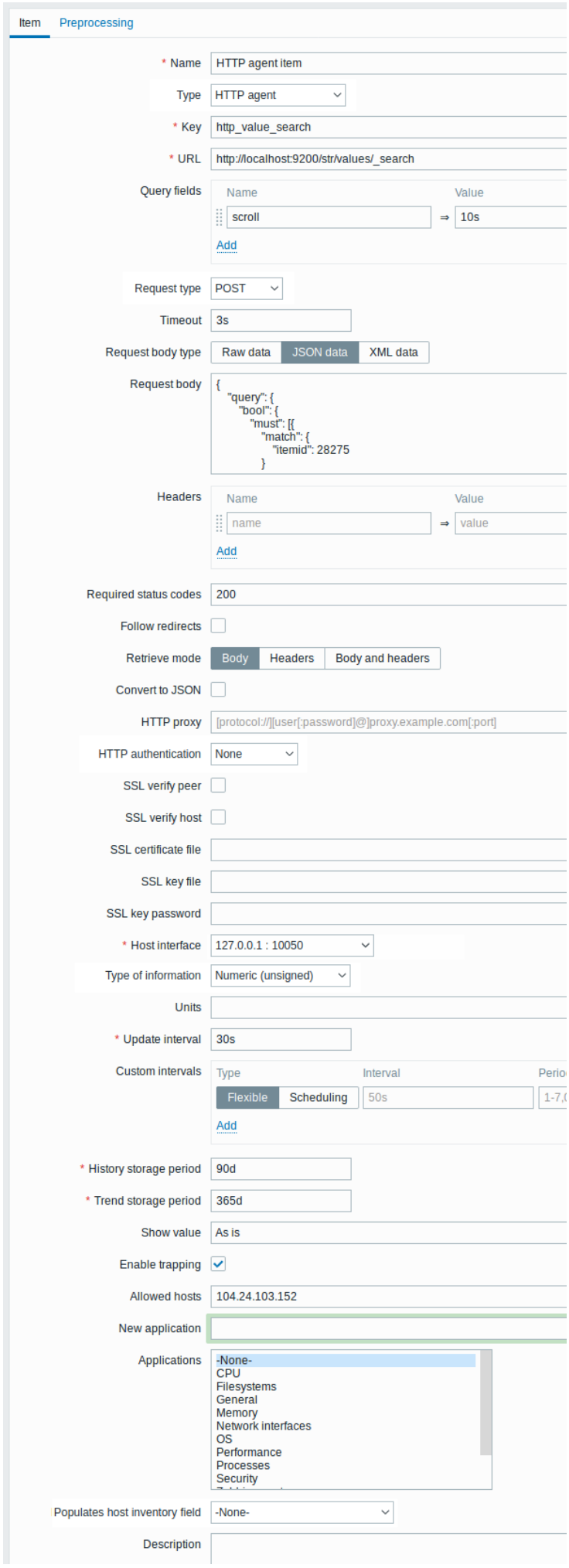
HTTP agent supports both HTTP and HTTPS. Zabbix will optionally follow redirects (see the *Follow redirects* option below). Maximum number of redirects is hard-coded to 10 (using cURL option CURLOPT_MAXREDIRS).

Attention:
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


Item
Preprocessing
* Name
HTTP agent item
Type
HTTP agent
* Key
http_value_search
* URL
http://localhost:9200/str/values/_search
Query fields
Name
Value
scroll
10s
Add
Request type
POST
Timeout
3s
Request body type
Raw data
JSON data
XML data
Request body
{
"query": {
"bool": {
"must": [{
"match": {
"itemid": 28275
}
Headers
Name
Value
name
value
Add
Required status codes
200
Follow redirects
Retrieve mode
Body
Headers
Body and headers
Convert to JSON
HTTP proxy
[protocol://][user[:password]@]proxy.example.com[:port]
HTTP authentication
None
SSL verify peer
SSL verify host
SSL certificate file
SSL key file
SSL key password
* Host interface
127.0.0.1 : 10050
Type of information
Numeric (unsigned)
Units
* Update interval
30s
Custom intervals
Type
Interval
Flexible
Scheduling
50s
Add
* History storage period
90d
* Trend storage period
365d
Show value
As is
Enable trapping
Allowed hosts
104.24.103.152
New application
Applications
-None-
CPU
Filesystems
General
Memory
Network interfaces
OS
Performance
Processes
Security
Populates host inventory field
-None-
Description

All mandatory input fields are marked with a red asterisk.

The fields that require specific information for HTTP items are:

Type	Select **HTTP agent** here.
Key	Enter a unique item key.
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}, 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}, user macros, low-level discovery macros. This sets the CURLOPT_URL cURL option.
Request type	Select request method type: *GET*, *POST*, *PUT* or *HEAD*
Timeout	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.
Request body type	Select the request body type: **Raw data** - custom HTTP request body, macros are substituted but no encoding is performed **JSON data** - 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" **XML data** - 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.
Request body	Enter the request body. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, user macros, low-level discovery macros.

Headers	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}, user macros, low-level discovery macros. This sets the CURLOPT_HTTPHEADER cURL option.
Required status codes	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 CURLINFO_RESPONSE_CODE cURL option.
Follow redirects	Mark the checkbox to follow HTTP redirects. This sets the CURLOPT_FOLLOWLOCATION cURL option.
Retrieve mode	Select the part of response that must be retrieved: **Body** - body only **Headers** - headers only **Body and headers** - body and headers
Convert to JSON	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: `{ "header": { "<key>": "<value>", "<key2>": "<value>" }, "body": <body> }`
HTTP proxy	You can specify an HTTP proxy to use, using the format `[protocol://][username[:password]@]proxy.example.com[:po` 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}, 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. 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_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[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}, 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}, 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 DNS names. 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: 127.0.0.1, 192.168.1.0/24, 192.168.3.1-255, 192.168.1-10.1-255, ::1,2001:db8::/32, mysqlserver1, zabbix.example.com, {HOST.HOST} 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.

Note:
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: `%%http_proxy=http://proxy_ip:proxy_port`.
For HTTPS - 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*.

Attention:
[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" : "kH4CYqh5QfqgeTsjh2F9zg",
  "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:

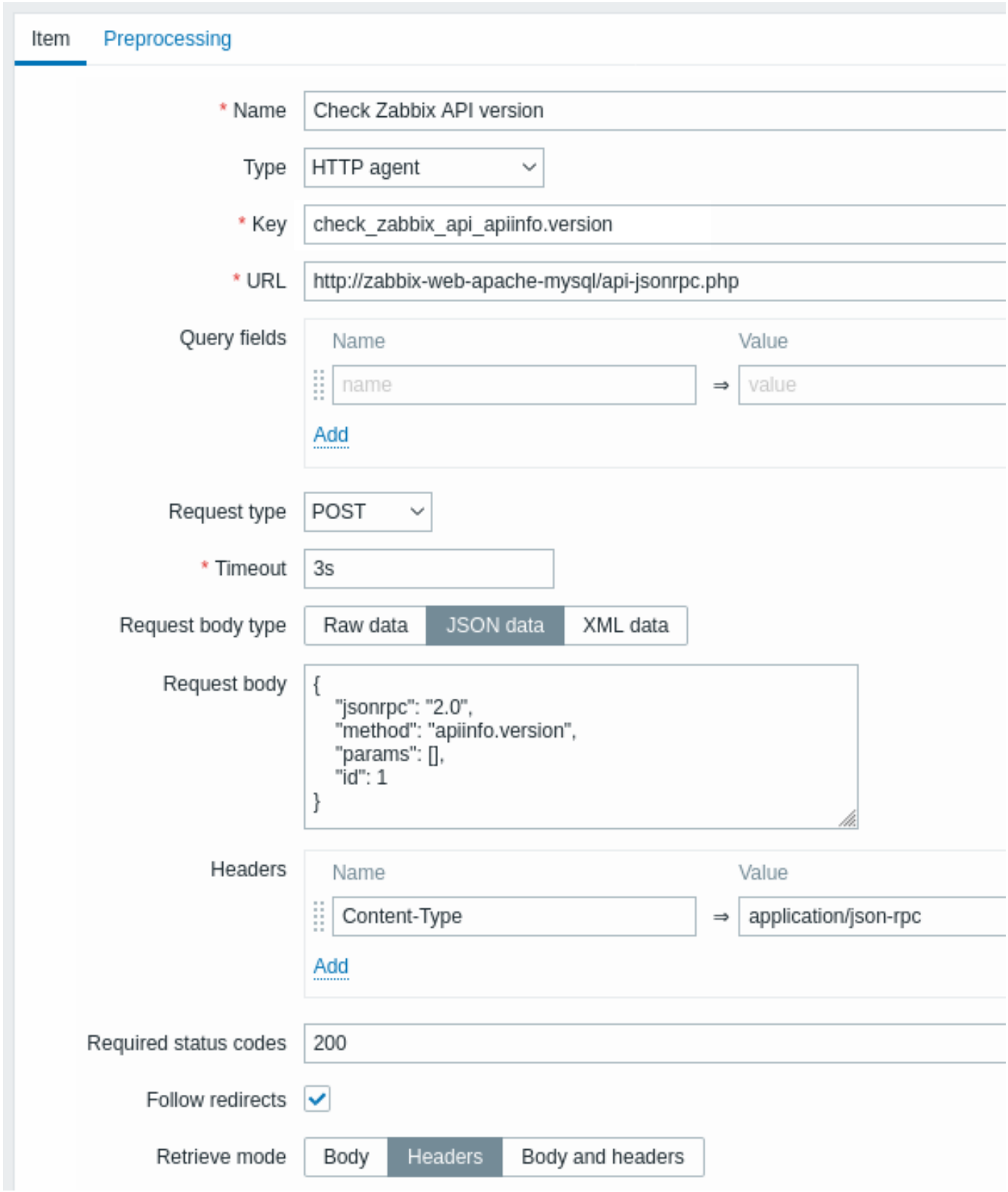
```
{
    "_scroll_id": "DnF1ZXJ5VGhlbkZldGNoBQAAAAAAAAAkFllRMlZBWS1UU1pxTmdEeGVwQjRBTFEAAAAAAAAAJRZZUTJWQVktVFN
    "took": 18,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": 1,
        "max_score": 1.0,
        "hits": [{
            "_index": "dbl",
            "_type": "values",
            "_id": "dqX9VWEBV6sEKSMyk6sw",
            "_score": 1.0,
            "_source": {
                "itemid": 28275,
                "value": "0.138750",
                "clock": 1517566136,
                "ns": 25388713,
                "ttl": 604800
            }
        }]
    }
}
```

- Now use a JSONPath preprocessing step to get the item value: `$.hits.hits[0]._source.value`

Example 3

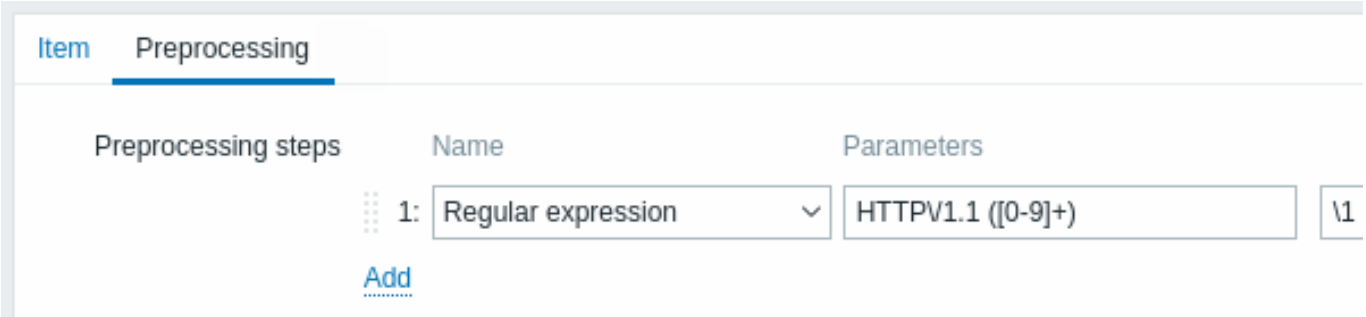
Checking if Zabbix API is alive, using apiinfo.version.

- Item configuration:

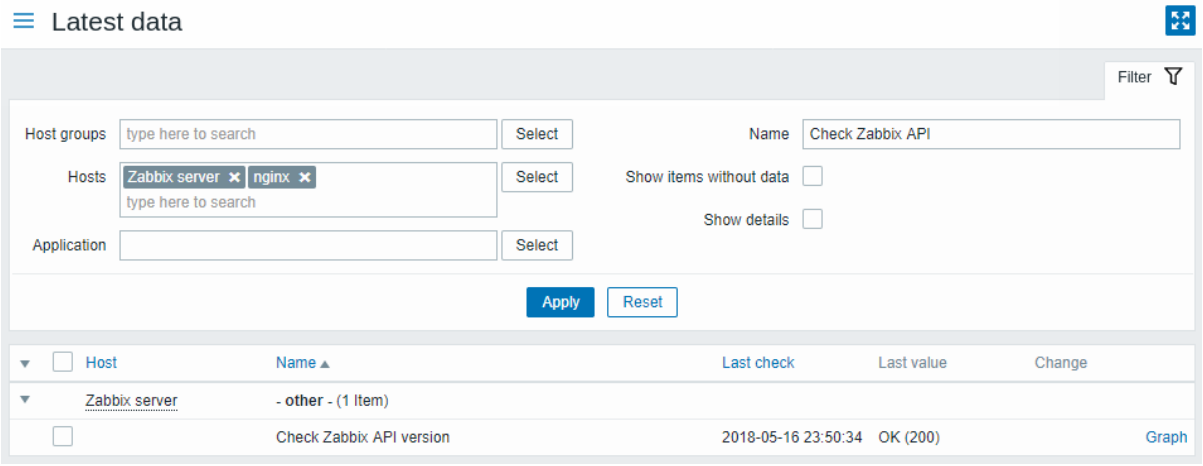


Note the use of the POST method with JSON data, setting request headers and asking to return headers only:

- Item value preprocessing with regular expression to get HTTP code:



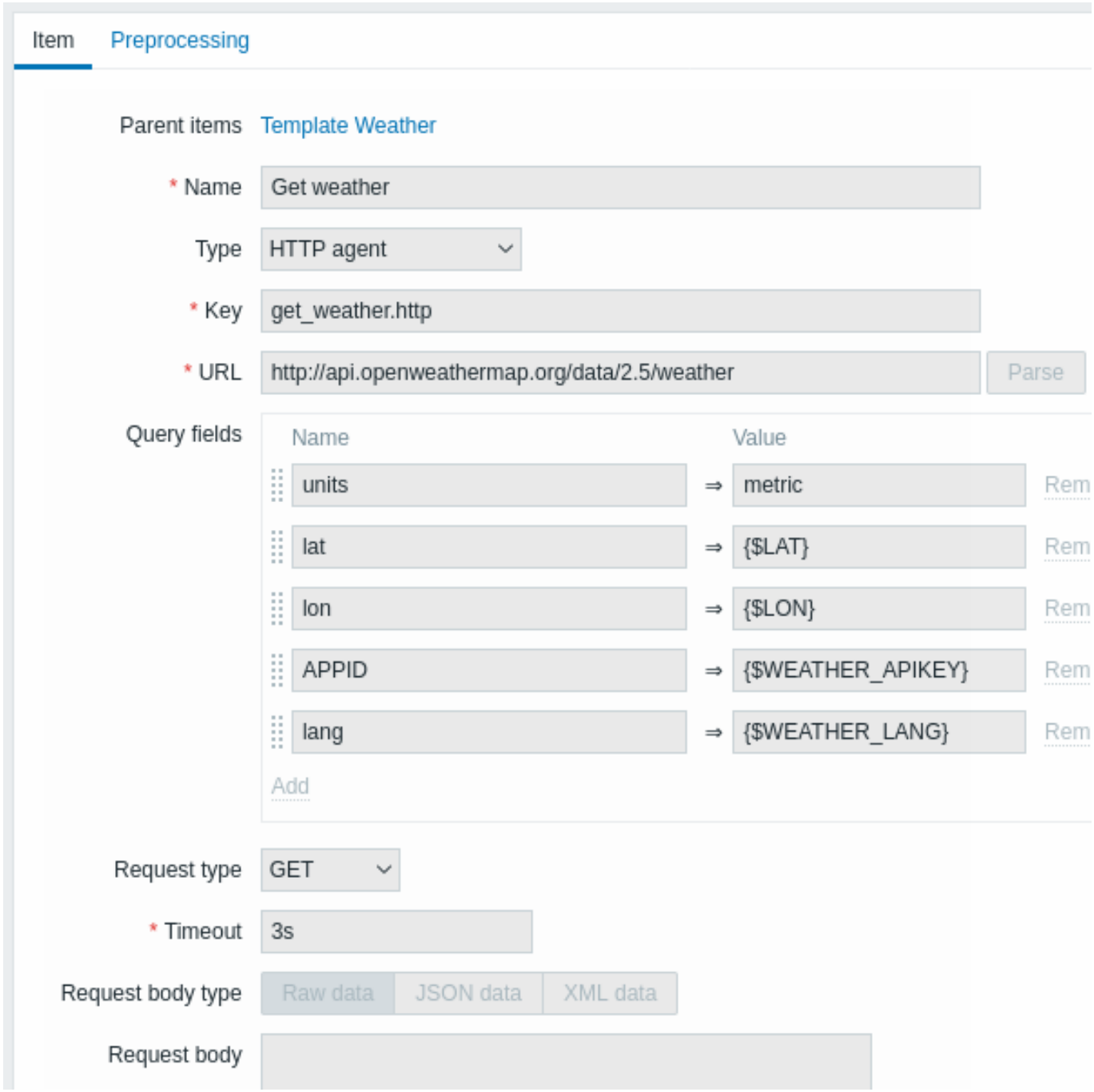
- Checking the result in *Latest data*:



Example 4

Retrieving weather information by connecting to the Openweathermap public service.

- Configure a master item for bulk data collection in a single JSON:



Note the usage of macros in query fields. Refer to the Openweathermap API for how to fill them.

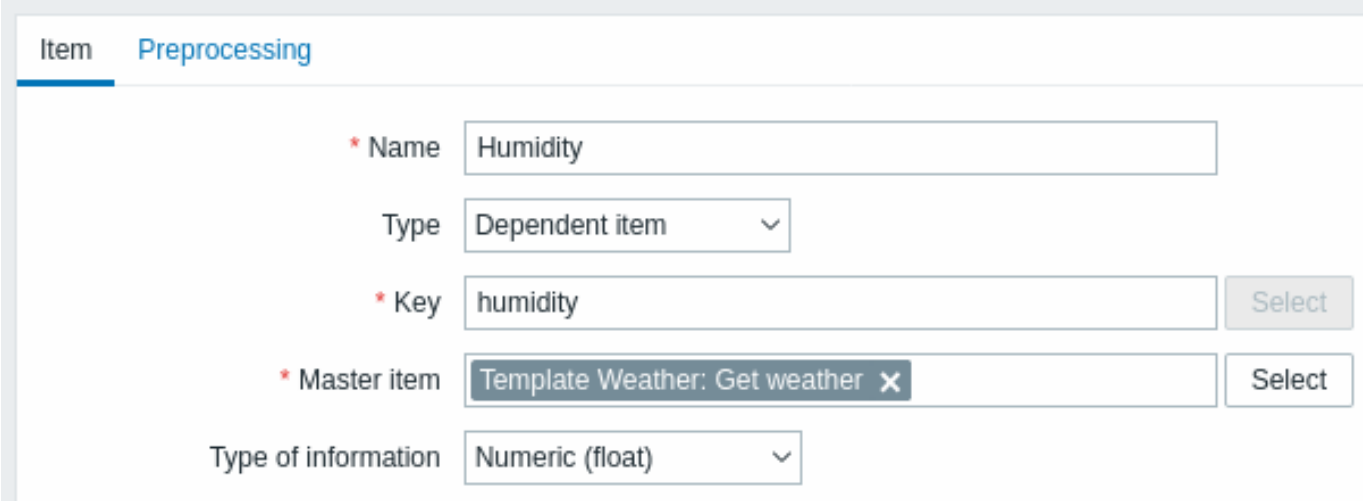
Sample JSON returned in response to HTTP agent:

```
{
    "body": {
        "coord": {
            "lon": 40.01,
            "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
    }
}
```

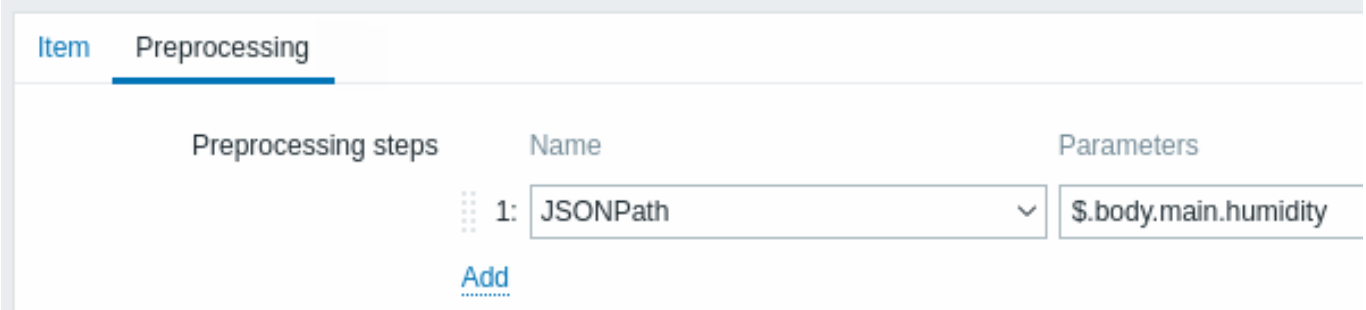

The next task is to configure dependent items that extract data from the JSON.

- Configure a sample dependent item for humidity:



Other weather metrics such as 'Temperature' are added in the same manner.

- Sample dependent item value preprocessing with JSONPath:



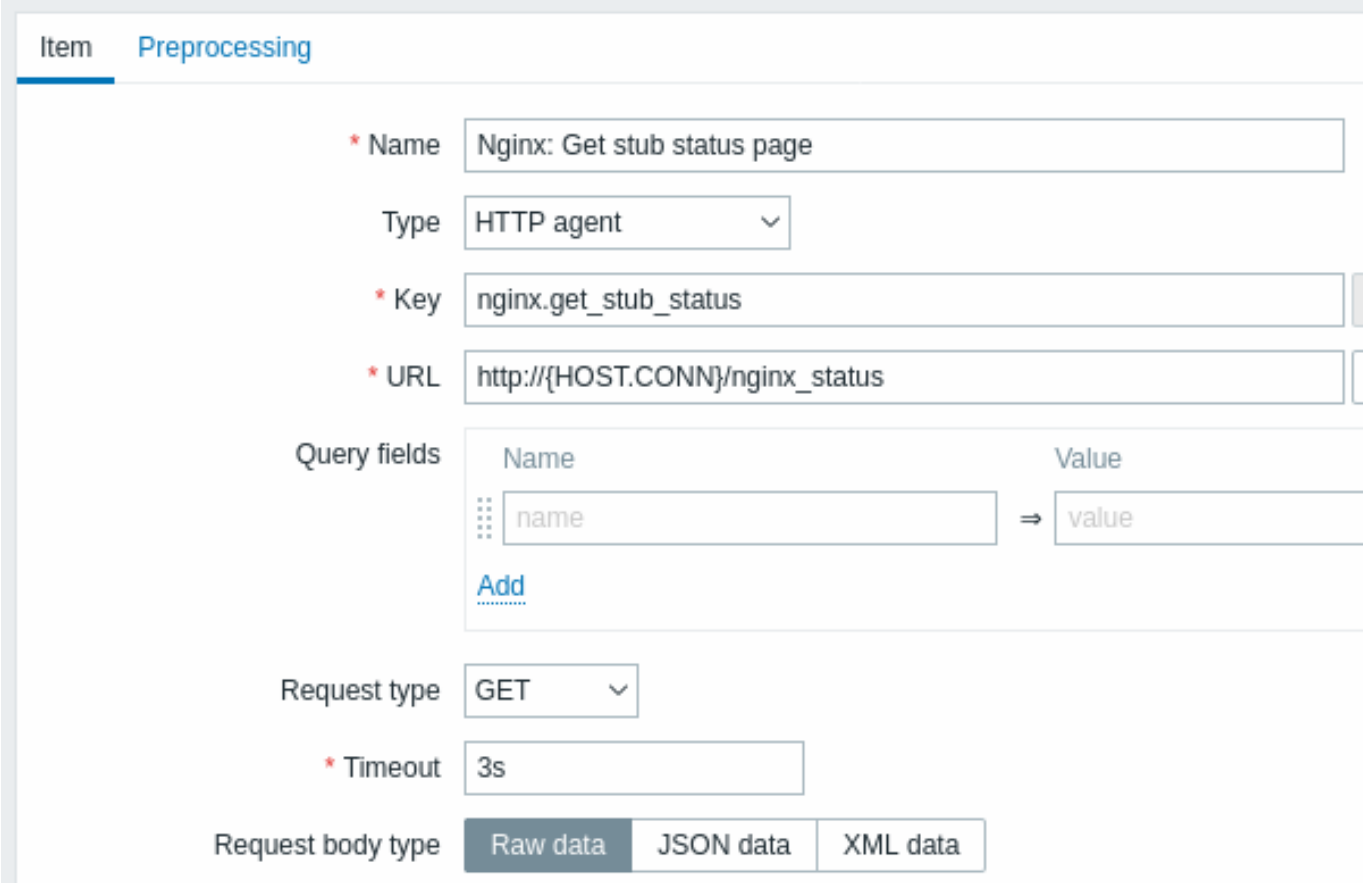
- Check the result of weather data in *Latest data*:

Host	Name ▲	Inter...	History	Trends	Type	Last check	Last value
weather	**Weather** (8 Items)						
	Get weather get_weather.http	10m	1d		HTTP agent	2018-05-17 01:23:45	{"body":{"coord":{"lon...
	Get weather HTTP response code get_weather.http_code		7d	0	Depende...	2018-05-17 01:23:45	OK (200)
	Humidity humidity		90d	365d	Depende...	2018-05-17 01:23:45	66 %
	Temperature temp		90d	365d	Depende...	2018-05-17 01:23:45	15.14 C
	Weather weather		90d		Depende...	2018-05-17 01:23:45	Clouds
	Weather condition id weather.condition.id		7d	0	Depende...	2018-05-17 01:23:45	801
	Weather description weather.description		90d		Depende...	2018-05-17 01:23:45	few clouds
	Wind speed wind.speed		90d	365d	Depende...	2018-05-17 01:23:45	1.86 m/s

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:

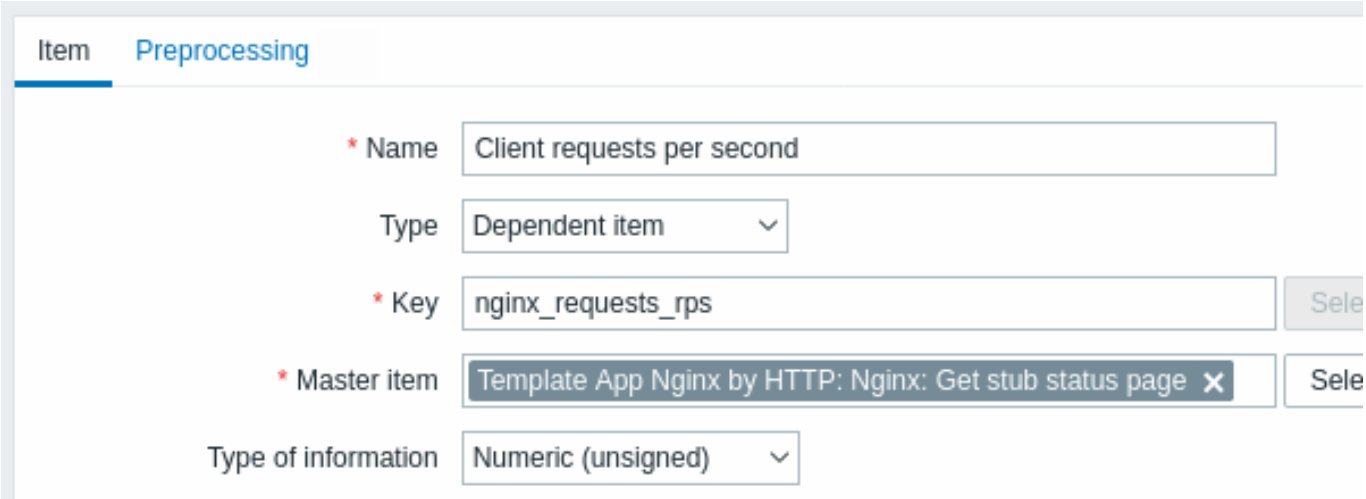


Sample Nginx stub status output:

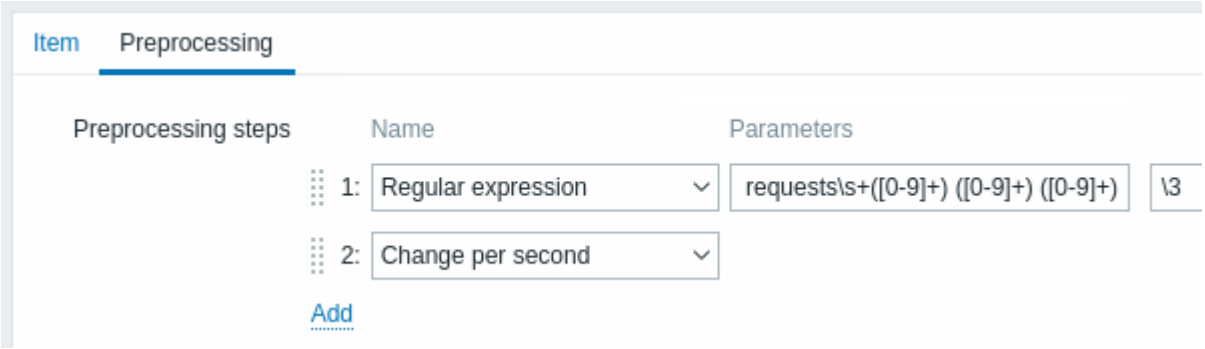
```
Active connections: 1 Active connections:
server accepts handled requests
 52 52 52
Reading: 0 Writing: 1 Waiting: 0
```

The next task is to configure dependent items that extract data.

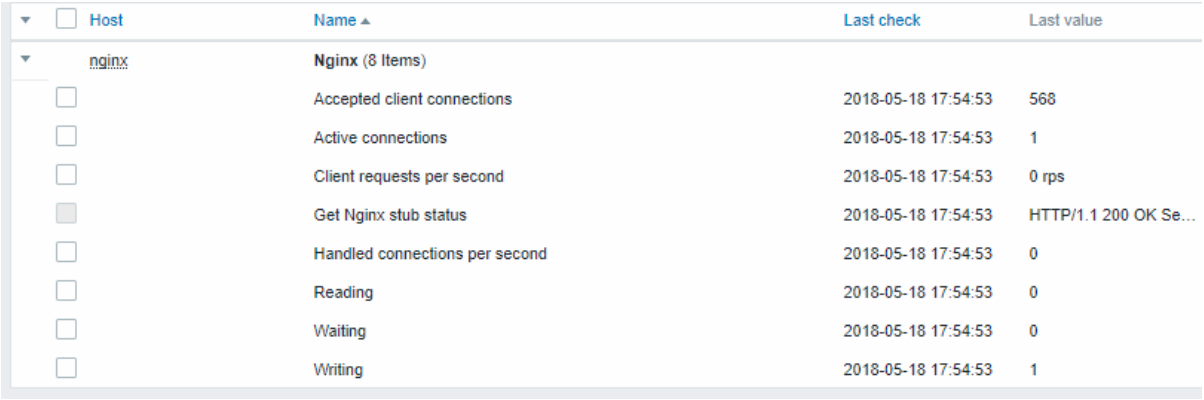
- Configure a sample dependent item for requests per second:



- Sample dependent item value preprocessing with regular expression:



- Check the complete result from stub module in *Latest data*:



18 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

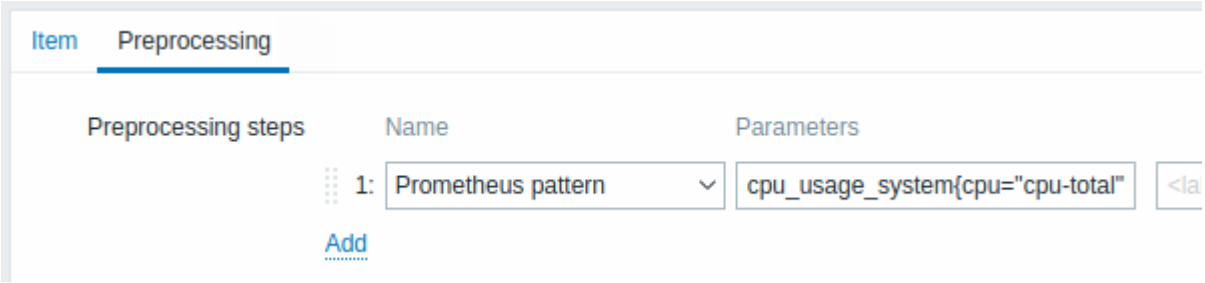
There are two Prometheus data preprocessing options:

- *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.

Configuration

Providing you have the HTTP master item configured, you need to create a dependent item that uses 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*)



Parameter	Description	Examples
Pattern	To define the required data pattern you may use a query language that is similar to Prometheus query language (see comparison table), e.g.: <metric name> - select by metric name {__name__="<metric name>"} - select by metric name {__name__=~"<regex>"} - select by metric name matching a regular expression {<label name>="<label value>",...} - select by label name {<label name>=~"<regex>",...} - select by label name matching a regular expression {__name__=~".*"}==<value> - select by metric value Or a combination of the above: <metric name>{<label1 name>="<label1 value>",<label2 name>=~"<regex>",...}==<value> 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.	*wmi_os_physical_memory_free_bytes* *cpu_usage_system{cpu="cpu-total"}* *cpu_usage_system{cpu=~".*"}* *cpu_usage_system{cpu="cpu-total",host=~".*"}* *wmi_service_state{name="dhcp"}==1* *wmi_os_timezone{timezone=~".*"}==1*
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.	

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.

PromQL instant vector selector	Zabbix Prometheus preprocessing
Differences	

PromQL instant vector selector	Zabbix Prometheus preprocessing
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>"}	<metric name> or {__name__="<metric name>"}
Selecting by metric name that matches regular expression	{__name__=~"<regex>"}	{__name__=~"<regex>"}

PromQL instant vector selector	Zabbix Prometheus preprocessing
Select by <label name> value that equals string	{<label name>="<label value>",...}	{<label name>="<label value>",...}
Select by <label name> value that matches regular expression	{<label name>=~"<regex>",...}	{<label name>=~"<regex>",...}
Select by value that equals string	{__name__=~".*"} == <value>	{__name__=~".*"} == <value>

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.

Attention:
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.

Note:
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.

Attention:
If trends are set to '0', Zabbix server does not calculate or store trends at all.

Note:
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.

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
```

Parameter	Description
Key	Unique item key. The [*] defines that this key accepts parameters within the brackets. Parameters are given when configuring the item.
Command	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.

Attention:
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.

Attention:
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]
wc[/etc/services,zabbix]
```

Command result

The return value of the command is standard output together with standard error.

> **Attention:**
> 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 -uroot 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

Restart Zabbix agent.

Agent will reload configuration file.

Test this 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()**:

```
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()**:

```
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.

```
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.

```
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.

```
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.

```
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:

```
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:

```
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:

```
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

Attention:
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;

typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    const char  *value;
    const char  *source;
    int     timestamp;
    int     logeventid;
    int     severity;
}
ZBX_HISTORY_LOG;
```

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.

Attention:
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. The agent will fail to start 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**.

```c
/*
** 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.
**/

####include <stdlib.h>
####include <string.h>
####include <time.h>
####include <stdint.h>

####include "module.h"

/* the variable keeps timeout setting for item processing */
static int  item_timeout = 0;

/* module SHOULD define internal functions as static and use a naming pattern different from Zabbix intern
```

```c
/* symbols (zbx_*) and loadable module API functions (zbx_module_*) to avoid conflicts
static int  dummy_ping(AGENT_REQUEST *request, AGENT_RESULT *result);
static int  dummy_echo(AGENT_REQUEST *request, AGENT_RESULT *result);
static int  dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result);

static ZBX_METRIC keys[] =
/*  KEY         FLAG        FUNCTION    TEST PARAMETERS */
{
    {"dummy.ping",      0,      dummy_ping, NULL},
    {"dummy.echo",      CF_HAVEPARAMS,  dummy_echo, "a message"},
    {"dummy.random",    CF_HAVEPARAMS,  dummy_random,   "1,1000"},
    {NULL}
};

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_api_version                                           *
 *                                                                            *
 * Purpose: returns version number of the module interface                    *
 *                                                                            *
 * Return value: ZBX_MODULE_API_VERSION - version of module.h module is       *
 *               compiled with, in order to load module successfully Zabbix   *
 *               MUST be compiled with the same version of this header file   *
 *                                                                            *
 ******************************************************************************/
int zbx_module_api_version(void)
{
    return ZBX_MODULE_API_VERSION;
}

/******************************************************************************
 *                                                                            *
 * 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:              *
 *                  REQUEST_PARAMETER_TYPE_UNDEFINED (key parameter is empty) *
 *                  REQUEST_PARAMETER_TYPE_ARRAY (array)                      *
 *                  REQUEST_PARAMETER_TYPE_STRING (quoted or unquoted string) *
 *                                                                            *
 *             result - structure that will contain result                    *
 *                                                                            *
 * Return value: SYSINFO_RET_FAIL - function failed, item will be marked      *
 *                                  as not supported by zabbix                *
 *               SYSINFO_RET_OK - success                                     *
 *                                                                            *
 * Comment: get_rparam(request, N-1) can be used to get a pointer to the Nth  *
 *          parameter starting from 0 (first parameter). Make sure it exists  *
 *          by checking value of request->nparam.                             *
 *          In the same manner get_rparam_type(request, N-1) can be used to   *
 *          get a parameter type.                                             *
 *                                                                            *
 ******************************************************************************/
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;
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_init                                                  *
 *                                                                            *
 * Purpose: the function is called on agent startup                           *
 *          It should be used to call any initialization routines             *
 *                                                                            *
 * Return value: ZBX_MODULE_OK - success                                      *
 *               ZBX_MODULE_FAIL - module initialization failed               *
 *                                                                            *
 * Comment: the module won't be loaded in case of ZBX_MODULE_FAIL             *
 *                                                                            *
 ******************************************************************************/
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, ... */
    }
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_history_write_cbs                                     *
 *                                                                            *
 * Purpose: returns a set of module functions Zabbix will call to export      *
```

```
 *          different types of historical data                                *
 *                                                                            *
 * Return value: structure with callback function pointers (can be NULL if    *
 *               module is not interested in data of certain types)           *
 *                                                                            *
 ******************************************************************************/
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:

```
perf_counter["\Processor(0)\Interrupts/sec"]
```

or

```
perf_counter["\Processor(0)\Interrupts/sec", 10]
```

For more information on using this key or its English-only equivalent perf_counter_en, see Windows-specific item keys.

In order to get a full list of performance counters available for monitoring, you may run:

```
typeperf -qx
```

Starting with Zabbix 5.0.1, you may also use low-level discovery to discover multiple 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 perf_counter, perf_counter_en item keys and in PerfCounter, 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 perf_counter_en key or PerfCounterEn configuration parameter which allow to use English names regardless of system locale.

To find out the numeric equivalents, run **regedit**, then find *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\00*

The registry entry contains information like this:

```
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
....
```

Here you can find the corresponding numbers for each string part of the performance counter, like in '\System\% Processor Time':

```
System → 2
% Processor Time → 6
```

Then you can use these numbers to represent the path in numbers:

```
\2\6
```

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:

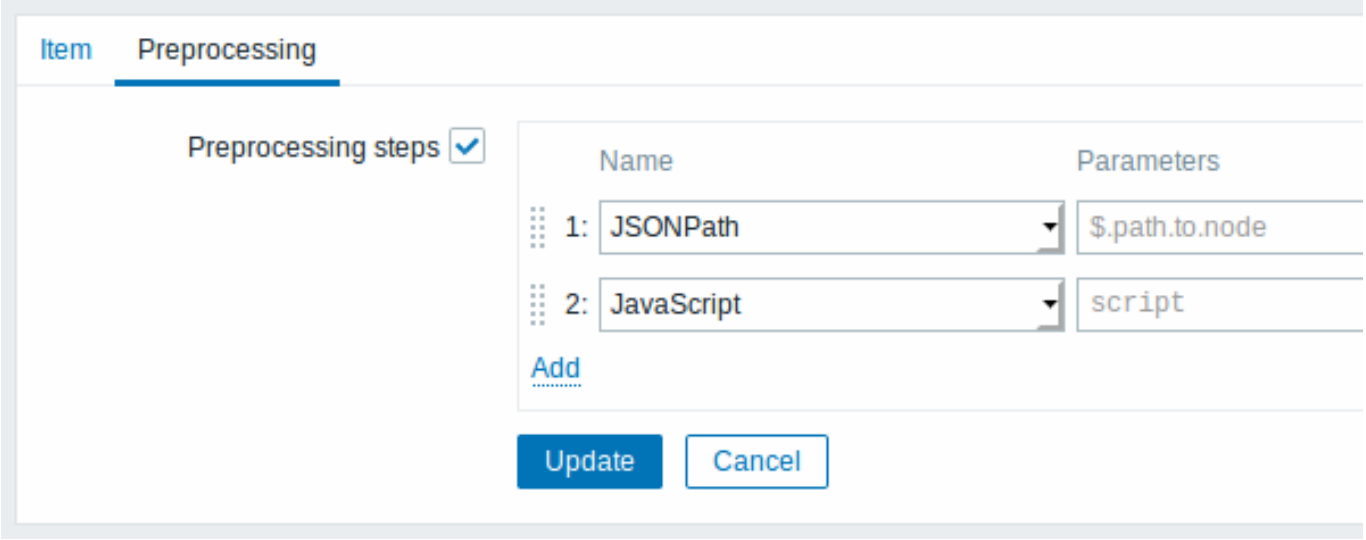
- Mark the checkboxes of the items to update in the list
- Click on *Mass update* below the list
- Navigate to the tab with required attributes (*Item* or *Preprocessing*)
- Mark the checkboxes of the attributes to update
- Enter new values for the attributes


Item
Preprocessing
Type
Original
Host interface
Original
JMX endpoint
Original
URL
Original
Request body type
Original
Request body
Original
Headers
Original
SNMP community
Original
Context name
Original
Security name
Original
Security level
Original
Authentication protocol
Original
Authentication passphrase
Original
Privacy protocol
Original
Privacy passphrase
Original
Port
Original
Type of information
Original
Units
Original
Authentication method
Original
User name
Original
Public key file
Original
Private key file
Original
Password
Original
Update interval
Original
History storage period
7d
Trend storage period
Original
Status
Original
Log time format
Original
Show value
Original
Enable trapping
Original
Allowed hosts
Original
Applications
Add
Replace
Remove
type here to search
Master item
Original
Description
Original
Update
Cancel

The *Applications* option allows to:

- *Add* - add new or existing applications to the items by specifying application name
- *Replace* - replace existing applications of the items with the one(s) specified in this field
- *Remove* - only remove specified applications from the items

The field for specifying applications is auto-complete - starting to type in it offers a dropdown of matching applications. If the application is new, it also appears in the dropdown and it is indicated by *(new)* after the string. Just scroll down to select.



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 values and string representations.

Value mappings can be used in both the Zabbix frontend and notifications sent by email, SMS or script.

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’

Thus, 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 *Show value* field.

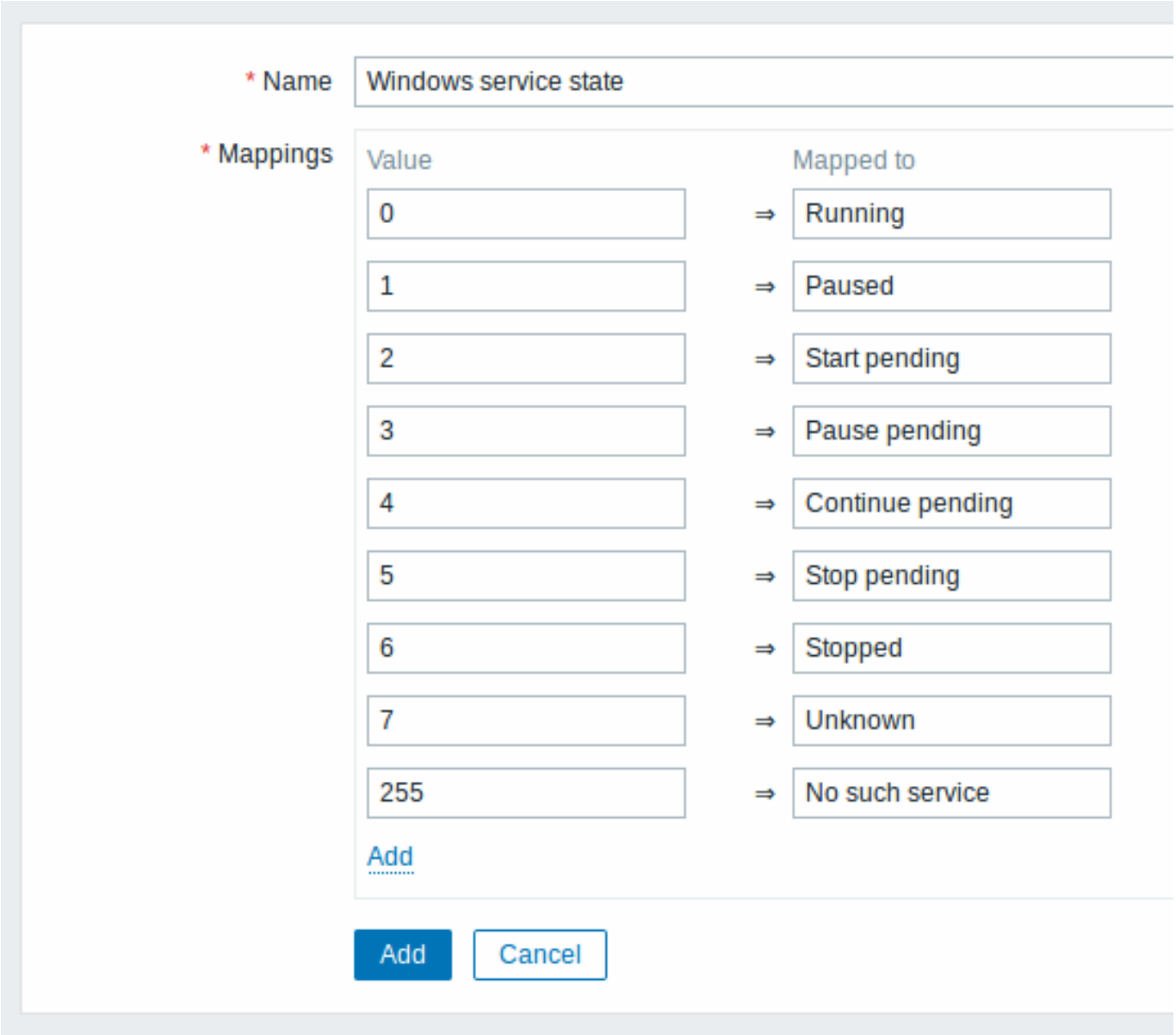
Note:
Value mapping can be used with items having *Numeric (unsigned)*, *Numeric (float)* and *Character* type of information.

Value mappings, starting with Zabbix 3.0, can be exported/imported, either separately, or with the respective template or host.

Configuration

To define a value map:

- Go to: *Administration* → *General*
- Select *Value mapping* from the dropdown
- Click on *Create value map* (or on the name of an existing map)



Parameters of a value map:

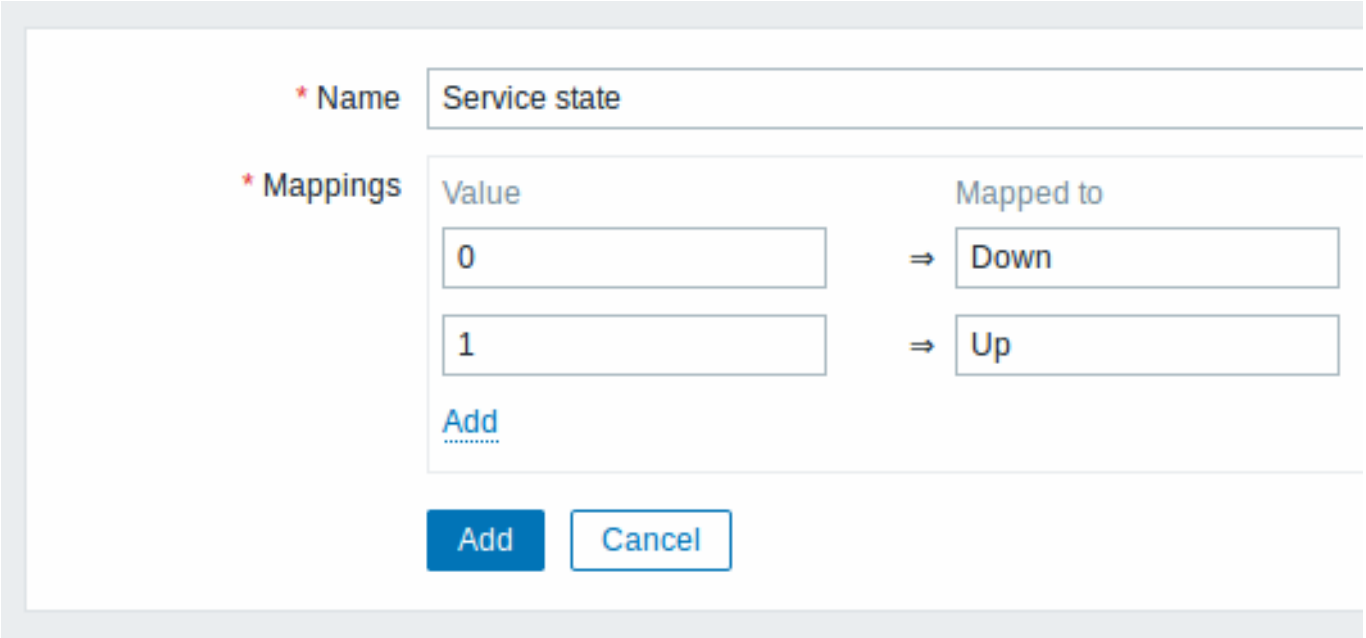
Parameter	Description
Name	Unique name of a set of value mappings.
Mappings	Individual mappings - pairs of numeric values and their string representations.

All mandatory input fields are marked with a red asterisk.

To add a new individual mapping, click on *Add*.

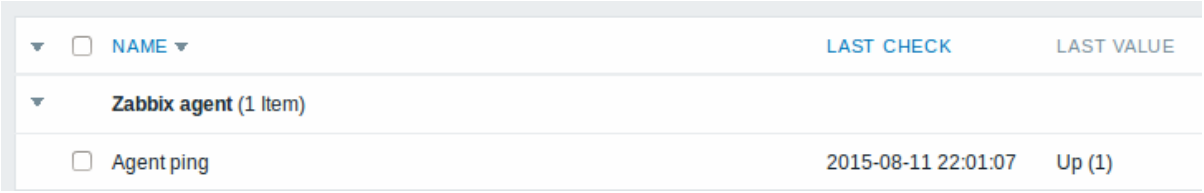
How this works

For example, one of the predefined agent items 'Ping to the server (TCP)' uses an existing value map called 'Service state' to display its values.



In the item configuration form you can see a reference to this value map in the *Show value* field:

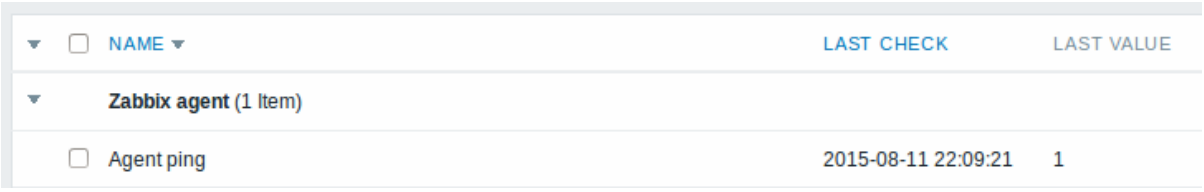
So in *Monitoring* → *Latest data* the mapping is put to use to display 'Up' (with the raw value in parentheses).



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 parenthesis).

Note:
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:



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 Applications

Overview

Applications are used to group items in logical groups.

For example, the *MySQL Server* application can hold all items related to the MySQL server: availability of MySQL, disk space, processor load, transactions per second, number of slow queries, etc.

Applications are also used for grouping web scenarios.

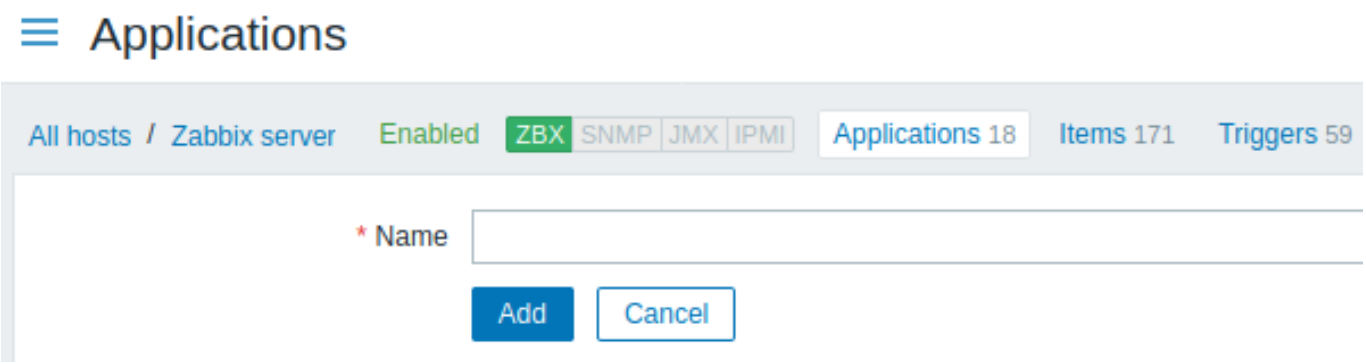
If you are using applications, then in *Monitoring* → *Latest data* you will see items and web scenarios grouped under their respective applications.

Configuration

To work with applications you must first create them and then link items or web scenarios to them.

To create an application, do the following:

- Go to *Configuration* → *Hosts* or *Templates*
- Click on *Applications* next to the required host or template
- Click on *Create application*
- Enter the application name and click on *Add* to save it



You can also create a new application directly in the item properties form.

Items are linked to applications in the item properties form. Select one or more applications the item will belong to.

Web scenarios are linked to applications in the web scenario definition form. Select the application the scenario will belong to.

11 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.

Attention:
Items are not counted in the queue if the item interface becomes unavailable due to connection problems or agent not working properly.

Reading the queue

To read the queue, go to *Administration* → *Queue*.

Queue overview

Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	1	11	1	0	0	0
Zabbix agent (active)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	0	0
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	0	0	0	0	0	0
Zabbix aggregate	0	0	0	0	0	0
External check	0	0	0	0	0	0
Database monitor	0	0	0	0	0	0
HTTP agent	0	0	0	0	0	0

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.

Queue details

Scheduled check	Delayed by	Host	Name	Proxy
2019-09-02 11:46:40	58s	My host	CPU idle time	Remote proxy
2019-09-02 11:46:41	57s	My host	CPU interrupt time	Remote proxy
2019-09-02 11:46:42	56s	My host	CPU iowait time	Remote proxy
2019-09-02 11:46:43	55s	My host	CPU nice time	Remote proxy
2019-09-02 11:46:44	54s	My host	CPU softirq time	Remote proxy
2019-09-02 11:46:45	53s	My host	CPU steal time	Remote proxy
2019-09-02 11:46:46	52s	My host	CPU system time	Remote proxy

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 overview

Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	0	1	1	26	0	0
Zabbix agent (active)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	0	0
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	0	0	0	0	0	0
Zabbix aggregate	0	0	0	0	0	0
External check	0	0	0	0	0	0
Database monitor	0	0	0	0	0	0
HTTP agent	0	0	0	0	0	0
IPMI agent	0	0	0	0	0	0
SSH agent	0	0	0	0	0	0
TELNET agent	0	0	0	0	0	0
JMX agent	0	0	0	0	0	0
Calculated	0	0	0	0	0	0

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.

12 Value cache

Overview

To make the calculation of trigger expressions, calculated/aggregate items and some macros much faster, since Zabbix 2.2 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.

13 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:

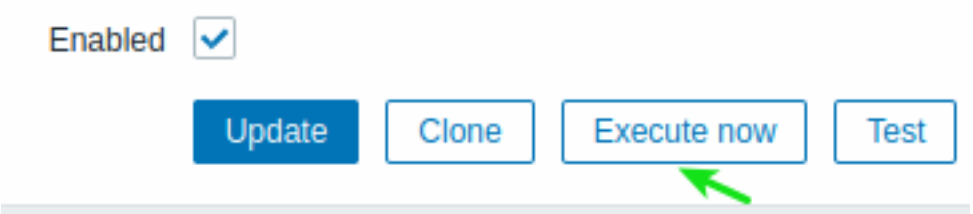
- Zabbix agent (passive)
- SNMPv1/v2/v3 agent
- IPMI agent
- Simple check
- Zabbix internal
- Zabbix aggregate
- External check
- Database monitor
- JMX agent
- SSH agent
- Telnet
- Calculated
- HTTP agent

Attention:
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.

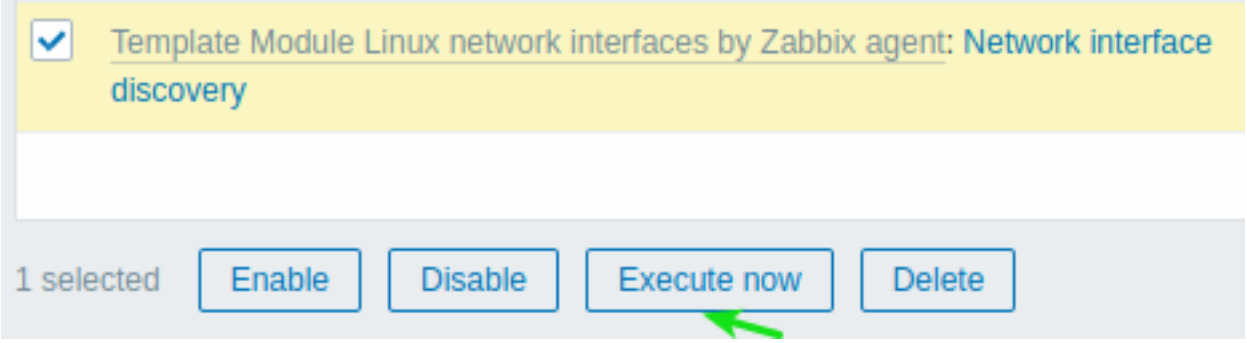
Configuration

To execute a passive check immediately:

- click on *Execute now* in an existing item (or discovery rule) configuration form:



- click on *Execute now* for selected items/rules in the list of items/discovery rules:



In the latter case several items/rules can be selected and "executed now" at once.

14 Plugins

Overview

Plugins provide an option to extend the monitoring capabilities of Zabbix. The plugins are written in Go programming language and supported for Zabbix agent 2 only. They 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:

- single configuration file (all plugin configuration parameters are located in the same file as parameters of the agent itself);
- task queue management with respect to schedule and task concurrency;
- plugin-level timeouts.

Configuring plugins

Common configuration principles and best practices are described in this section.

All plugins are configured using *Plugins.** parameter of the Zabbix agent 2 configuration file. Unlike other agent parameters, it is not a key/value type of parameter. It is a separate section where specific parameters of the plugin can be described.

A typical plugin parameter has the following structure:

- *Plugins.<PluginName>.<SessionName>.<Parameter>=<Value>* used for defining separate sets of parameters for different monitoring targets via named sessions.

Additionally, there are two specific groups of parameters:

- *Plugins.<PluginName>.Default.<Parameter>=<Value>* used for defining default parameter values.
- *Plugins.<PluginName>.<SessionName>.<Parameter>=<Value>* used for defining separate sets of parameters for different monitoring targets via named sessions.

All 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.

Default values

Since Zabbix 5.0.35, you can set default values for the connection-related parameters (URI, username, password, etc.) in the configuration file in the format:

Plugins.<PluginName>.Default.<Parameter>=<Value>

For example, *Plugins.Mysql.Default.Username=zabbix*, *Plugins.MongoDB.Default.Uri=tcp://127.0.0.1:27017*, etc.

If a value for such parameter is not provided in an item key or in the named session parameters, the plugin will use the default value. If a default parameter is also undefined, hardcoded defaults will be used.

> **Note:**
> If an item key does not have any parameters, Zabbix agent 2 will attempt to collect the metric using values defined in the default parameters section.

Named sessions

Named sessions represent an additional level of plugin parameters and can be used to specify separate sets of authentication parameters for each of the instances being monitored. Each named session parameter should have the following structure:

Plugins.<PluginName>.Sessions.<SessionName>.<Parameter>=<Value>

A session name can be used as a connString item key parameter instead of specifying a URI, username, and/or password separately.

In item keys, the first parameter can be either a connString or a URI. If the first key parameter doesn't match any session name, it will be treated as a URI. Note that embedding credentials into a URI is not supported, use named session parameters instead.

The list of available named session parameters depends on the plugin, see Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows) for details.

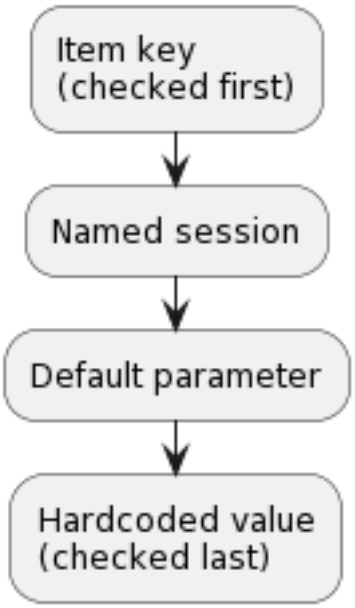
In Zabbix versions before 5.0.34, when providing a connString (session name) in key parameters, item key parameters for username and password must be empty. The values will be taken from the session parameters. If an authentication parameter is not specified for the named session, a hardcoded default value will be used.

Since Zabbix 5.0.34, it is possible to override session parameters by specifying new values in the item key parameters (see example).

Since Zabbix 5.0.35, if a parameter is not defined for the named session, Zabbix agent 2 will use the value defined in the default plugin parameter.

Parameter priority

Since version 5.0.34, Zabbix agent 2 plugins search for connection-related parameter values in the following order:



1. The first item key parameter is compared to session names. If no match is found it is treated as an actual value; in this case, step 3 will be skipped. If a match is found, the parameter value (usually, a URI) must be defined in the named session.
2. Other parameters will be taken from the item key if defined.
3. If an item key parameter (for example, password) is empty, plugin will look for the corresponding named session parameter.
4. If the session parameter is also not specified, the value defined in the corresponding default parameter will be used.
5. If all else fails, the plugin will use the hardcoded default value.

Example 1

Monitoring of two instances “MySQL1” and “MySQL2”.

Configuration parameters:

```
Plugins.Mysql.Sessions.MySQL1.Uri=tcp://127.0.0.1:3306
Plugins.Mysql.Sessions.MySQL1.User=mysql1_user
Plugins.Mysql.Sessions.MySQL1.Password=unique_password
Plugins.Mysql.Sessions.MySQL2.Uri=tcp://192.0.2.0:3306
Plugins.Mysql.Sessions.MySQL2.User=mysql2_user
Plugins.Mysql.Sessions.MySQL2.Password=different_password
```

Item keys: `mysql.ping[MySQL1]`, `mysql.ping[MySQL2]`

Example 2

Providing some of the parameters in the item key (supported since Zabbix 5.0.34).

Configuration parameters:

```
Plugins.Postgres.Sessions.Session1.Uri=tcp://192.0.2.234:5432
Plugins.Postgres.Sessions.Session1.User=old_username
Plugins.Postgres.Sessions.Session1.Password=session_password
```

Item key: `pgsql.ping[session1,new_username,,postgres]`

As a result of this configuration, the agent will connect to PostgreSQL using the following parameters:

- URI from session parameter: *192.0.2.234:5432*
- Username from the item key: *new_username*
- Password from session parameter (since it is omitted in the item key): *session_password*
- Database name from the item key: *postgres*

Example 3

Collecting a metric using default configuration parameters.

Configuration parameters:

```
Plugins.Postgres.Default.Uri=tcp://192.0.2.234:5432
Plugins.Postgres.Default.User=zabbix
Plugins.Postgres.Default.Password=password
```

Item key: `pgsql.ping[,,,postgres]`

As a result of this configuration, the agent will connect to PostgreSQL using the parameters:

- Default URI: *192.0.2.234:5432*
- Default username: *zabbix*
- Default password: *password*
- Database name from the item key: *postgres*

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.

Note:
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 for Zabbix agent 2 are collected by plugins. The following plugins for Zabbix agent 2 are available out-of-the-box:

Plugin name	Description	Supported item keys	Comments
Agent	Metrics of the Zabbix agent being used.	agent.hostname, agent.ping, agent.version	Supported keys have the same parameters as Zabbix agent keys.
Ceph	Ceph monitoring.	ceph.df.details, ceph.osd.stats, ceph.osd.discovery, ceph.osd.dump, ceph.ping, ceph.pool.discovery, ceph.status	Supported keys can be used with Zabbix agent 2 only. See also: - Plugin documentation - Plugin configuration parameters (Unix/Windows)
CPU	System CPU monitoring (number of CPUs/CPU cores, discovered CPUs, utilization percentage).	system.cpu.discovery, system.cpu.num, system.cpu.util	Supported keys have the same parameters as Zabbix agent keys.

Plugin name	Description	Supported item keys	Comments
Docker	Monitoring of Docker containers.	docker.container_info, docker.container_stats, docker.containers, docker.containers.discovery, docker.data_usage, docker.images, docker.images.discovery, docker.info, docker.ping	Supported keys can be used with Zabbix agent 2 only. See also: Plugin configuration parameters (Unix/Windows)
File	File metrics collection.	vfs.file.cksum, vfs.file.contents, vfs.file.exists, vfs.file.md5sum, vfs.file.regexp, vfs.file.regmatch, vfs.file.size, vfs.file.time	Supported keys have the same parameters as Zabbix agent keys.
Kernel	Kernel monitoring.	kernel.maxfiles, kernel.maxproc	Supported keys have the same parameters as Zabbix agent keys.
Log	Log file monitoring.	log, log.count, logrt, logrt.count	Supported keys have the same parameters as Zabbix agent keys. See also: Plugin configuration parameters (Unix/Windows)
Memcached	Memcached server monitoring.	memcached.ping, memcached.stats	Supported keys can be used with Zabbix agent 2 only. See also: - Plugin documentation - Plugin configuration parameters (Unix/Windows)
Modbus	Reads Modbus data.	modbus.get	Supported keys have the same parameters as Zabbix agent keys. See also: - Plugin documentation - Plugin configuration parameters (Unix/Windows)

Plugin name	Description	Supported item keys	Comments
Mongo DB	Monitoring of MongoDB servers and clusters (document-based, distributed database).	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	Supported MongoDB versions: 3.6, 4.0, 4.2, 4.4. Supported keys can be used with Zabbix agent 2 only. See also: - Plugin documentation - Plugin configuration parameters (Unix/Windows)
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 - Plugin configuration parameters (Unix/Windows)
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.

Plugin name	Description	Supported item keys	Comments
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 - Plugin configuration parameters (Unix/Windows)
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. See also: - Plugin documentation - Plugin configuration parameters (Unix/Windows)
Proc	Process CPU utilization percentage.	proc.cpu.util	Supported key has the same parameters as Zabbix agent key.
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 - Plugin configuration parameters (Unix/Windows)

Plugin name	Description	Supported item keys	Comments
Smart	S.M.A.R.T. monitoring.	smart.attribute.discovery, smart.disk.discovery, smart.disk.get	Sudo/root access rights to smartctl are required for the user executing Zabbix agent 2. The minimum required smartctl version is 7.1. Supported keys can be used with Zabbix agent 2 only on Linux/Windows, both as a passive and active check.
Swap	Swap space size in bytes/percentage.	system.swap.size	Supported key has the same parameters as Zabbix agent key.
SystemRun	Runs specified command.	system.run	Supported key has the same parameters as Zabbix agent key. See also: Plugin configuration parameters (Unix/Windows)
Systemd	Monitoring of systemd services.	systemd.unit.discovery, systemd.unit.get, systemd.unit.info	Supported keys can be used with Zabbix agent 2 only.
TCP	TCP connection availability check.	net.tcp.port	Supported key has the same parameters as Zabbix agent key.
UDP	Monitoring of the UDP services availability and performance.	net.udp.service, net.udp.service.perf	Supported keys have the same parameters as Zabbix agent keys.
Uname	Retrieval of information about the system.	system.hostname, system.sw.arch, system.uname	Supported keys have the same parameters as Zabbix agent keys.
Uptime	System uptime metrics collection.	system.uptime	Supported key has the same parameters as Zabbix agent key.
VFSDev	VFS metrics collection.	vfs.dev.discovery, vfs.dev.read, vfs.dev.write	Supported keys have the same parameters as Zabbix agent keys.

Plugin name	Description	Supported item keys	Comments
WebCertificate	Monitoring of TLS/SSL website certificates.	web.certificate.get	Supported key can be used with Zabbix agent 2 only. Supported since Zabbix 5.0.15
WebPage	Web page monitoring.	web.page.get, web.page.perf, web.page.regexp	Supported keys have the same parameters as Zabbix agent keys.
ZabbixAsync	Asynchronous metrics collection.	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	Supported keys have the same parameters as Zabbix agent keys.
ZabbixStats	Zabbix server/proxy internal metrics or number of delayed items in a queue.	zabbix.stats	Supported keys have the same parameters as Zabbix agent keys.
ZabbixSync	Synchronous metrics collection.	net.dns, net.dns.record, net.tcp.service, net.tcp.service.perf, proc.mem, proc.num, system.hw.chassis, system.hw.devices, system.sw.packages, system.users.num, vfs.dir.count, vfs.dir.size, vfs.fs.get, vfs.fs.inode, vfs.fs.size, vm.memory.size.	Supported keys have the same parameters as Zabbix agent keys.

15 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;
- 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,*]
```

Attention:
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

Pattern	Description	Matches	No match
*	Matches all possible keys with or without parameters.	Any	None
vfs.file.contents	Matches `vfs.file.contents` without parameters.	vfs.file.contents	vfs.file.contents[/etc/passwd]
vfs.file.contents[]	Matches `vfs.file.contents` with empty parameters.	vfs.file.contents[]	vfs.file.contents
vfs.file.contents[*]	Matches `vfs.file.contents` with any parameters; will not match `vfs.file.contents` without square brackets.	vfs.file.contents[] vfs.file.contents[/path/to/file]	vfs.file.contents

Pattern	Description	Matches	No match
vfs.file.contents[/etc/passwd,]*	Matches `vfs.file.contents` with first parameters matching /etc/passwd and all other parameters having any value (also empty).	vfs.file.contents[/etc/passwd] vfs.file.contents[/etc/passwd,utf8]	vfs.file.contents[/var/log/zabbix_server.log] vfs.file.contents[]
*vfs.file.contents[*passwd*]*	Matches `vfs.file.contents` with first parameter matching *passwd* and no other parameters.	vfs.file.contents[/etc/passwd]	vfs.file.contents[/etc/passwd,] vfs.file.contents[/etc/passwd, utf8]
*vfs.file.contents[*passwd*,*]*	Matches `vfs.file.contents` with only first parameter matching *passwd* and all following parameters having any value (also empty).	vfs.file.contents[/etc/passwd] vfs.file.contents[/etc/passwd, utf8]	vfs.file.contents[/etc/passwd] vfs.file.contents[/tmp/test]
vfs.file.contents[/var/log/zabbix_server.log,,abc]*	Matches `vfs.file.contents` with first parameter matching /var/log/zabbix_server.log, third parameter matching 'abc' and any (also empty) second parameter.	vfs.file.contents[/var/log/zabbix_server.log,,abc] vfs.file.contents[/var/log/zabbix_server.log,utf8,abc]	vfs.file.contents[/var/log/zabbix_server.log]
vfs.file.contents[/etc/passwd,utf8]	Matches `vfs.file.contents` with first parameter matching /etc/passwd, second parameter matching 'utf8' and no other arguments.	vfs.file.contents[/etc/passwd,utf8]	vfs.file.contents[/etc/passwd,] vfs.file.contents[/etc/passwd,utf16]
*vfs.file.**	Matches any keys starting with `vfs.file.` without any parameters.	vfs.file.contents vfs.file.size	vfs.file.contents[] vfs.file.size[/var/log/zabbix_server.log]
vfs.file.[*]*	Matches any keys starting with `vfs.file.` with any parameters.	vfs.file.size.bytes[] vfs.file.size[/var/log/zabbix_server.log, utf8]	vfs.file.size.bytes
vfs..contents*	Matches any key starting with `vfs.` and ending with `.contents` without any parameters.	vfs.mount.point.file.contents vfs..contents	vfs.contents

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 frontend 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.

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 its state to PROBLEM.

A trigger may have the following states:

State	Description
OK	This is a normal trigger state.
Problem	Something has happened. For example, the processor load is too high.
Unknown	The trigger value cannot be calculated. See Unknown state.

Trigger state (the expression) is recalculated every time Zabbix server receives a new value that is part of the expression.

Triggers are evaluated based on history data only; trend data are never considered.

If time-based functions (**nodata()**, **date()**, **dayofmonth()**, **dayofweek()**, **time()**, **now()**) are used in the expression, the trigger is recalculated every 30 seconds by a Zabbix *history syncer* process. If both time-based and non-time-based functions are used in an expression, it is recalculated when a new value is received **and** every 30 seconds.

You can build trigger expressions with different degrees of complexity.

Unknown state

It is possible that an unknown operand appears in a trigger expression if:

- an unsupported item is used
- the function evaluation for a supported item results in an error

In this case a trigger generally evaluates to ”unknown” (although there are some exceptions). For more details, see Expressions with unknown operands.

It is possible to get notified on unknown triggers.

1 Configuring a trigger

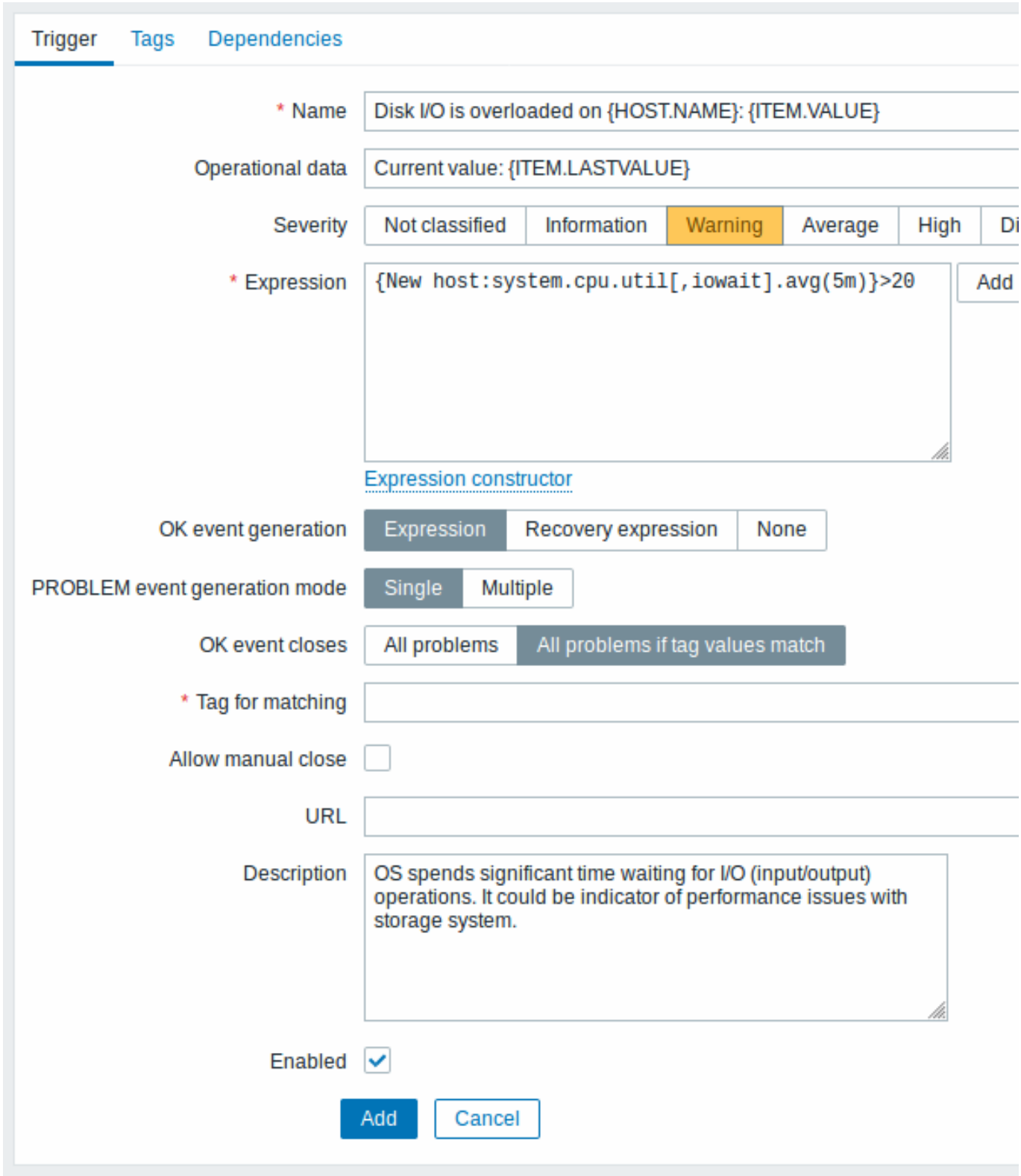
Overview

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

Configuration

The **Trigger** tab contains all the essential trigger attributes.

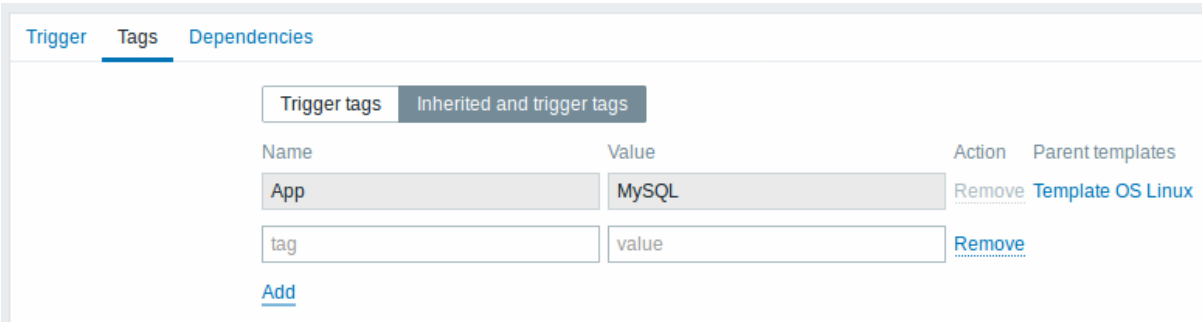


All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Trigger name. Supported macros are: {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {ITEM.VALUE}, {ITEM.LASTVALUE} and {$MACRO} user macros. **$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 "Processor load above $1 on {HOST.NAME}" will automatically change to "Processor load above 5 on New host" if the expression is {New host:system.cpu.load[percpu,avg1].last()}>5 In this location, resolved {ITEM.VALUE} and {ITEM.LASTVALUE} macro values are truncated to 20 characters. To see the entire values you may use macro functions with these macros, e.g. {{ITEM.VALUE}.regsub("(.*)", \1)}, {{ITEM.LASTVALUE}.regsub("(.*)", \1)} as a workaround.
Operational data	Operational data allow to define arbitrary strings along with macros. The macros will resolve dynamically to real time data in *Monitoring* → *Problems*. While macros in the trigger name (see above) will resolve to their values at the moment of a problem happening and will become the basis of a static problem name, the macros in the operational data maintain the ability to display the very latest information dynamically. Supported macros are: {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT}, {ITEM.VALUE}, {ITEM.LASTVALUE} and {$MACRO} user macros. This field is supported since Zabbix 4.4.
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*.
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. Supported since Zabbix 3.2.0.
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*. Supported since Zabbix 3.2.0.
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.

Parameter	Description
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 Supported since Zabbix 3.2.0.
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. Supported since Zabbix 3.2.0.
Allow manual close	Check to allow manual closing of problem events generated by this trigger. Manual closing is possible when acknowledging problem events. Supported since Zabbix 3.2.0.
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. Supported macros: {EVENT.ID}, {ITEM.VALUE}, {ITEM.LASTVALUE}, {TRIGGER.ID}, several {HOST.*} macros, user macros (secret macros in URLs will not be resolved). In this location, resolved {ITEM.VALUE} and {ITEM.LASTVALUE} macro values 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.
Description	Text field used to provide more information about this trigger. May contain instructions for fixing specific problem, contact detail of responsible staff, etc. *Starting with Zabbix 2.2*, the description may contain the same set of macros as 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.

Parameter	Description
Name/Value	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. Supported since Zabbix 3.2.0.

The **Dependencies** tab contains all the dependencies of the trigger.

Click on *Add* to add a new dependency.

Note:
You can also configure a trigger by opening an existing one, pressing the *Clone* button and then saving under a different name.

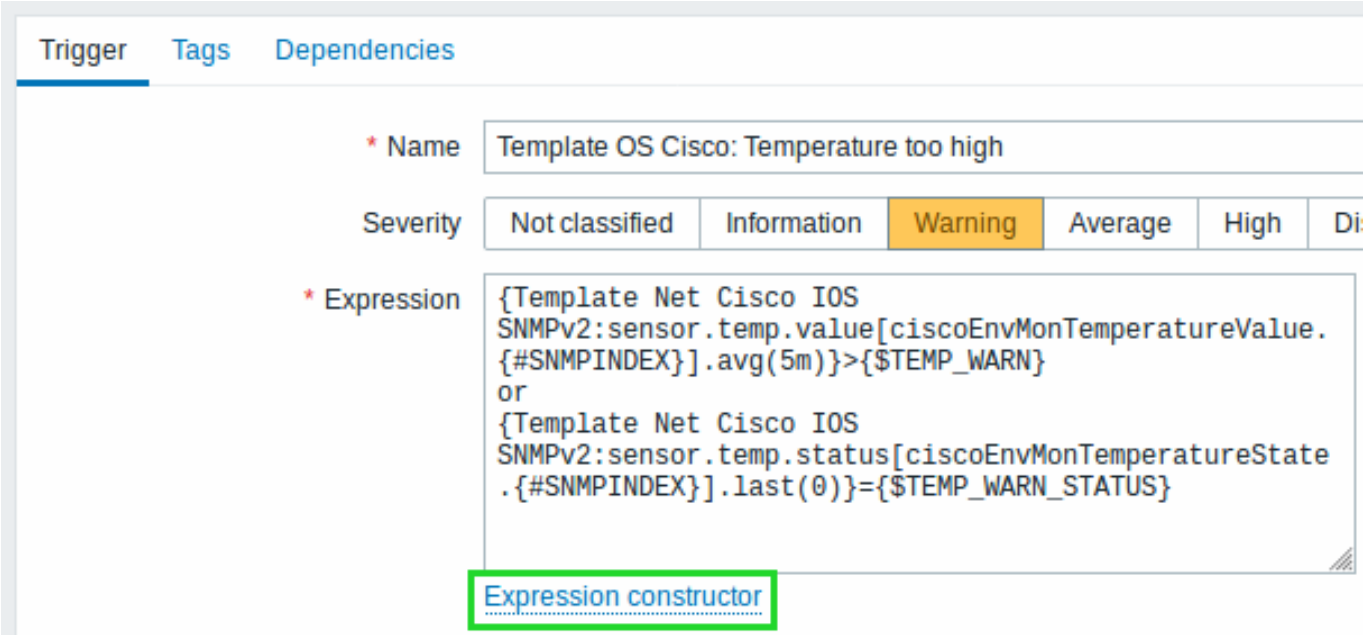
Testing expressions

It is possible to test the configured trigger expression as to what the expression result would be depending on the received value.

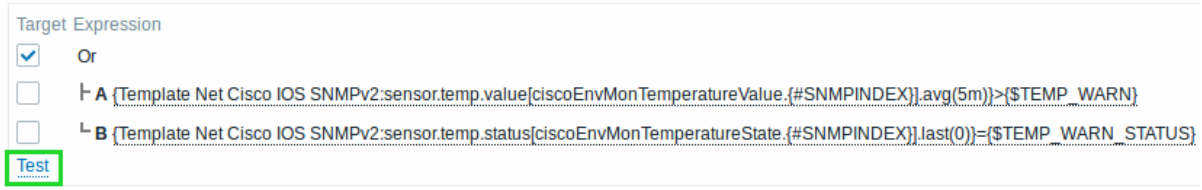
Following expression from an official template is taken as an example:

```
{Template Net Cisco IOS SNMPv2:sensor.temp.value[ciscoEnvMonTemperatureValue.{#SNMPINDEX}].avg(5m)}>{$TEMP
or
{Template Net Cisco IOS SNMPv2:sensor.temp.status[ciscoEnvMonTemperatureState.{#SNMPINDEX}].last(0)}={$TEM
```

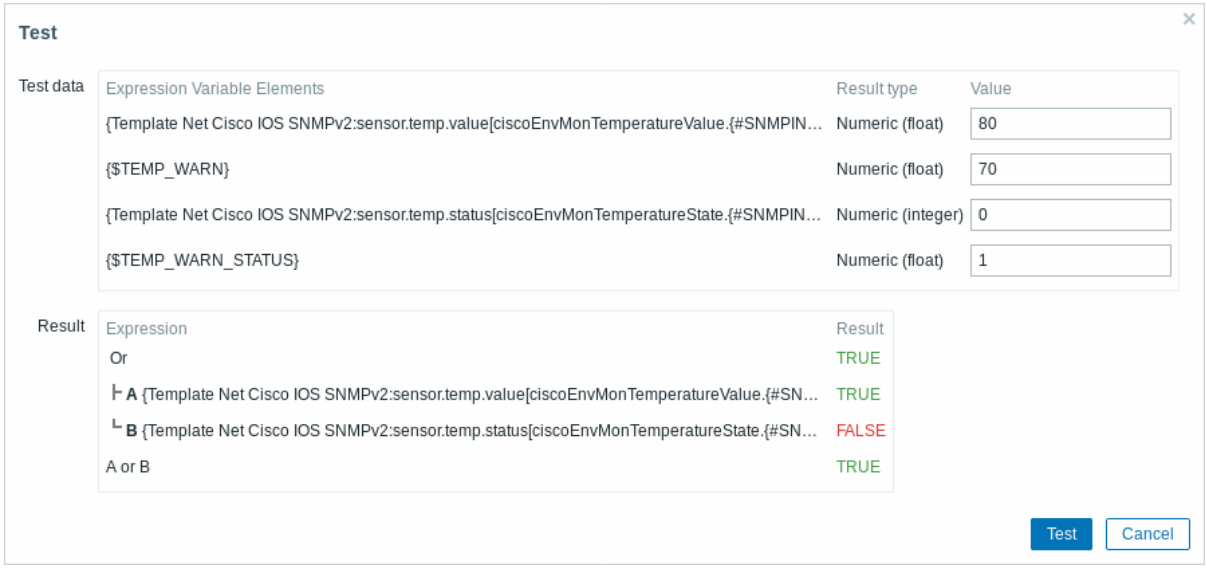
To test the expression, click on *Expression constructor* under the expression field.



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” result means the specified expression is correct. In this particular case A, ”80” is greater than {$TEMP_WARN} specified value, ”70” in this example. Respectively, ”TRUE” result appears.

”FALSE” result means the specified expression is incorrect. In this particular case B, {$TEMP_WARN_STATUS}, ”1” in this example, needs to be equal with specified ”0” value, that is wrong. Respectively, ”FALSE” result appears.

Chosen Expression type is ”OR”/”TRUE”. If at least one of the specified conditions (A or B in this case) is TRUE, overall result will be TRUE as well. That means, 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 useful expression might look like:

```
{<server>:<key>.<function>(<parameter>)}<operator><constant>
```

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

Trigger functions allow to reference the collected values, current time and other factors.

A complete list of supported functions is available.

Typically functions return numeric values. However, returning strings for string comparison is also possible with **=** and **<>** operators; see operators and trigger examples for more details.

Function parameters

Most of numeric functions accept the number of seconds as a parameter.

You may use the prefix **#** to specify that a parameter has a different meaning:

FUNCTION CALL	MEANING
sum(600)	Sum of all values in no more than the latest 600 seconds
sum(#5)	Sum of all values in no more than the latest 5 values

The function **last** uses a different meaning for values when prefixed with the hash mark - it makes it choose the n-th previous value, so given the values 3, 7, 2, 6, 5 (from most recent to least recent), **last(#2)** would return *7* and **last(#5)** would return *5*.

Several functions support an additional, second `time_shift` parameter. This parameter allows to reference data from a period of time in the past. For example, **avg(1h,1d)** will return the average value for an hour one day ago.

You can use the supported unit symbols in trigger expressions, for example '5m' (minutes) instead of '300' seconds or '1d' (day) instead of '86400' seconds. '1K' will stand for '1024' bytes.

Numbers with a '+' sign are not supported.

Operators

The following operators are supported for triggers **(in descending priority of execution)**:

Priority	Operator	Definition	**Notes for unknown values**	Force cast operand to float [1]
1	**-**	Unary minus	**-**Unknown → Unknown	Yes
2	**not**	Logical NOT	**not** Unknown → Unknown	Yes
3	*	Multiplication	0 * Unknown → Unknown (yes, Unknown, not 0 - to not lose Unknown in arithmetic operations) 1.2 * Unknown → Unknown	Yes
	/	Division	Unknown / 0 → error Unknown / 1.2 → Unknown 0.0 / Unknown → Unknown	Yes
4	**+**	Arithmetical plus	1.2 **+** Unknown → Unknown	Yes
	-	Arithmetical minus	1.2 **-** Unknown → Unknown	Yes
5	**<**	Less than. The operator is defined as: A<B ⇔ (A<B-0.000001)	1.2 **<** Unknown → Unknown	Yes

Priority	Operator	Definition	**Notes for unknown values**	Force cast operand to float [1]
	<=	Less than or equal to. The operator is defined as: A<=B ⇔ (A≤B+0.000001)	Unknown **<=** Unknown → Unknown	Yes
	>	More than. The operator is defined as: A>B ⇔ (A>B+0.000001)		Yes
	>=	More than or equal to. The operator is defined as: A>=B ⇔ (A≥B-0.000001)		Yes
6	**=**	Is equal. The operator is defined as: A=B ⇔ (A≥B-0.000001) and (A≤B+0.000001)		No [1]

Priority	Operator	Definition	Notes for unknown values	Force cast operand to float [1]
	<>	Not equal. The operator is defined as: A<>B ⇔ (A<B-0.000001) or (A>B+0.000001)		No [1]
7	**and**	Logical AND	0 **and** Unknown → 0 1 **and** Unknown → Unknown Unknown **and** Unknown → Unknown	Yes
8	**or**	Logical OR	1 **or** Unknown → 1 0 **or** Unknown → Unknown Unknown **or** Unknown → Unknown	Yes

[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 **-(-1)** and **not (not 1)** should be used instead of **--1** and **not not 1**).

Evaluation result:

- **<**, **<=**, **>**, **>=**, **=**, **<>** 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

Processor load is too high on www.example.com

```
{www.example.com:system.cpu.load[all,avg1].last()}>5
```

'www.example.com:system.cpu.load[all,avg1]' gives a short name of the monitored parameter. It specifies that the server is 'www.example.com' and the key being monitored is 'system.cpu.load[all,avg1]'. By using the function 'last()', we are referring to the most recent value. Finally, '>5' means that the trigger is in the PROBLEM state whenever the most recent processor load measurement from www.example.com is greater than 5.

Example 2

www.example.com is overloaded

```
{www.example.com:system.cpu.load[all,avg1].last()}>5 or {www.example.com:system.cpu.load[all,avg1].min(10m
```

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

Use of function diff:

```
{www.example.com:vfs.file.cksum[/etc/passwd].diff()}=1
```

The expression is true when the previous value of checksum of /etc/passwd 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:

```
{www.example.com:net.if.in[eth0,bytes].min(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:

```
{smtp1.example.com:net.tcp.service[smtp].last()}=0 and {smtp2.example.com:net.tcp.service[smtp].last()}=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 str():

```
{example.example.com:agent.version.str("beta8")}=1
```

The expression is true if Zabbix agent has version beta8 (presumably 1.0beta8).

Example 7

Server is unreachable

```
{example.example.com:icmpping.count(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():

```
{example.example.com:tick.nodata(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():

```
{zabbix:system.cpu.load[all,avg1].min(5m)}>2 and {zabbix:system.cpu.load[all,avg1].time()}>000000 and {zab
```

The trigger may change its state to true only at night time (00:00 - 06:00).

Example 10

CPU activity at any time with exception

Use of function time() and **not** operator:

```
{zabbix:system.cpu.load[all,avg1].min(5m)}>2
and not ({zabbix:system.cpu.load[all,avg1].dayofweek(0)}=7 and {zabbix:system.cpu.load[all,avg1].time(0)}>
and not ({zabbix:system.cpu.load[all,avg1].dayofweek(0)}=1 and {zabbix:system.cpu.load[all,avg1].time(0)}<
```

The trigger may change its state to true at any time, except for 2 hours on a week change (Sunday, 23:00 - Monday, 01:00).

Example 11

Check if client local time is in sync with Zabbix server time

Use of function fuzzytime():

```
{MySQL_DB:system.localtime.fuzzytime(10)}=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 12

Comparing average load today with average load of the same time yesterday (using a second `time_shift` parameter).

```
{server:system.cpu.load.avg(1h)}/{server:system.cpu.load.avg(1h,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 13

Using the value of another item to get a trigger threshold:

```
{Template PfSense:hrStorageFree[{#SNMPVALUE}].last()}<{Template PfSense:hrStorageSize[{#SNMPVALUE}].last()
```

The trigger will fire if the free storage drops below 10 percent.

Example 14

Using evaluation result to get the number of triggers over a threshold:

```
({server1:system.cpu.load[all,avg1].last()}>5) + ({server2:system.cpu.load[all,avg1].last()}>5) + ({server
```

The trigger will fire if at least two of the triggers in the expression are over 5.

Example 15

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

```
{Riga Zabbix server:vfs.file.contents[/etc/os-release].last()}<>{London Zabbix server:vfs.file.contents[/e
```

Example 16

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:

```
net.dns.record[8.8.8.8,{$WEBSITE_NAME},{$DNS_RESOURCE_RECORD_TYPE},2,1]
```

with macros defined as

```
{$WEBSITE_NAME} = example.com
{$DNS_RESOURCE_RECORD_TYPE} = MX
```

and normally returns:

```
example.com             MX       0 mail.example.com
```

So our trigger expression to detect if the DNS query result deviated from the expected result is:

```
{Zabbix server:net.dns.record[8.8.8.8,{$WEBSITE_NAME},{$DNS_RESOURCE_RECORD_TYPE},2,1].last()}<>"{$WEBSITE
```

Notice the quotes around the second operand.

Example 17

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

```
{Zabbix server:vfs.file.contents[/tmp/hello].last()}="\\\" //hello ?\\\""
```

Notice how \ and ” characters are escaped when the string gets compared directly.

Option 2) use a macro

```
{$HELLO_MACRO} = \" //hello ?\"
```

in the expression:

```
{Zabbix server:vfs.file.contents[/tmp/hello].last()}={$HELLO_MACRO}
```

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.

Warning:
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:

```
{server:temp.last()}>20
```

Recovery expression:

```
{server:temp.last()}<=15
```

Example 2

Free disk space is too low.

Problem expression: it is less than 10GB for last 5 minutes

```
{server:vfs.fs.size[/,free].max(5m)}<10G
```

Recovery expression: it is more than 40GB for last 10 minutes

```
{server:vfs.fs.size[/,free].min(10m)}>40G
```

Expressions with unknown operands

Generally an unknown operand (such as an unsupported item) in the expression will immediately render the trigger value to Unknown.

However, in some cases unknown operands (unsupported items, function errors) are admitted into expression evaluation:

- The nodata() function is evaluated regardless of whether the referenced item is supported or not.
- Logical expressions with OR and AND can be evaluated to known values in two cases regardless of unknown operands:
  - **Case 1**: ”1 or unsupported_item1.some_function() or unsupported_item2.some_function() or ...” can be evaluated to known result (’1’ or ”Problem”),

- **Case 2**: "`0 and unsupported_item1.some_function() and unsupported_item2.some_function() and ...`" can be evaluated to known result ('0' or "OK").
  Zabbix tries to evaluate such logical expressions by taking unsupported items as unknown operands. In the two cases above a known value will be produced ("Problem" or "OK", respectively); in **all other** cases the trigger will evaluate to `Unknown`.

- If the function evaluation for a supported item results in error, the function value becomes `Unknown` and it takes part as unknown operand in further expression evaluation.

Note that unknown operands may "disappear" only in logical expressions as described above. In arithmetic expressions unknown operands always lead to the result `Unknown` (except division by 0).

Attention:
An expression that results in `Unknown` does not change the trigger state ("Problem/OK"). So, if it was "Problem" (see Case 1), it stays in the same problem state even if the known part is resolved ('1' becomes '0'), because the expression is now evaluated to `Unknown` and that does not change the trigger state.

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 a 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.

While the parent trigger is in the PROBLEM state, its dependents may report values that cannot be trusted. Therefore dependent triggers will not be re-evaluated until the parent trigger (the router in the example above):

- goes back from 'PROBLEM' to 'OK' state;
- 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 the dependent trigger;
- is disabled, has a disabled item or a disabled item host

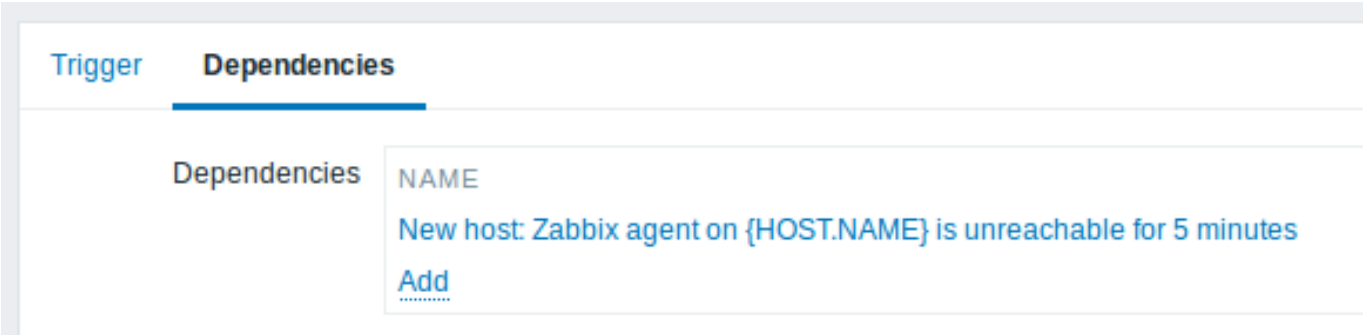
In all of the cases mentioned above, the dependent trigger (server) will be re-evaluated only when a new metric for it is received. This means that the dependent trigger may not be updated immediately.

Also:

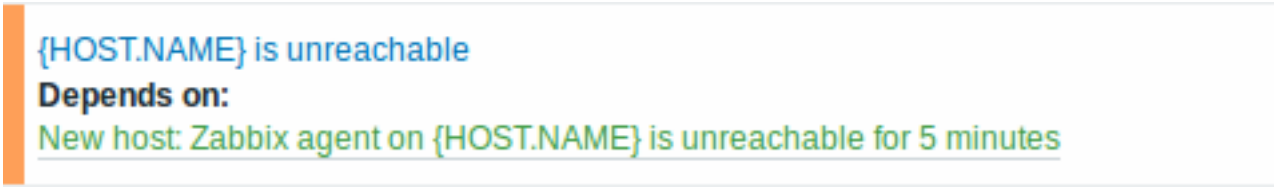
- Trigger dependency may be added from any host trigger to any other host trigger, as long as it wouldn't result in a circular dependency.
- Trigger dependency may be added from a template to a template. If a trigger from template A depends on a trigger from template B, template A may only be linked 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 represents the level of importance of a trigger.

Zabbix supports the following default trigger severities.

Severity	Color	Description
Not classified	Gray	Can be used where the severity level of an event is unknown, has not been determined, is not part of the regular monitoring scope, etc., for example, during initial configuration, as a placeholder for future assessment, or as part of an integration process.
Information	Light blue	Can be used for informational events that do not require immediate attention, but can still provide valuable insights.
Warning	Yellow	Can be used to indicate a potential issue that might require investigation or action, but that is not critical.
Average	Orange	Can be used to indicate a significant issue that should be addressed relatively soon to prevent further problems.
High	Light red	Can be used to indicate critical issues that need immediate attention to avoid significant disruptions.
Disaster	Red	Can be used to indicate a severe incident that requires immediate action to prevent, for example, system outages or data loss.

Note:
Trigger severity names and colors can be customized.

Trigger 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 for triggers of *High* and *Disaster* severity, and Email for triggers of other severities);
- limiting actions by conditions against trigger severities.

5 Customizing trigger severities

Trigger severity names and colors for severity related GUI elements can be configured in *Administration* → *General* → *Trigger severities*. Colors are shared among all GUI themes.

Translating customized severity names

Attention:
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, 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

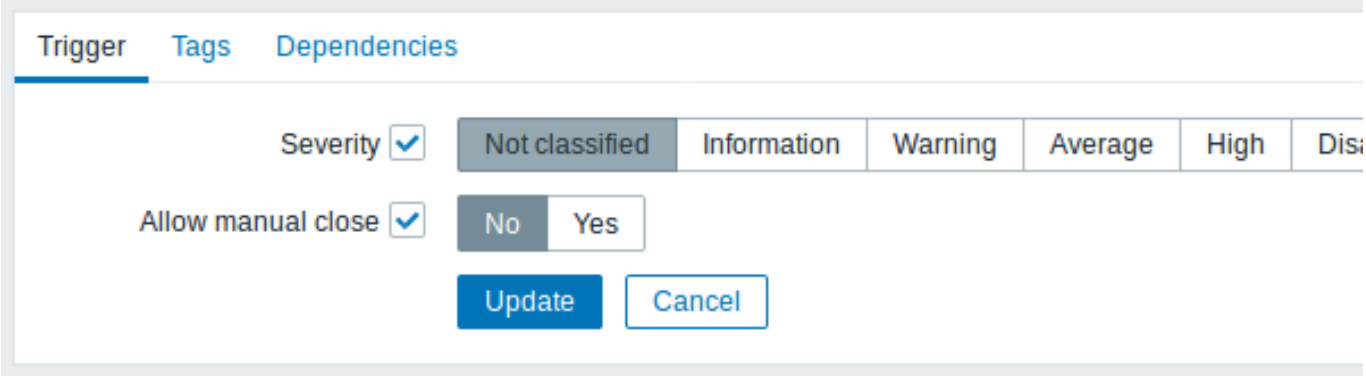
Overview

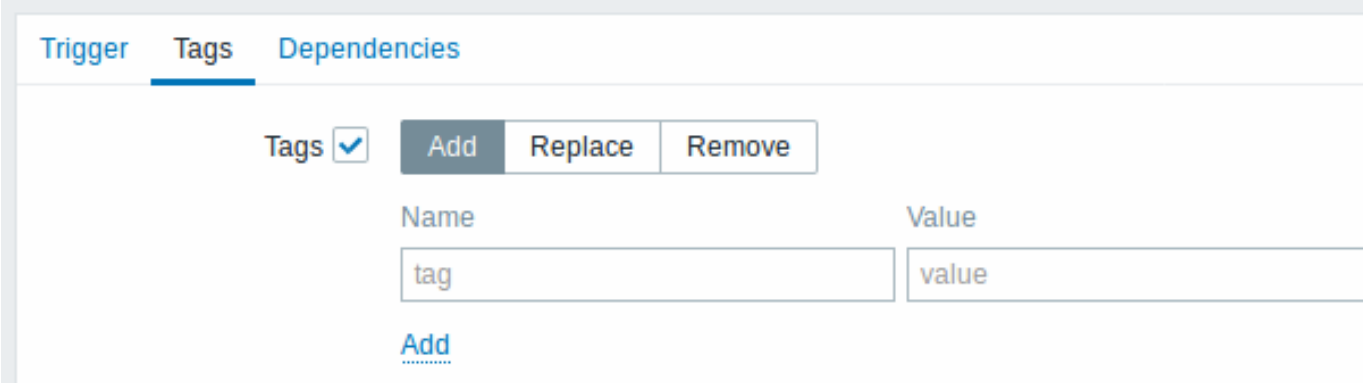
With mass update you may change some attribute for a number of triggers at once, saving you the need to open each individual trigger for editing.

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

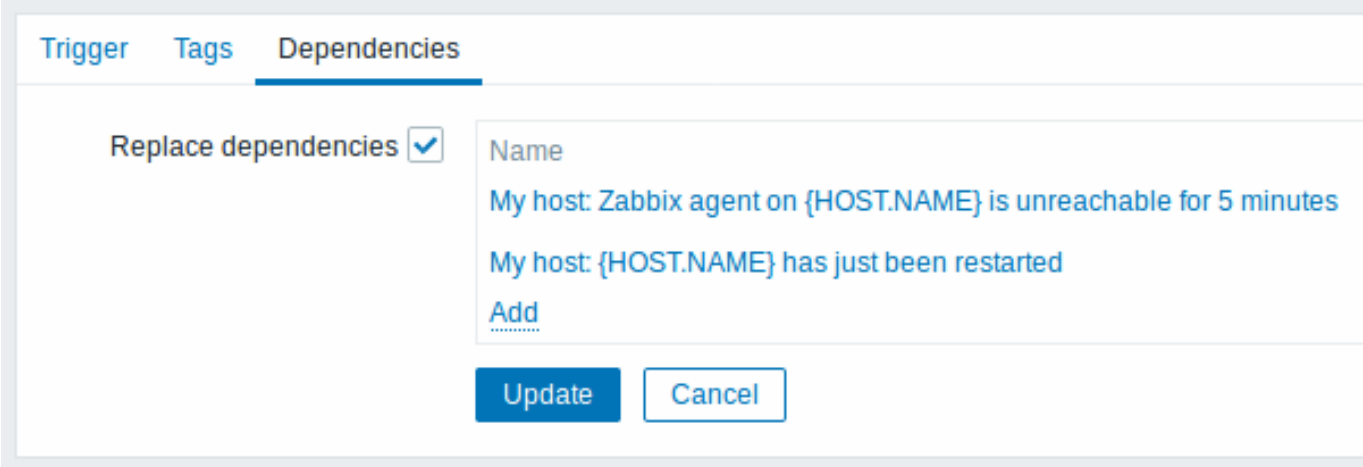




The following options are available when selecting the respective button for tag update:

- *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.



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.

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: trigger must fire when the system after "time to

act" is expected to be in problem state. Second: 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.

Parameters

Both functions use almost the same set of parameters. Use the list of supported functions for reference.

Time interval

First of all you should specify the historic period Zabbix should analyze to come up with prediction. You do it in a familiar way by means of `sec` or `#num` parameter and optional `time_shift` like you do it with **avg**, **count**, **delta**, **max**, **min** and **sum** functions.

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.

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) = `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.

Note:
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]

Fit functions

Default `fit` is the *linear* function. But if your monitored system is more complicated you have more options to choose from.

`fit`	x = f(t)
linear	$x = a + b*t$
polynomialN[2]	$x = a_0 + a_1*t + a_2*t^2 + ... + a_n*t^n$
exponential	$x = a*\exp(b*t)$
logarithmic	$x = a + b*\log(t)$
power	$x = a*t^b$

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.

`mode`	**forecast** result
value	f(now + `time`)
max	$\max_{\text{now <= t <= now + time}}$ f(t)
min	$\min_{\text{now <= t <= now + time}}$ f(t)
delta	*max* - *min*
avg	average of f(t) (now <= t <= now + `time`) according to definition

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 10^9, epoch squared is 10^{18}, 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.

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].

Note:
No warnings or errors are flagged if chosen fit poorly describes provided data or there is just too few data for accurate prediction.

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:

```
{host:vfs.fs.size[/,free].timeleft(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]:

```
{host:vfs.fs.size[/,free].timeleft(1h,,0)}<1h and {host:vfs.fs.size[/,free].timeleft(1h,,0)}<>-1
```

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 `{host:item.forecast(...)}<...` or like `{host:item.forecast(...)}>...`

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 operator **=** works). So add `... or {host:item.forecast(...)}=-1` or `... and {host:item.forecast(...)}<>-1` if you want or don't want to treat -1 as a problem respectively.

See also

- Predictive trigger functions (pdf)

Footnotes

[1] For example, a simple trigger like `{host:item.timeleft(1h,,X)} < 1h` may go into problem state when the item value approaches X and then suddenly recover once value X is reached. If the problem is item value being below X use: `{host:item.last()} < X or {host:item.timeleft(1h,,X)} < 1h` If the problem is item value being above X use: `{host:item.last()} > X or {host:item.timeleft(1h,,X)} < 1h`

[2] Polynomial degree can be from 1 to 6, *polynomial1* is equivalent to *linear*. However, use higher degree polynomials with caution. If the evaluation period contains less points than needed to determine polynomial coefficients, polynomial degree will be lowered (e.g *polynomial5* is requested, but there are only 4 points, therefore *polynomial3* will be fitted).

[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 *avg* involves computing polynomial antiderivative (analytically). Computing *max*, *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: `{host:vfs.fs.size[/,free].time and ({TRIGGER.VALUE}=0 and {host:vfs.fs.size[/,free].timeleft(1h,,0)}<>-1 or {TRIGGER.VALUE}=1)`

4 Events

Overview

There are several types of events generated in Zabbix:

- trigger events - whenever a trigger 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

Note:
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

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:

Event	When generated
Service Up	Every time Zabbix detects active service.
Service Down	Every time Zabbix cannot detect service.
Host Up	If at least one of the services is UP for the IP.
Host Down	If all services are not responding.
Service Discovered	If the service is back after downtime or discovered for the first time.
Service Lost	If the service is lost after being up.
Host Discovered	If host is back after downtime or discovered for the first time.
Host Lost	If host is lost after being up.

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.

Note:
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:

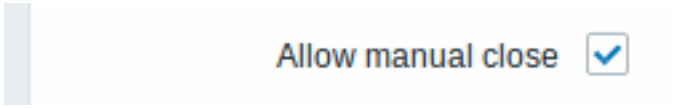
- When new data arrive for any item included in the trigger expression (note that the values discarded by a throttling preprocessing step are not considered as received and will not cause trigger expression to be re-evaluated);
- When time-based functions are used in the expression. Complete time-based function list can be found on Triggers page.

Configuration

Two steps are required to close a problem manually.

Trigger configuration

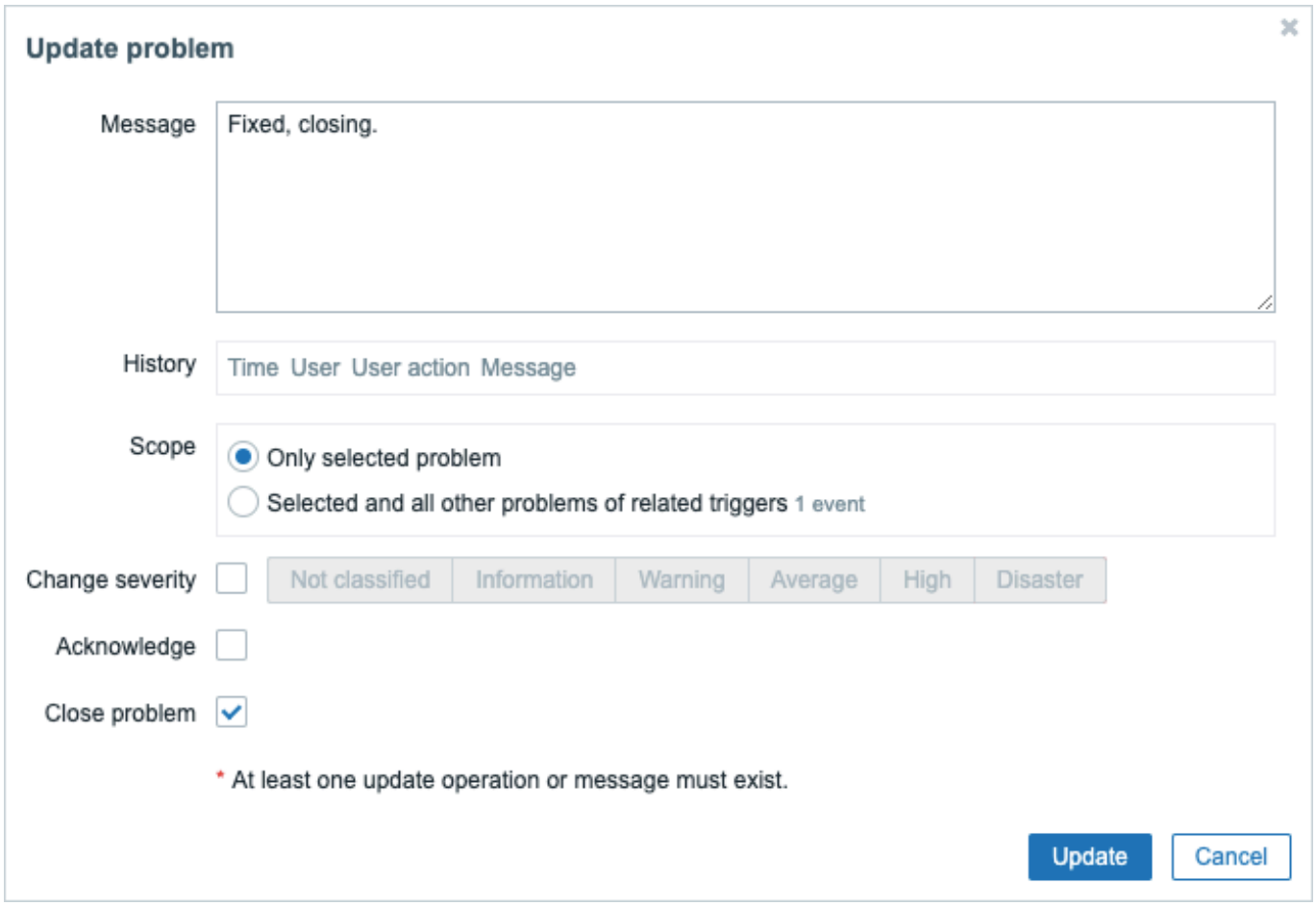
In trigger configuration, enable the *Allow manual close* option.



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 the event tags. 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
```

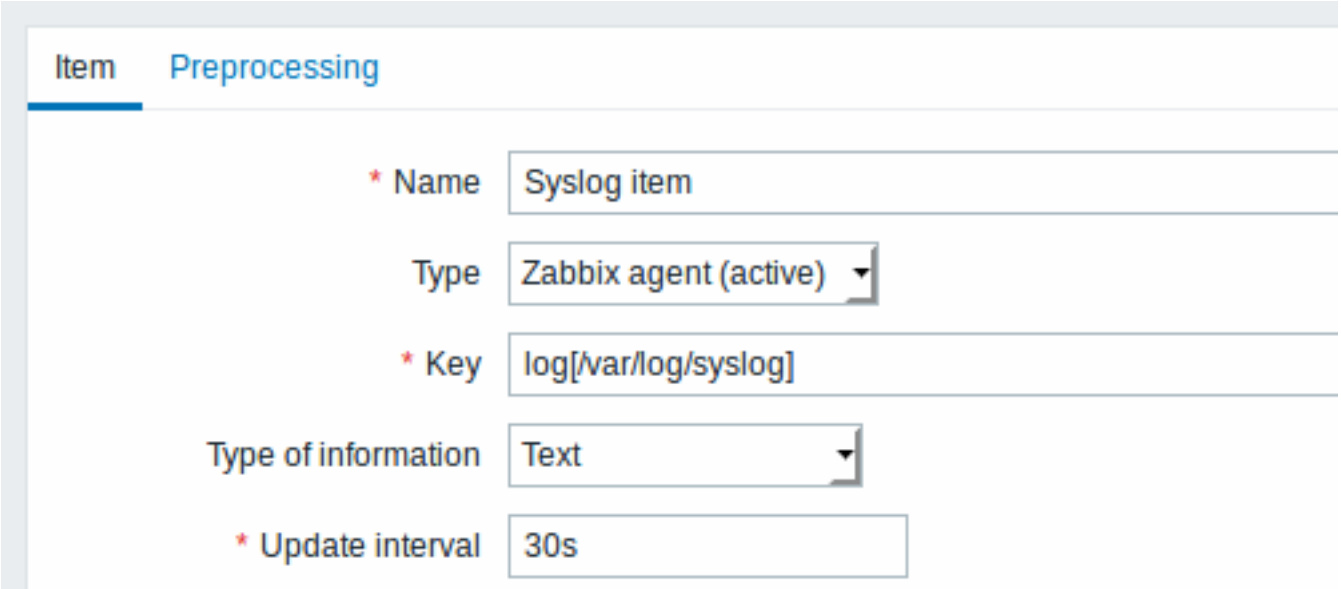
To do this you need to tag these related events as, for example, ”Application 1” and ”Application 2”. That can be done by applying a regular expression to the log line to extract the tag value. Then, when events are created, they are tagged ”Application 1” and ”Application 2” respectively and problem 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:

```
{My host:log[/var/log/syslog].regexp("Stopping")}=1
```

Attention:
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'.

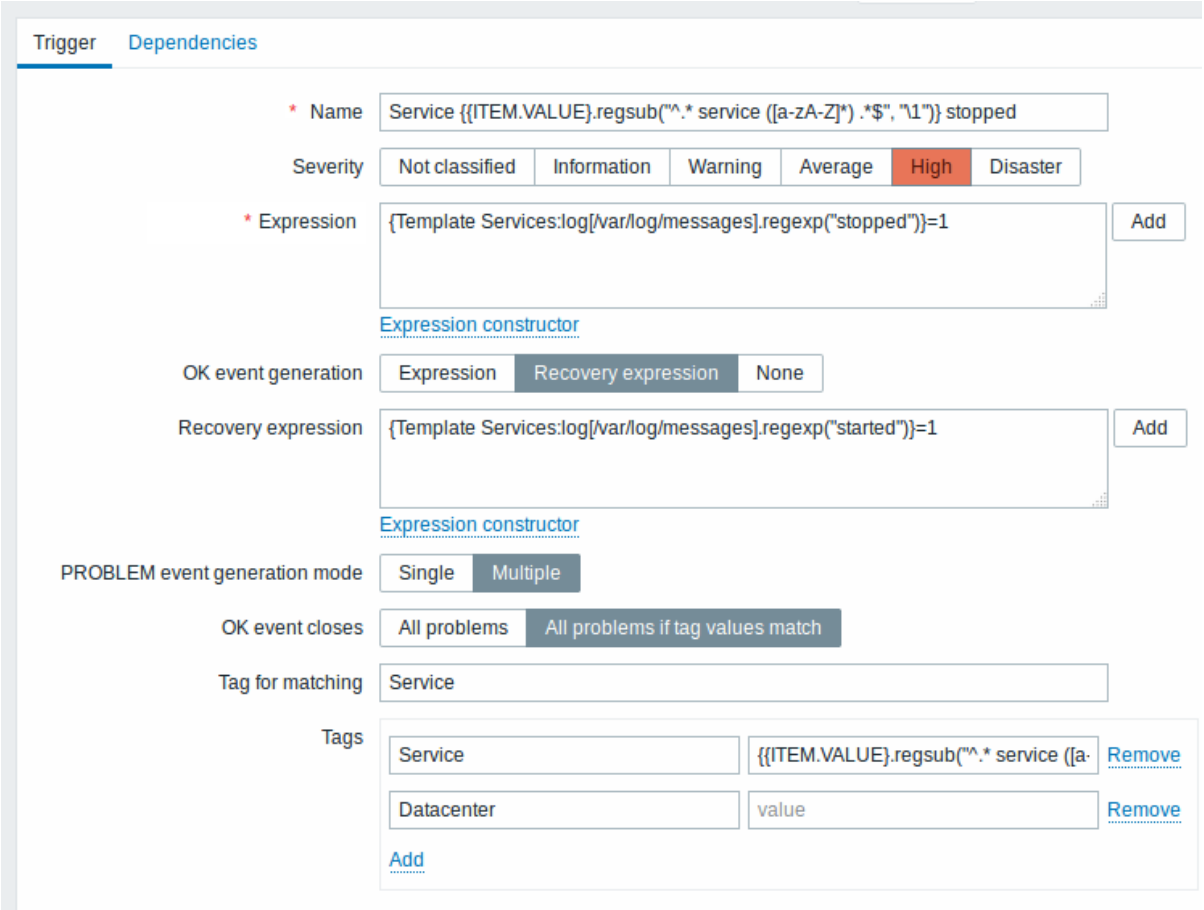
Then define a recovery expression. The following recovery expression will resolve all problems if a log line is found containing the string "Starting":

```
{My host:log[/var/log/syslog].regexp("Starting")}=1
```

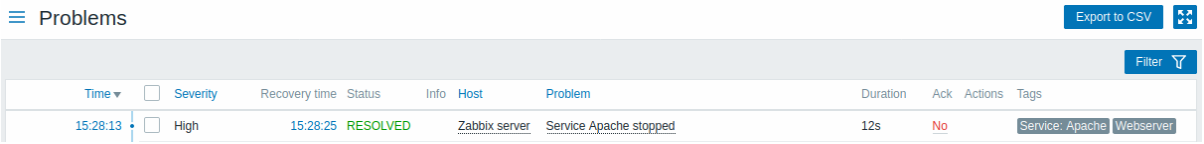
Since we do not want that it's important to make sure somehow that the corresponding root problems are closed, not just all problems. That's where tagging can help.

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
- configure the tags to extract tag values from log lines



If configured successfully you will be able to see problem events tagged by application and matched to their resolution in *Monitoring → Problems*.



Warning:
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.

Event tags

Overview

There is an option to define custom event tags in Zabbix. Tags can be defined on template, host and trigger levels.

After the tags are defined, corresponding new events get marked with tag data:

- with template level tags - host problems that are created by triggers from this template will be marked
- with host level tags - all problems of the host will be marked
- with trigger level tags - problem of this trigger will be marked

An event inherits all tags from the whole chain of templates, hosts, 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. Most importantly, events can be correlated based on event tags. In other uses, actions can be defined based on event tags.

Event tags are realized as a pair of the *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 (trigger, template, host or event) may have several tags with the same name, but different values - these tags will not be considered 'duplicates'. Similarly, a tag with no value and the same tag with a value can be used simultaneously.

Note:
Tags are not supported for host prototypes and hosts created from prototypes.

Use cases

Some use cases for this functionality are as follows:

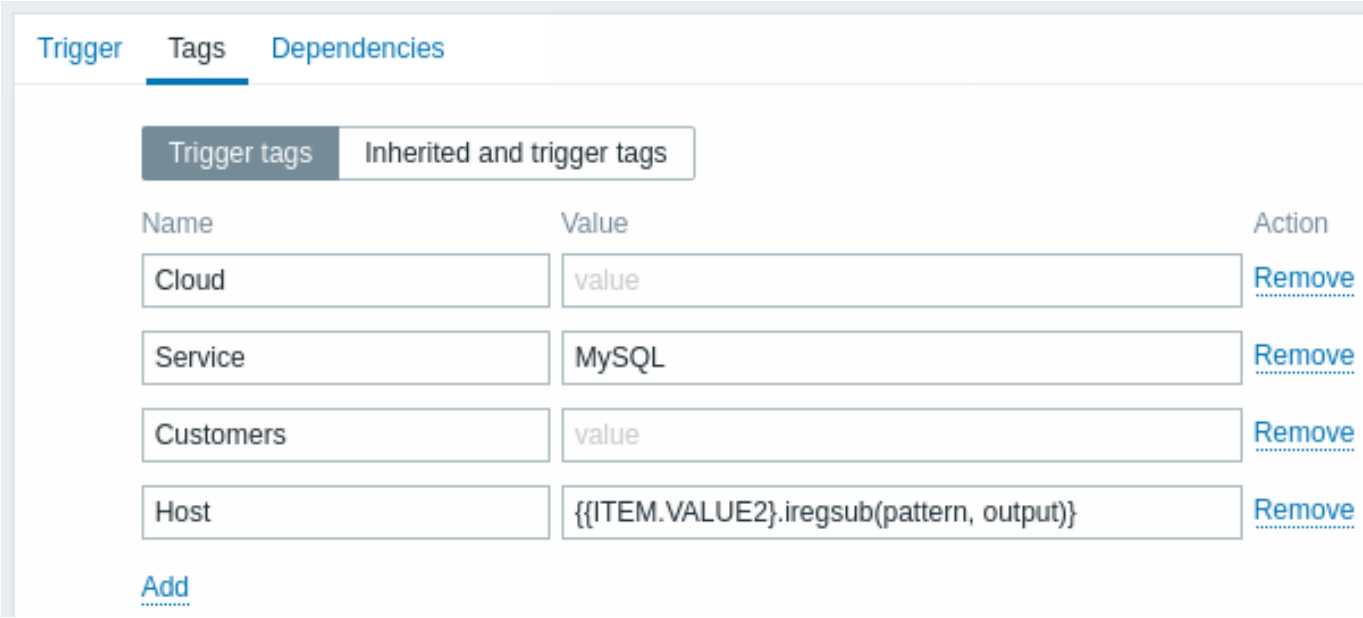
1. Mark trigger events in the frontend

```
   * Define tags on trigger level;
   * See how all trigger problems are marked with these tags in //Monitoring// → //Problems//.
- 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 tag
- 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//
- 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.V
   * In trigger configuration, have multiple problem event generation mode;
   * In trigger configuration, use [[:manual/config/event_correlation|event correlation]]: select the opti
   * See problem events created with a tag and closed individually.
- 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 dat
- 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.
- 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.
- 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.
- 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

Event tags can be defined in:

- template configuration - affecting all triggers from the template when linked to hosts
- host configuration - affecting all triggers of the host
- individual trigger configuration:



Event tags can be defined for triggers, template triggers and trigger prototypes.

Macro support

The following macros may be used in trigger-level 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 (supported since 4.0.0).
- 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.TAGSJSON} and {EVENT.RECOVERY.TAGSJSON} macros will resolve to a JSON array containing event tag objects or recovery event tag objects.

The following macros may be used in template and host-level tags:

- {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros
- {INVENTORY.*} macros
- User macros

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)}
```

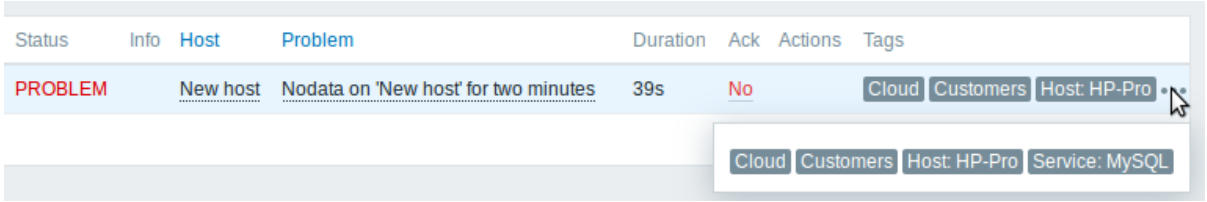
Tag name and value will be cut to 255 characters if their length exceeds 255 characters after macro resolution.

See also: Using macro functions in low-level discovery macros for event tagging.

Viewing event tags

Event tags, if defined, can be seen with new events in:

- *Monitoring → Problems*
- *Monitoring → Problems → Event details*
- *Monitoring → Dashboard → Problems* widget (in popup window that opens when rolling the mouse over problem name|



Only the first three tag entries are displayed. If there are more than three tag entries, it is indicated by three dots. If you roll your mouse over these three dots, all tag entries are displayed in a pop-up window.

Note that the order in which tags are displayed is affected by tag filtering and the *Tag display priority* option in the filter of *Monitoring → Problems* or the *Problems* dashboard widget.

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 Zabbix Super Admin level users only.

> **Attention:**
> Event correlation must be configured very carefully, as it can negatively affect event processing performance or, if mis-configured, 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:

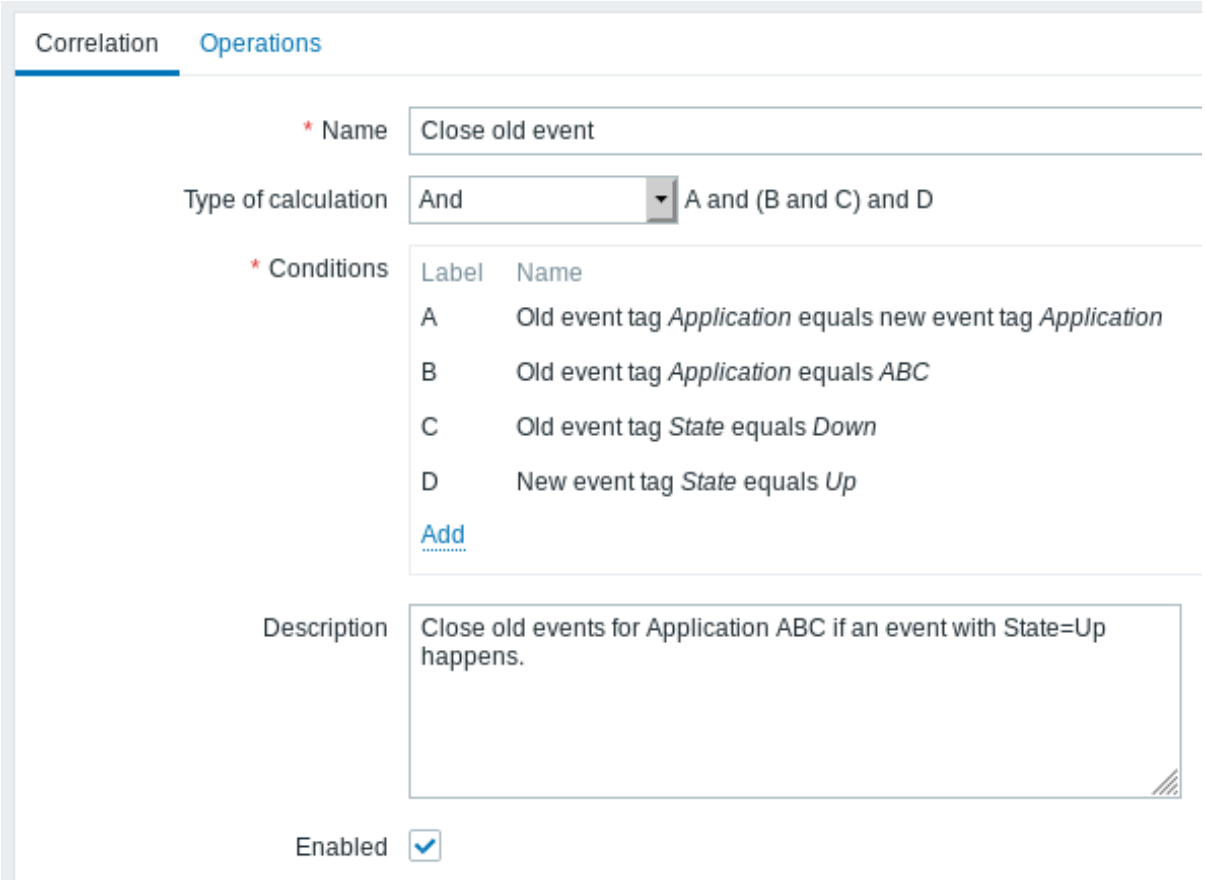
- Reduce the correlation scope. Always set a unique tag for the new event that is paired with old events and use the *New event tag* correlation condition;
- Add a condition based on the old event when using the *Close old event* operation (or else all existing problems could be closed);
- Avoid using 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.

Configuration

To configure event correlation rules globally:

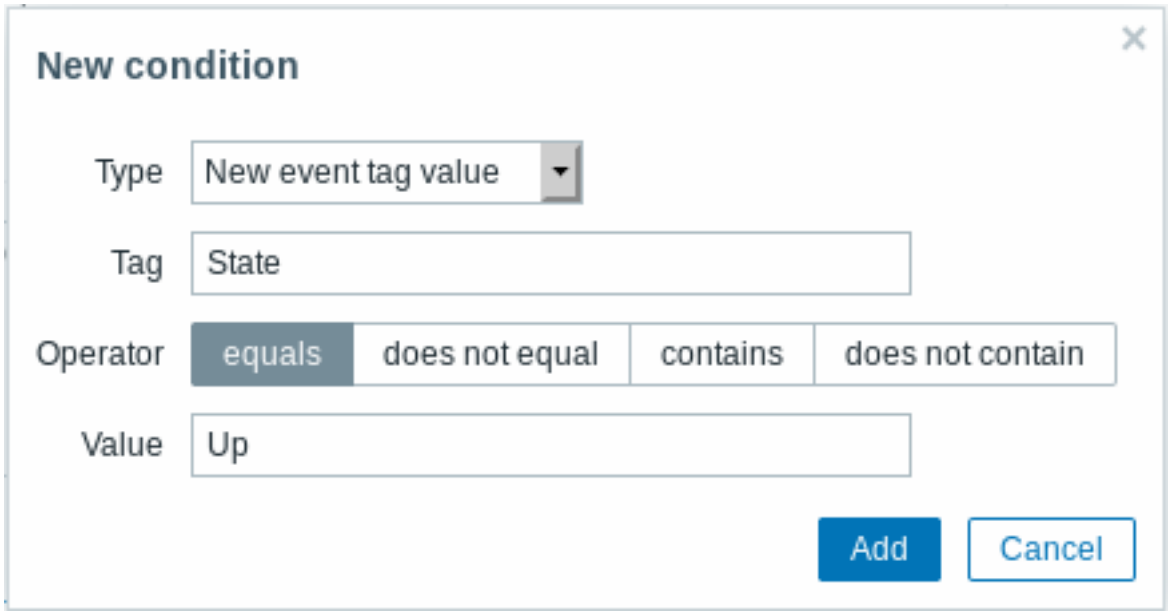
- Go to *Configuration → Event correlation*
- Click on *Create correlation* to the right (or on the correlation name to edit an existing rule)
- Enter parameters of the correlation rule in the form



All mandatory input fields are marked with a red asterisk.

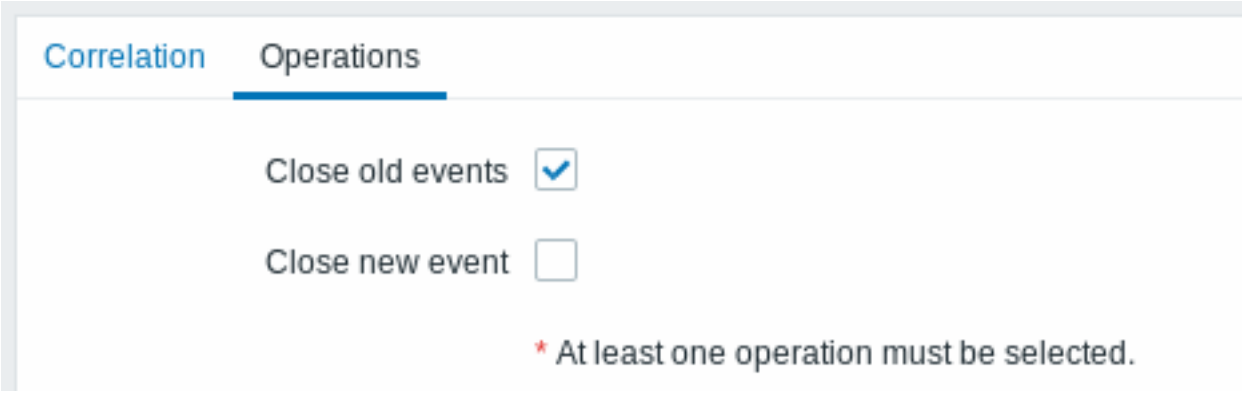
Parameter	Description
Name	Unique correlation rule name.
Type of calculation	The following options of calculating conditions are available: **And** - all conditions must be met **Or** - enough if one condition is met **And/Or** - AND with different condition types and OR with the same condition type **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).
Conditions	List of conditions. See below for details on configuring a condition.
Description	Correlation rule description.
Enabled	If you mark this checkbox, the correlation rule will be enabled.

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.



Parameter	Description
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

- Select the operation of the correlation rule in the form



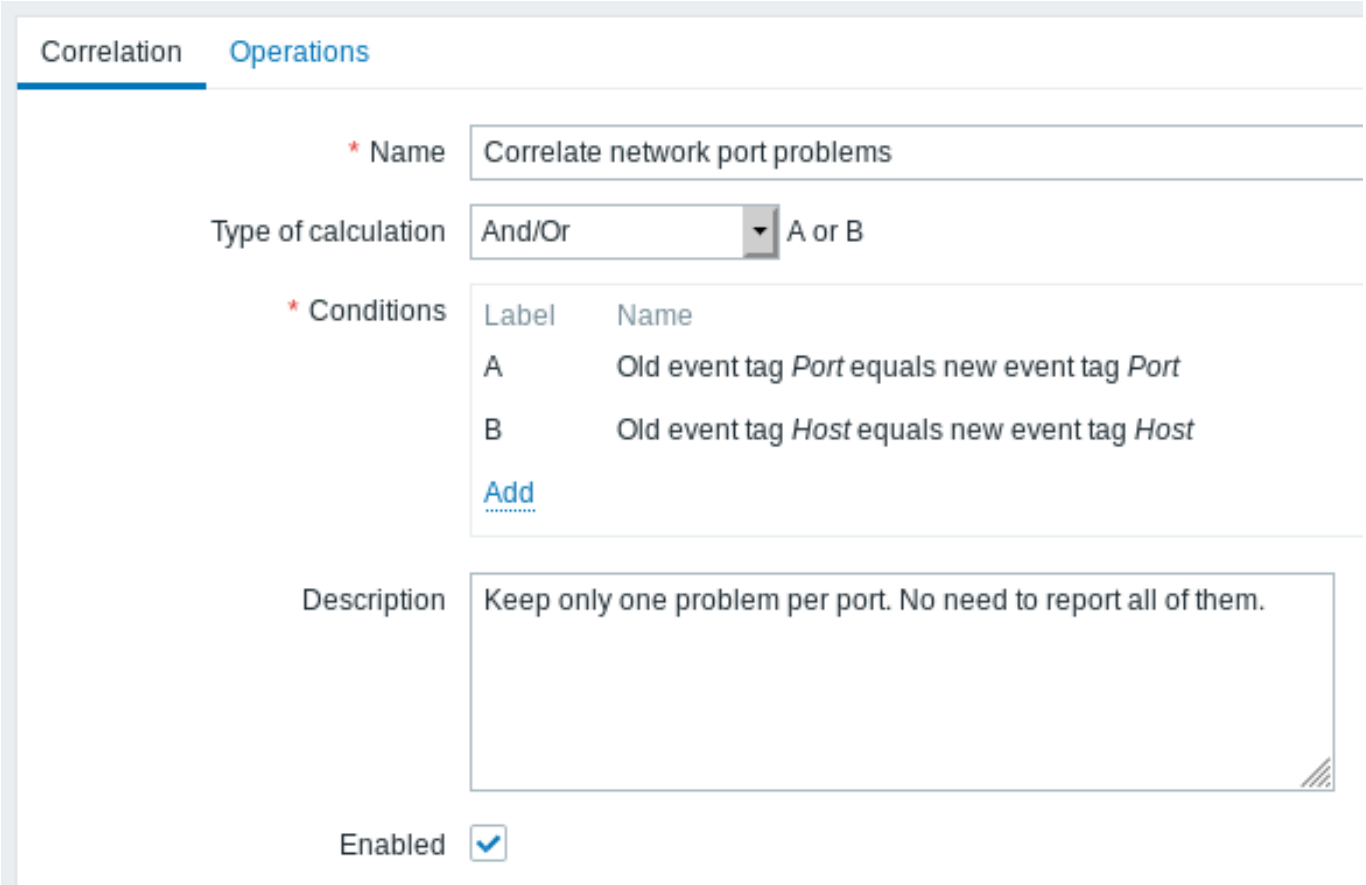
Parameter	Description
Operations *New operation*	List of operations, selected from the *New operation* field. Select operation to perform when event is correlated and click on *Add*. The following operations are available: **Close old events** - close old events when a new event happens. Always add a 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

Warning:
Because misconfiguration is possible, when similar event tags may be created for **unrelated** problems, please review the cases outlined below!

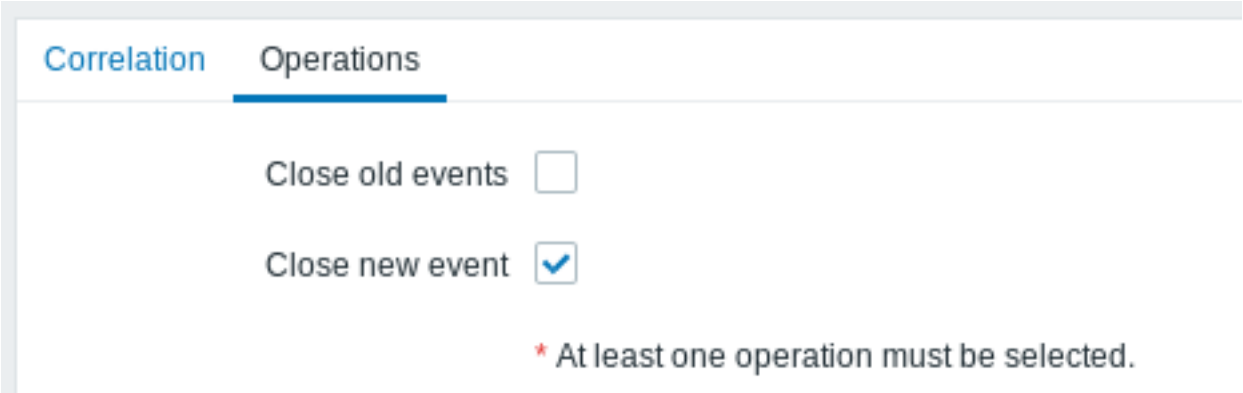
- Actual tags and tag values only become visible when a trigger fires. If the regular expression used is invalid, it is silently replaced with an *UNKNOWN* string. If the initial problem event with an *UNKNOWN* tag value is missed, there may appear subsequent OK events with the same *UNKNOWN* tag value 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.

Example

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.



This operation will close new problem events on the same network port, keeping only the original problem open.

6 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 customisable vector graphs

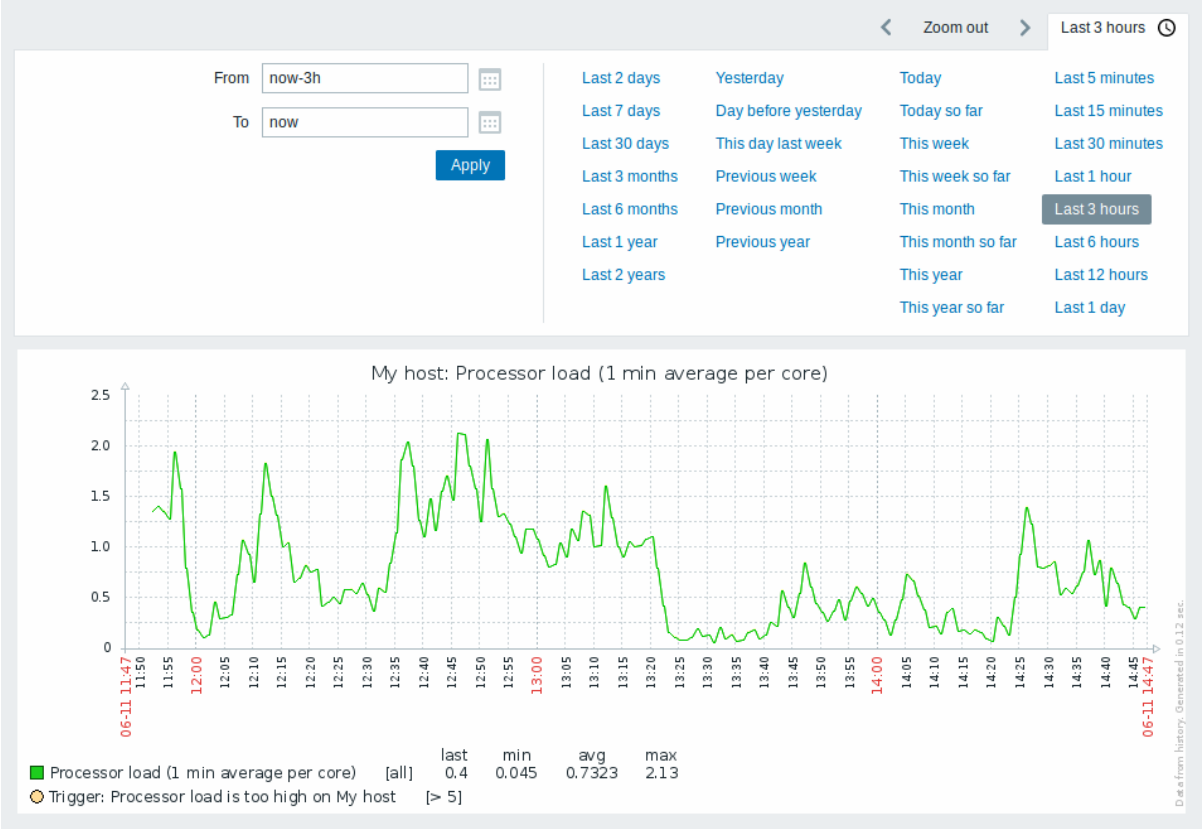
1 Simple graphs

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.



Note:
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:

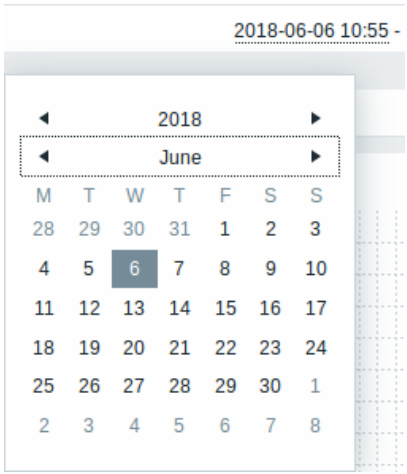
Precision	From	To
m	Y-m-d H:m:00	Y-m-d H:m:59
h	Y-m-d H:00:00	Y-m-d H:59:59
d	Y-m-d 00:00:00	Y-m-d 23:59:59
w	Monday of the week 00:00:00	Sunday of the week 23:59:59
M	First day of the month 00:00:00	Last day of the month 23:59:59
y	1st of January of the year 00:00:00	31st of December of the year 23:59:59

For example:

From	To	Selected period
`now/d`	`now/d`	00:00 - 23:59 today
`now/d`	`now/d+1d`	00:00 today - 23:59 tomorrow
`now/w`	`now/w`	Monday 00:00:00 - Sunday 23:59:59 this week
`now-1y/w`	`now-1y/w`	The week of Monday 00:00:00 - Sunday 23:59:59 one year ago

Date picker

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.

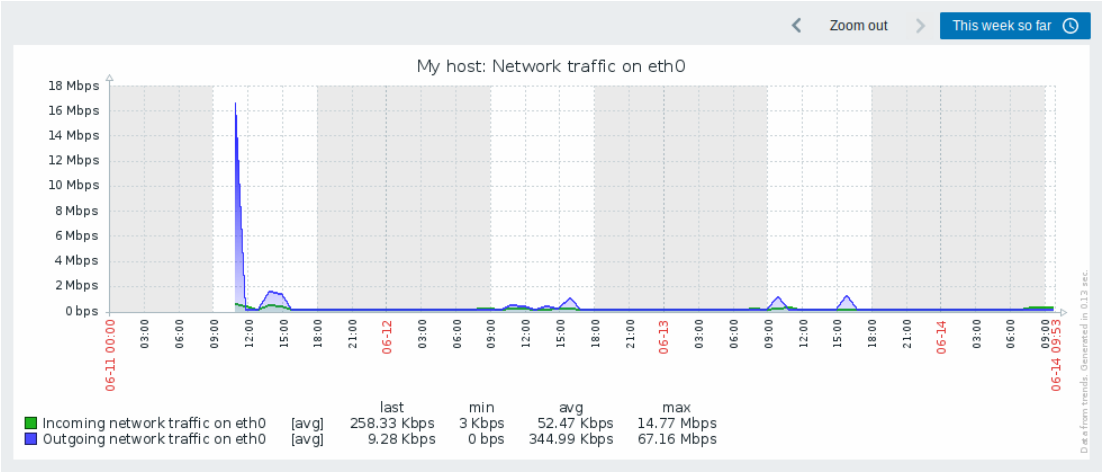
In case no time value is specified or field is left blank, time value will be set to "00:00:00". This doesn't apply to today's date selection: in that case time will be set to current value.

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.

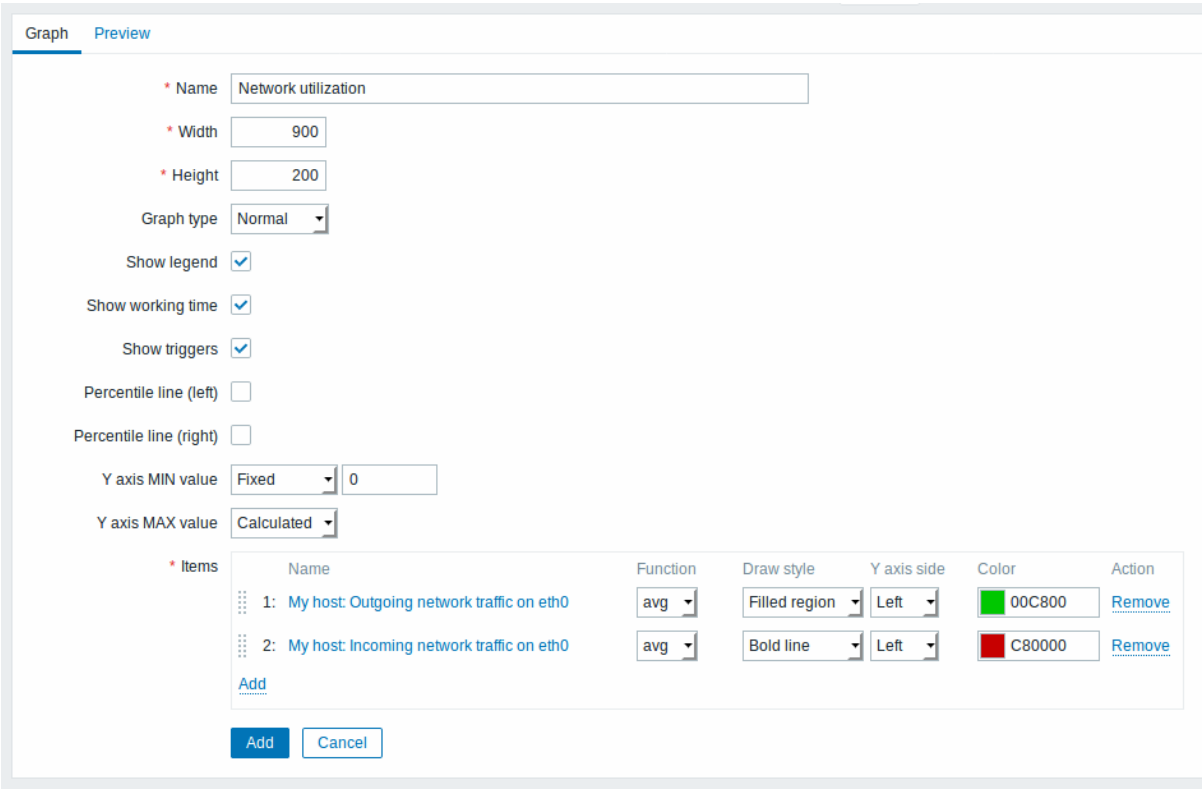
Custom graphs are configured manually.

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.

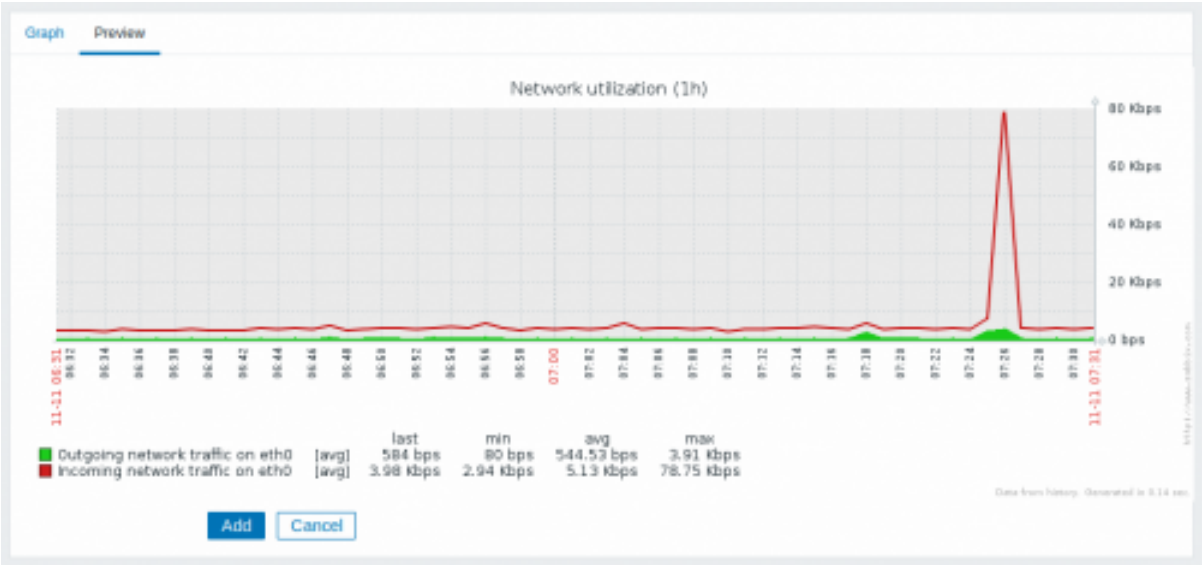
Graph attributes:

Parameter		Description
Name		Unique graph name. Starting with Zabbix 2.2, item values can be referenced in the name by using simple macros with the standard `{host:key.func(param)}` syntax. Only **avg**, **last**, **max** and **min** as functions with seconds as parameter are supported within this macro. {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 `{{HOST.HOST1}:key.func(param)}`.
Width		Graph width in pixels (for preview and pie/exploded graphs only).
Height		Graph height in pixels.
Graph type		Graph type: **Normal** - normal graph, values displayed as lines **Stacked** - stacked graph, filled areas displayed **Pie** - pie graph **Exploded** - ”exploded” pie graph, portions displayed as ”cut out” of the pie
Show legend		Checking this box will set to display the graph legend.
Show working time		If selected, non-working hours will be shown with gray background. Not available for pie and exploded pie graphs.
Show triggers		If selected, simple triggers will be displayed as lines with black dashes over trigger severity color. Not available for pie and exploded pie graphs.
Percentile line (left)		Display percentile for left Y axis. If, for example, 95% percentile is set, then the percentile line will be at the level where 95 per cent of the values fall under. Displayed as a bright green line. Only available for normal graphs.
Percentile line (right)		Display percentile for right Y axis. If, for example, 95% percentile is set, then the percentile line will be at the level where 95 per cent of the values fall under. Displayed as a bright red line. Only available for normal graphs.
Y axis MIN value		Minimum value of Y axis: **Calculated** - Y axis minimum value will be automatically calculated **Fixed** - 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
Y axis MAX value		Maximum value of Y axis: **Calculated** - Y axis maximum value will be automatically calculated **Fixed** - 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
3D view		Enable 3D style. For pie and exploded pie graphs only.
Items		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).
	Sort order (0→100)	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 in the beginning of line to set the sort order or which item is displayed in front of the other.
	Name	Name of the selected item is displayed as a link. Clicking on the link opens the list of other available items.

Parameter		Description
	Type	Type (only available for pie and exploded pie graphs): **Simple** - value of the item is represented proportionally on the pie **Graph sum** - 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.
	Function	Select what values will be displayed when more than one value exists per vertical graph pixel for an item: **all** - 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. **avg** - display the average values **last** - display the latest values. This function is only available if either *Pie/Exploded pie* is selected as graph type. **max** - display the maximum values **min** - display the minimum values
	Draw style	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*.
	Y axis side	Select the Y axis side to show the item data - *Left*, *Right*.
	Color	Select the color to apply to the item data.

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.

Note:
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

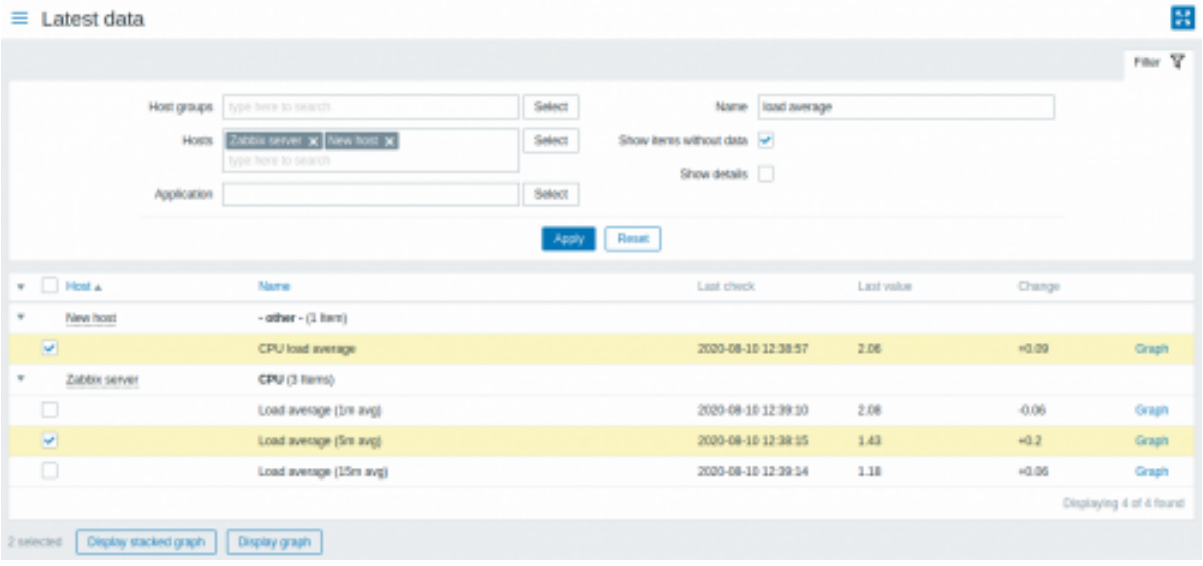
While a simple graph is great for accessing data of one item and custom graphs offer customization options, none of the two allow to quickly create a comparison graph for multiple items with little effort and no maintenance.

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



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

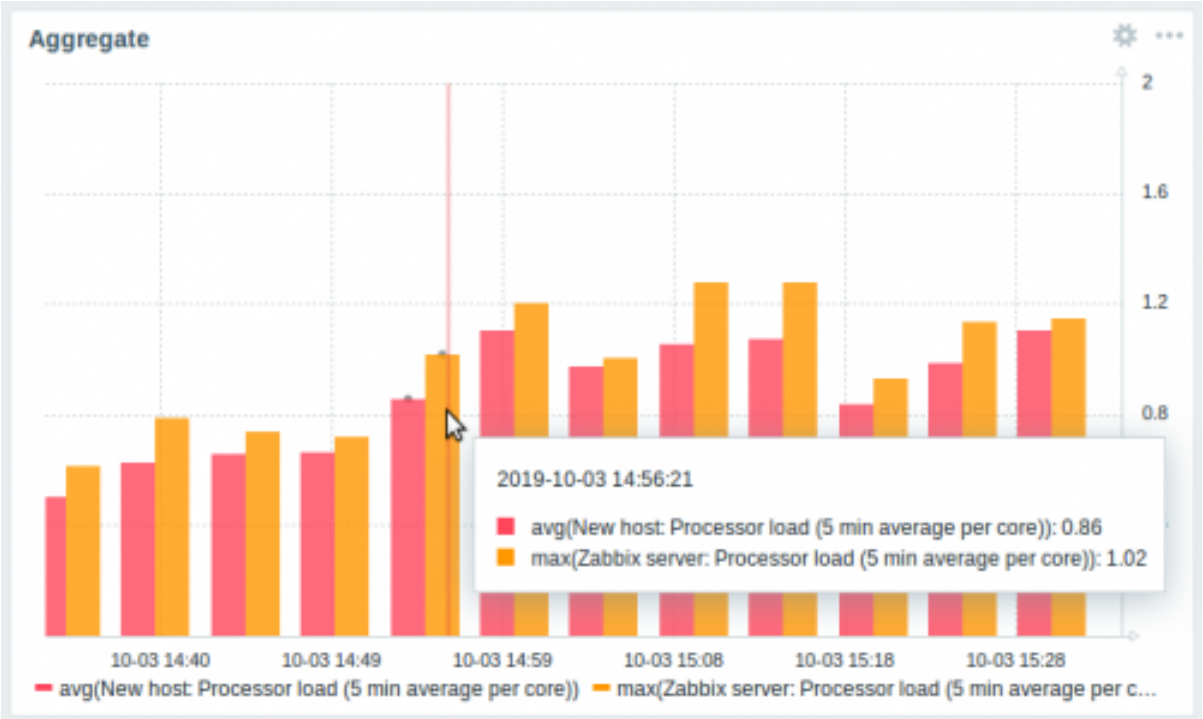
Overview

The aggregation functions, available in the graph widget of the dashboard, allow to display 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)

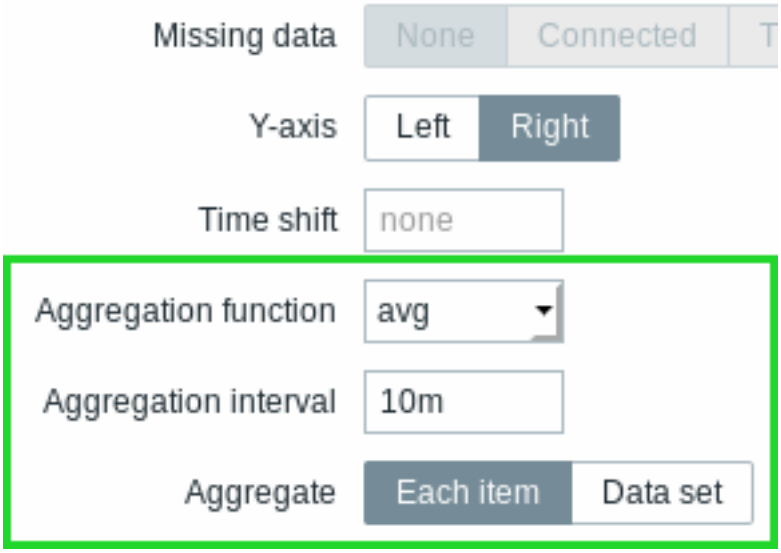
The most exciting use of data aggregation is the possibility to create nice side-by-side comparisons of data for some period:



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 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.

See the widget configuration page for more details.

Use cases

Average request count to Nginx server

View the average request count per second per day to 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 member 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, 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 the element.

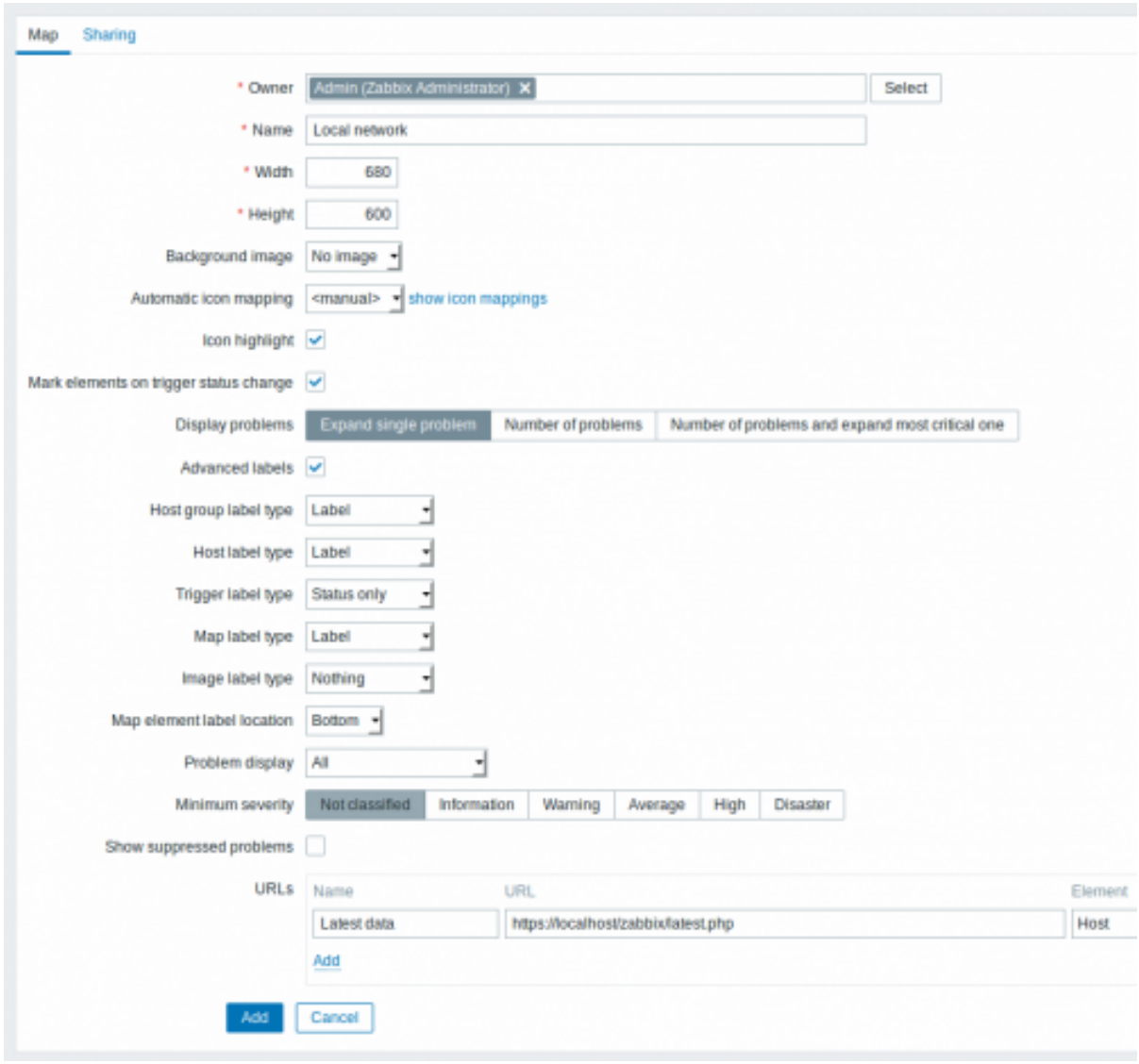
Creating a map

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.

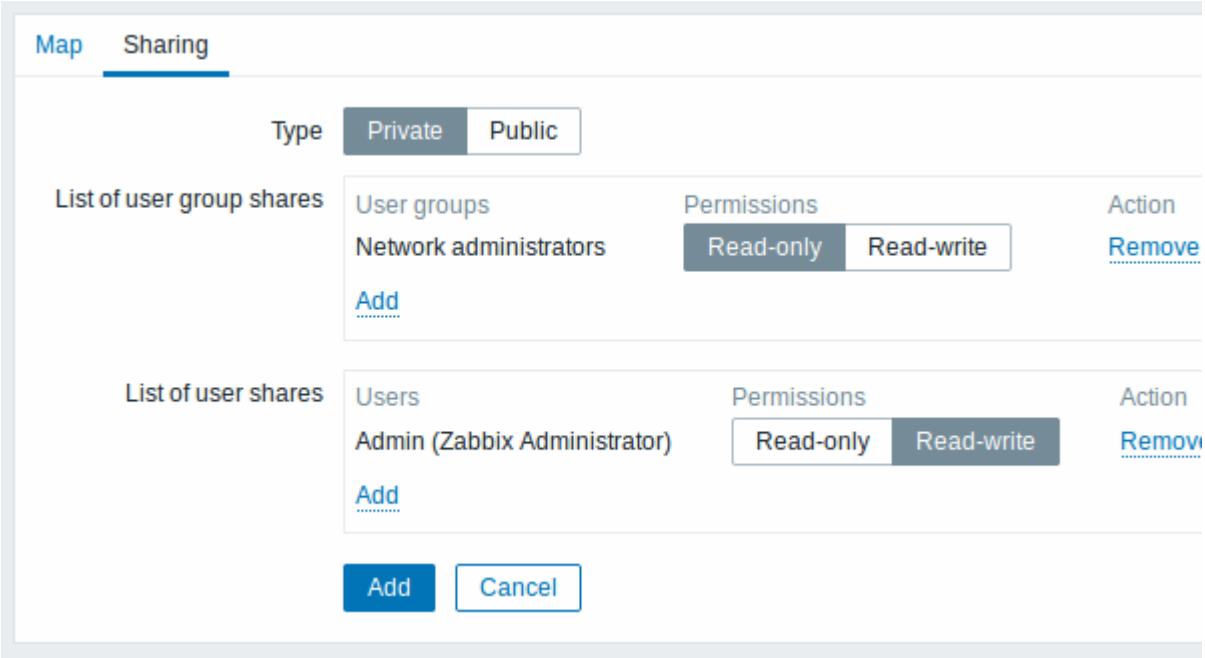
General map attributes:

Parameter	Description
Owner	Name of map owner.
Name	Unique map name.
Width	Map width in pixels.
Height	Map height in pixels.
Background image	Use background image: **No image** - no background image (white background) **Image** - 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.
Automatic icon mapping	You can set to use an automatic icon mapping, configured in *Administration → General → Icon mapping*. Icon mapping allows to map certain icons against certain host inventory fields.
Icon highlighting	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. See also: Viewing maps

Parameter	Description
Mark elements on trigger status change	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.
Display problems	Select how problems are displayed with a map element: **Expand single problem** - if there is only one problem, the problem name is displayed. Otherwise, the total number of problems is displayed. **Number of problems** - the total number of problems is displayed **Number of problems and expand most critical one** - name of the most critical problem and the total number of problems is displayed. '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.
Advanced labels	If you check this box you will be able to define separate label types for separate element types.
Map element label type	Label type used for map elements: **Label** - map element label **IP address** - IP address **Element name** - element name (for example, host name) **Status only** - status only (OK or PROBLEM) **Nothing** - no labels are displayed
Map element label location	Label location in relation to the map element: **Bottom** - beneath the map element **Left** - to the left **Right** - to the right **Top** - above the map element
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
Minimum trigger severity	Problems below the selected minimum severity level will not be displayed in the map. For example, with *Warning* selected, changes with *Information* and *Not classified* level triggers will not be reflected in the map. This parameter is supported starting with Zabbix 2.2.
Show suppressed problems	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.
URLs	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. 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'.

Sharing

The **Sharing** tab contains the map type as well as sharing options (user groups, users) for private maps:



Parameter	Description
Type	Select map type: **Private** - map is visible only to selected user groups and users **Public** - map is visible to all
List of user group shares	Select user groups that the map is accessible to. You may allow read-only or read-write access.
List of user shares	Select users that the map is accessible to. You may allow read-only or read-write access.

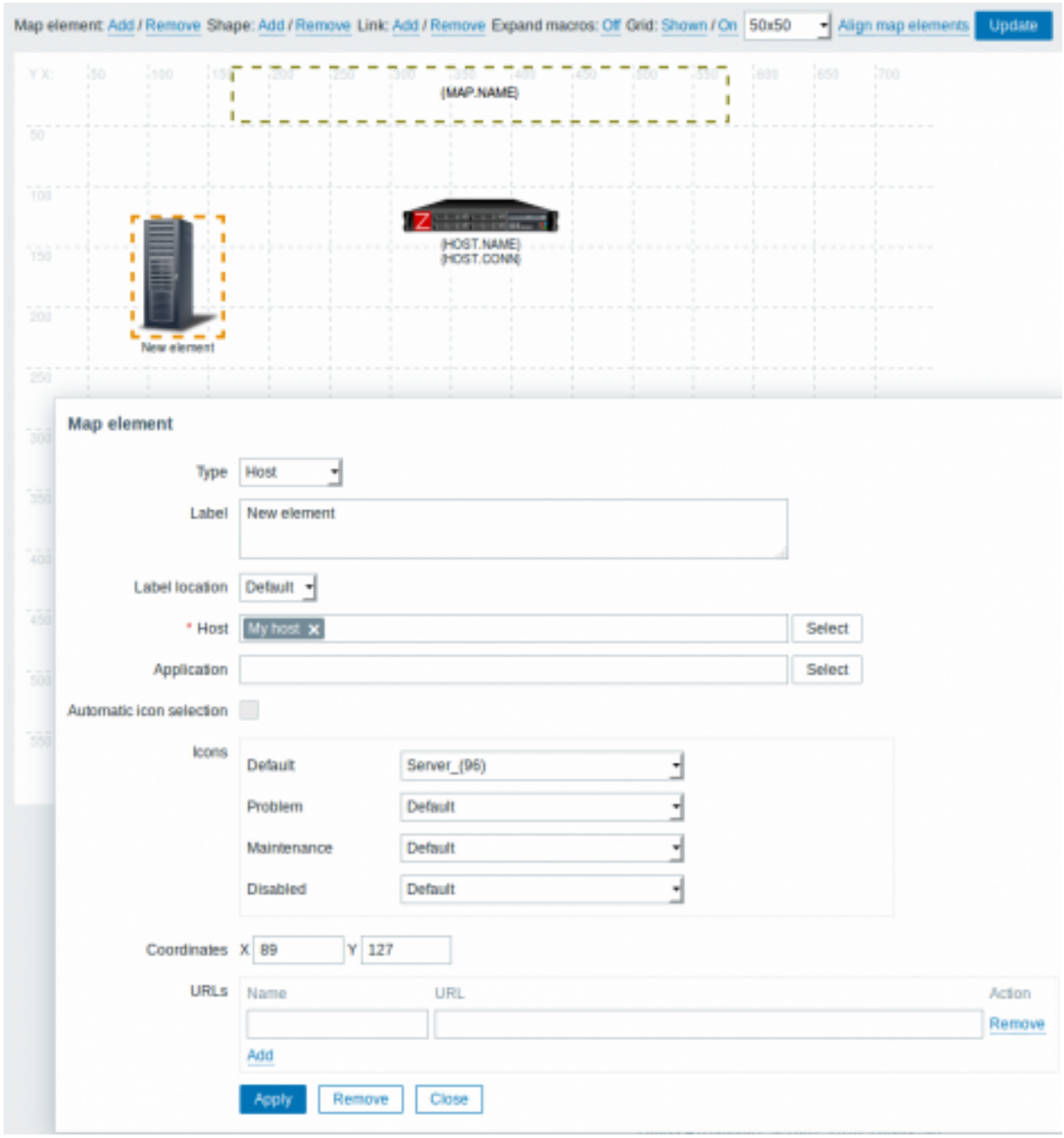
When you click on *Add* to save this map, you have created an empty map with a name, dimensions and certain preferences. Now you need to add some elements. For that, click on *Constructor* in the map list to open the editable area.

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:

Parameter	Description
Type	Type of the element: **Host** - icon representing status of all triggers of the selected host **Map** - icon representing status of all elements of a map **Trigger** - icon representing status of one or more triggers **Host group** - icon representing status of all triggers of all hosts belonging to the selected group **Image** - an icon, not linked to any resource
Label	Icon label, any string. Macros and multiline strings can be used in labels. For a full list of supported macros, see supported macros and search for 'map element labels'.
Label location	Label location in relation to the icon: **Default** - map's default label location **Bottom** - beneath the icon **Left** - to the left **Right** - to the right **Top** - above the icon

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.
Application	You can select an application, allowing to only display problems of triggers that belong to the given application. This field is available for host and host group element types, and supported since Zabbix **2.4.0**.
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'.

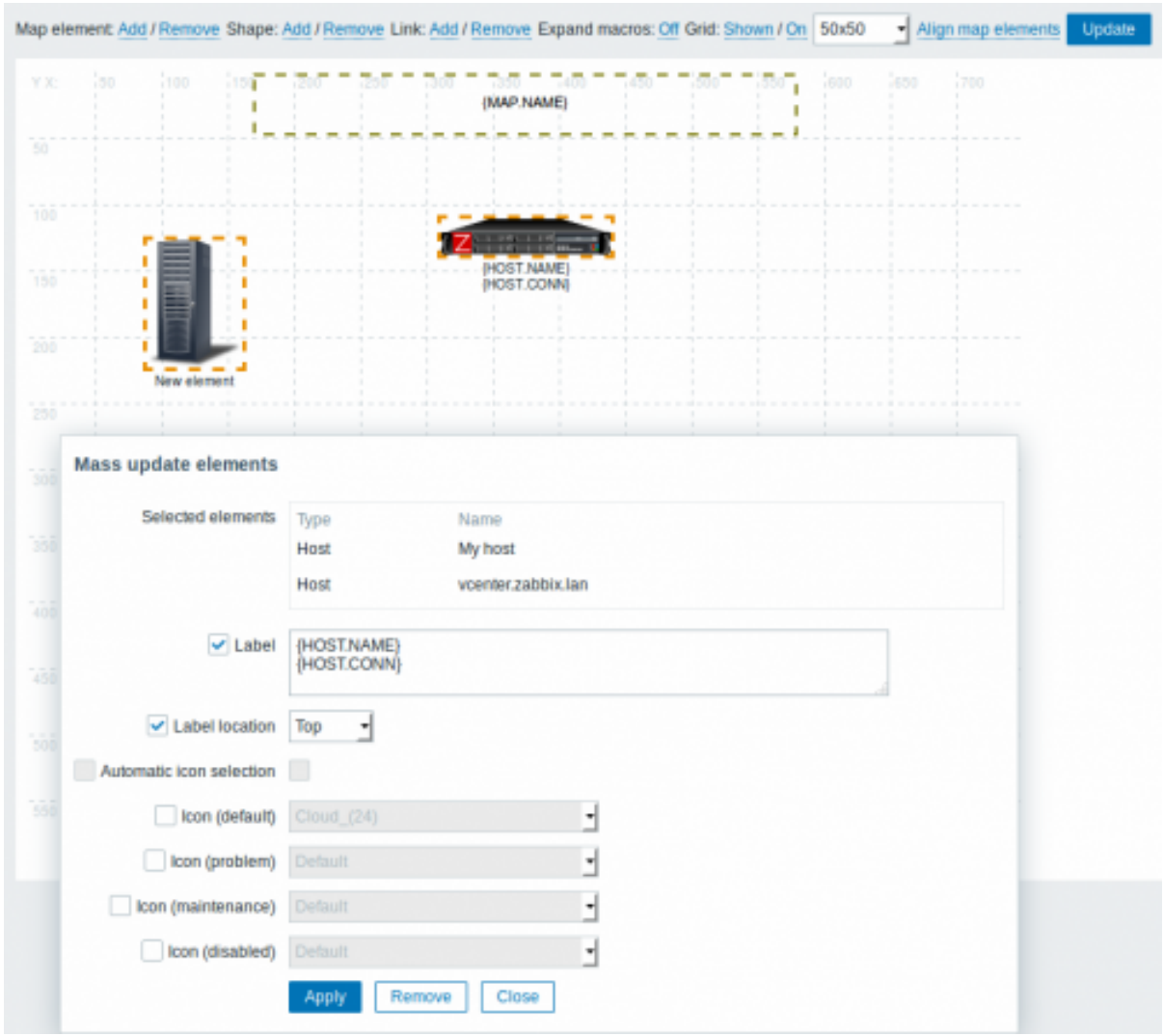
Attention:
Added elements are not automatically saved. If you navigate away from the page, all changes may be lost.
Therefore it is a good idea to click on the **Update** button in the top right corner. Once clicked, the changes are saved regardless of what you choose in the following popup.
Selected 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.

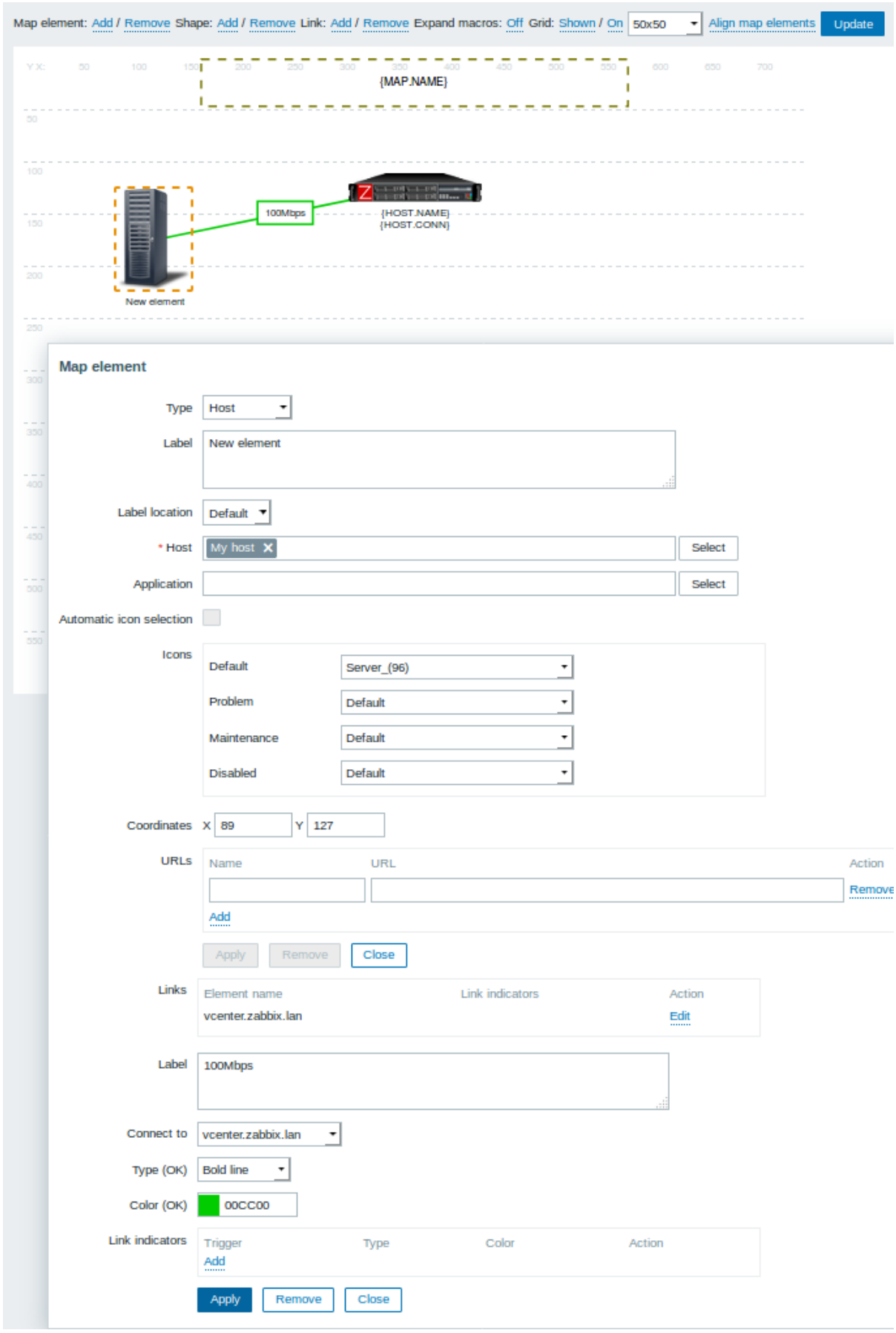
Once you select more than one element, the element property form shifts to the mass-update mode so you can change attributes of selected elements in one go. To do so, mark the attribute using the checkbox and enter a new value for it. You may use macros here (such as, say, {HOST.NAME} for the element label).



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.



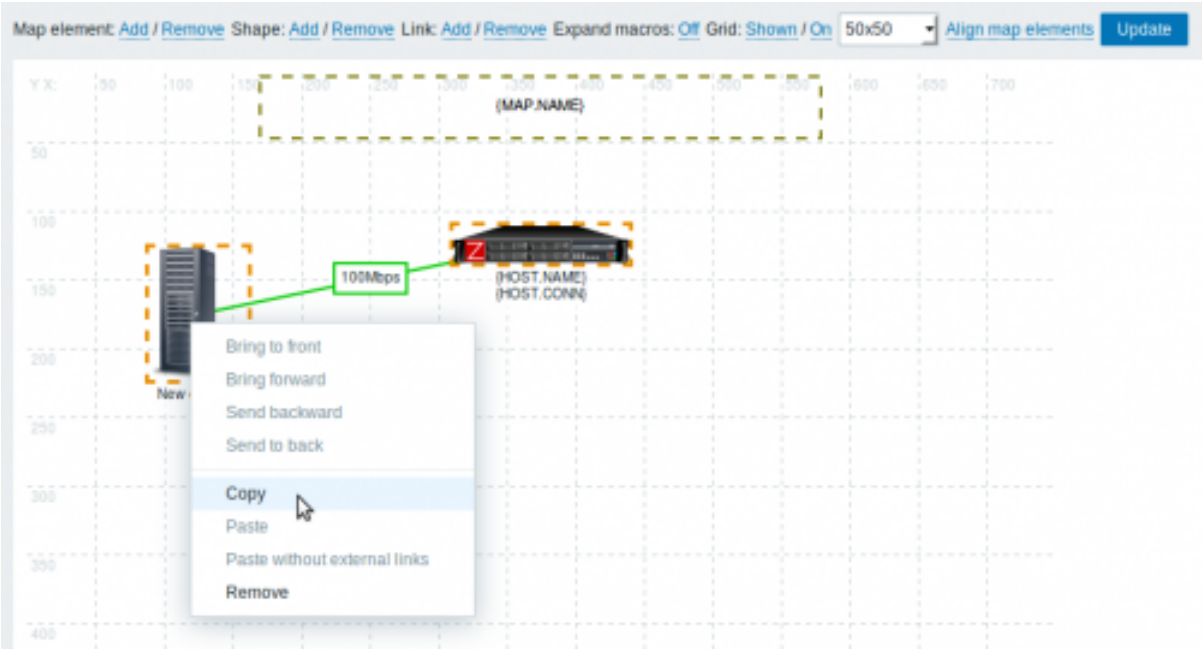
Link attributes:

Parameter	Description
Label	Label that will be rendered on top of the link. The {host:key.func(param)} macro is supported in this field, but only with `avg`, `last`, `min` and `max` trigger functions, with seconds as parameter.
Connect to	The element that the link connects to.
Type (OK)	Default link style: **Line** - single line **Bold line** - bold line **Dot** - dots **Dashed line** - dashed line
Color (OK)	Default link color.
Link indicators	List of triggers linked to the link. In case a trigger has status PROBLEM, its style is applied to the link.

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.

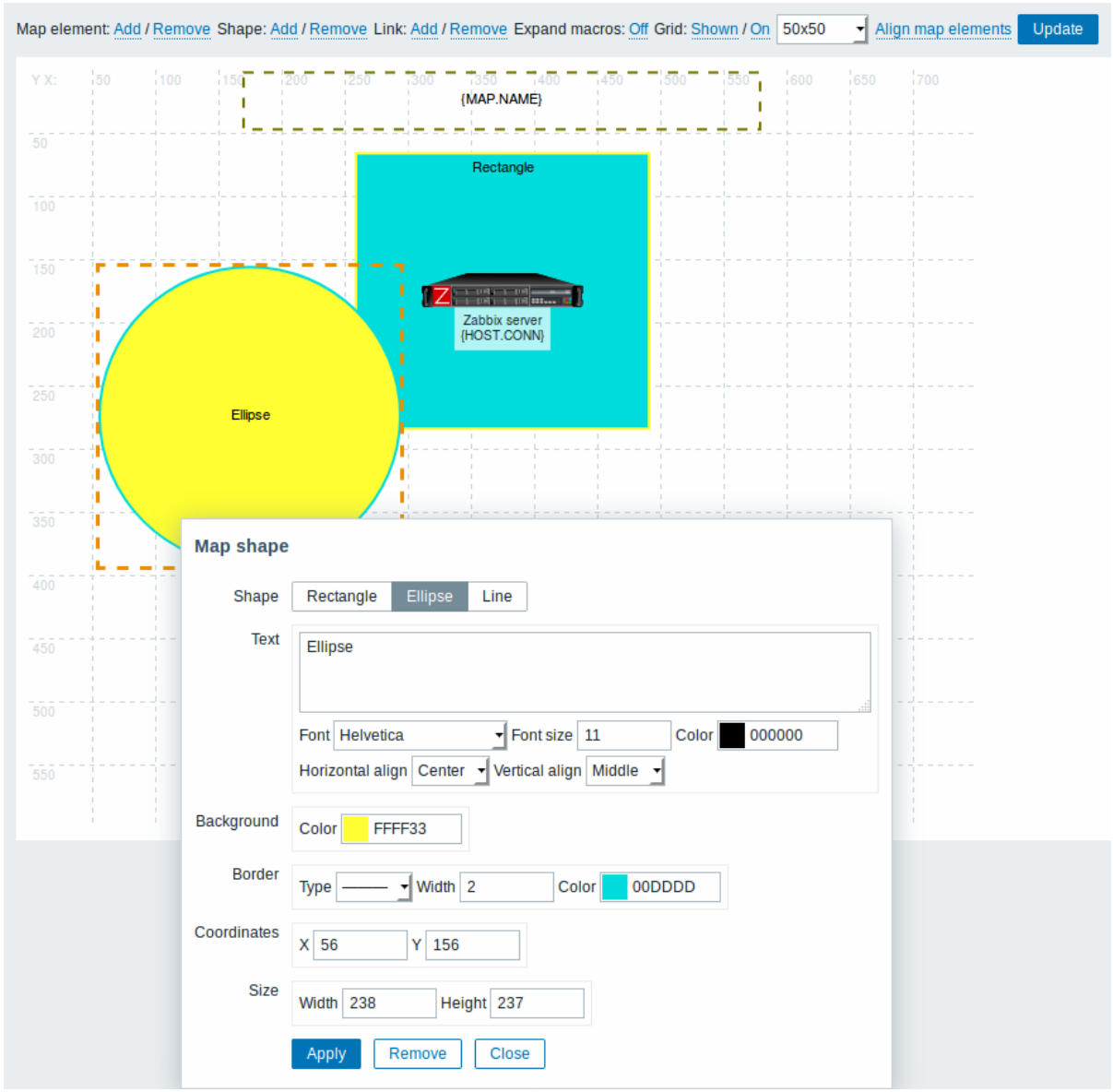
Copy-pasting works within the same browser window. Keyboard shortcuts are not supported.

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.

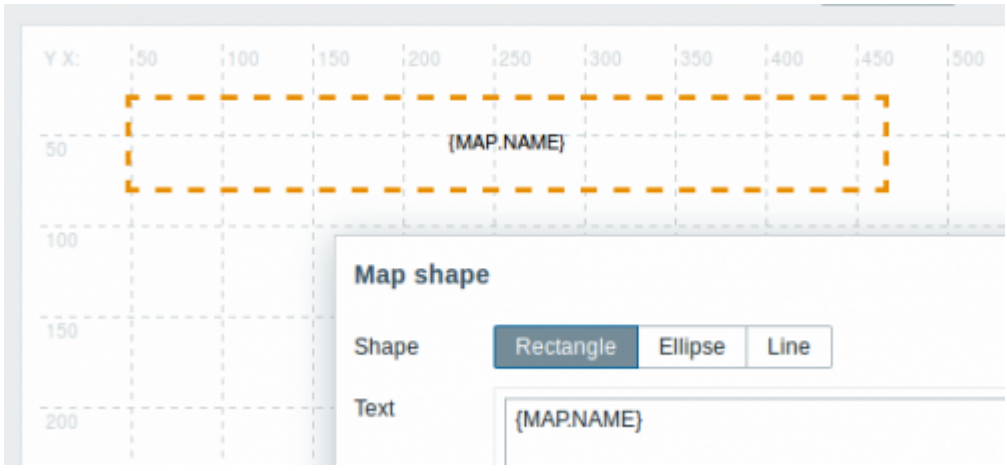
To add a shape, click on *Add* next to Shape. The new shape will appear at the top left corner of the map. Drag and drop it wherever you like.

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. 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 {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.

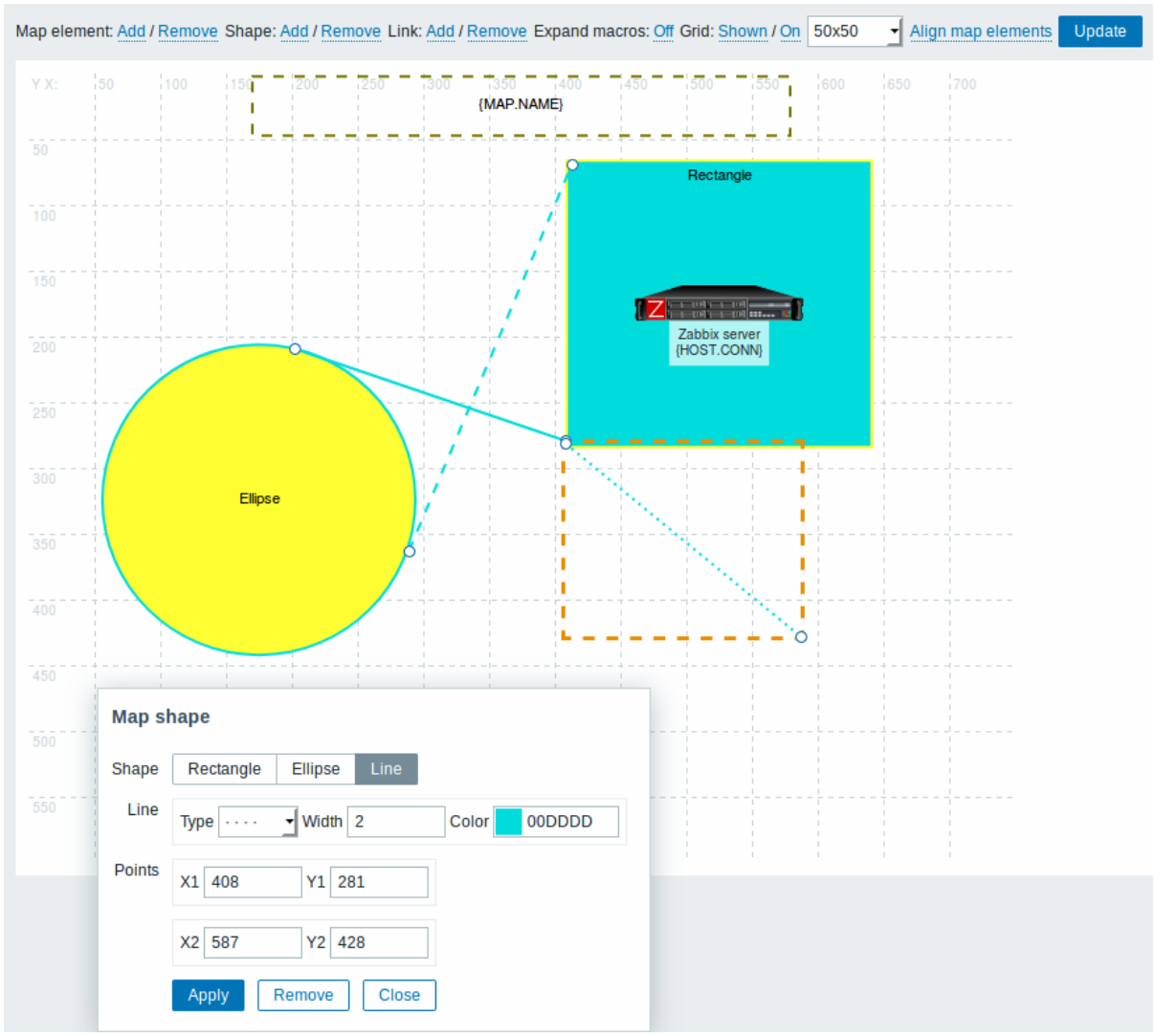
Line wrapping for text is always "on" within shapes. However, within an ellipse the lines are wrapped as though the ellipse were

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

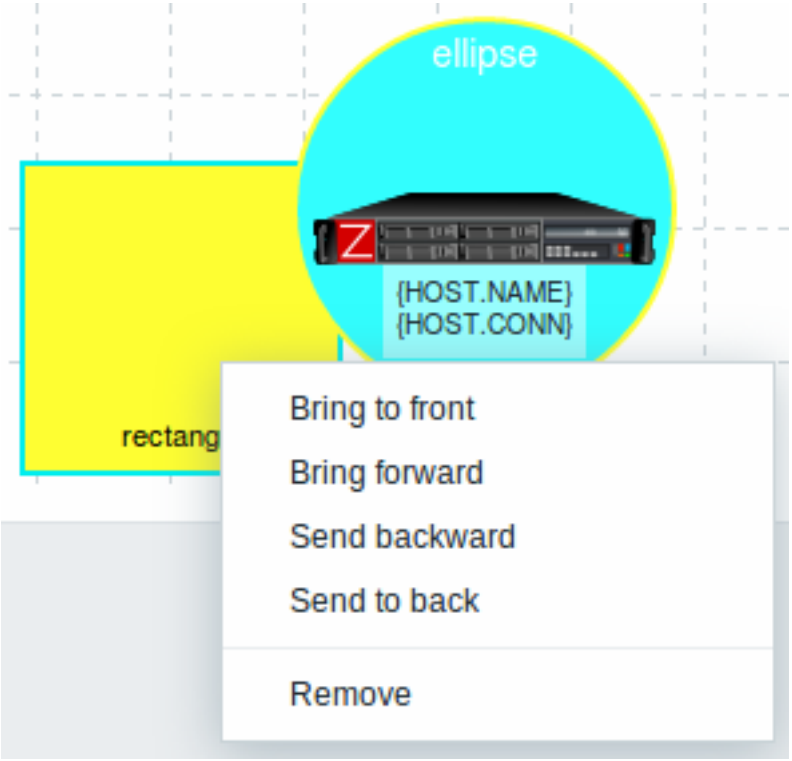
In addition to shapes, it is also possible to add some lines. Lines can be used to link elements or shapes in a map.

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.

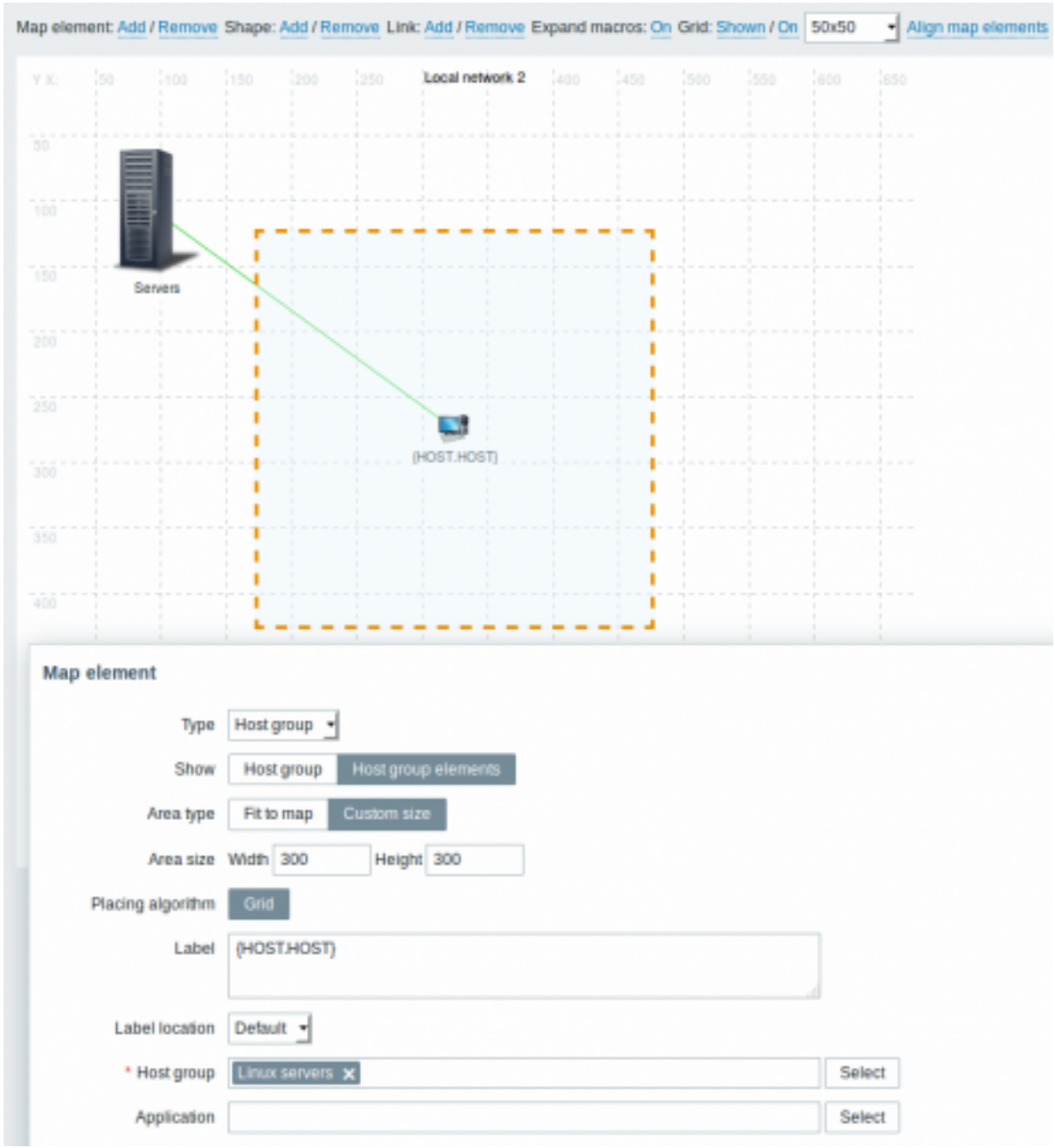


2 Host group elements

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.

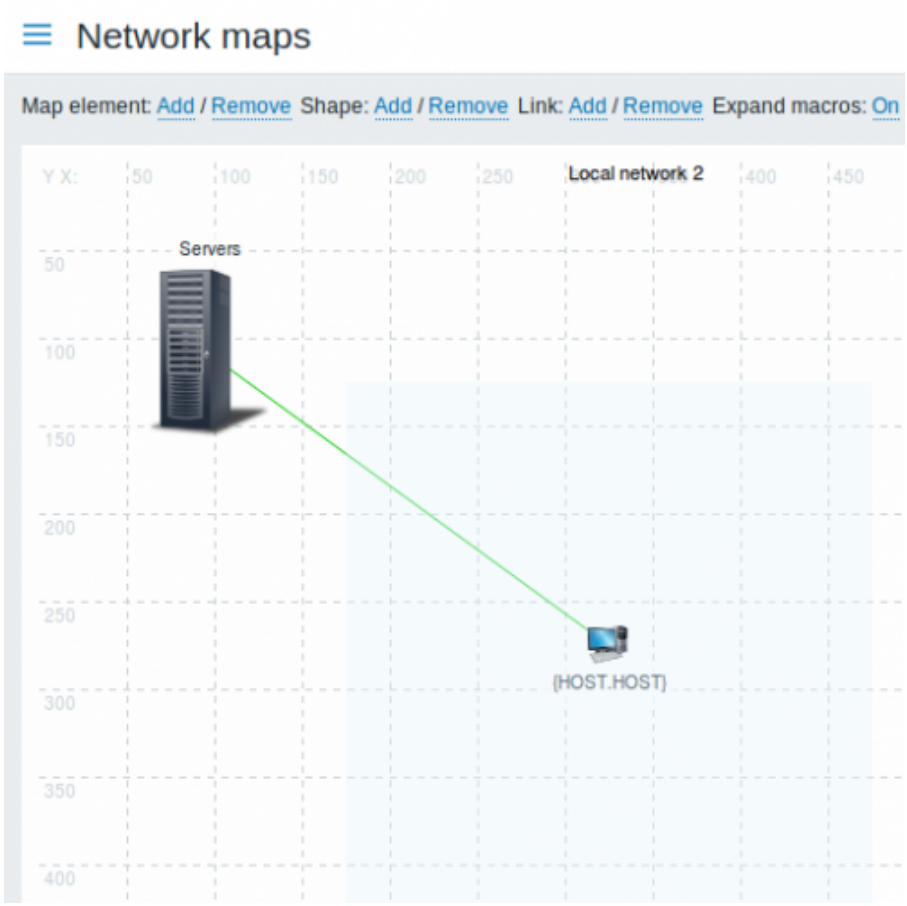
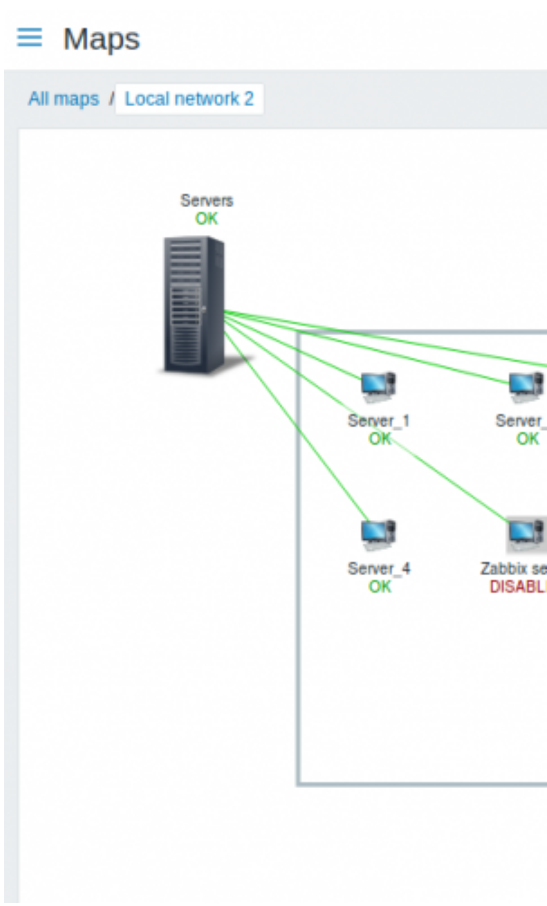
This table consists of parameters typical for *Host group* element type:

Parameter	Description
Type	Select Type of the element: **Host group** - icon representing status of all triggers of all hosts belonging to the selected group
Show	Show options: **Host group** - selecting this option will result as one single icon displaying corresponding information about the certain host group **Host group elements** - selecting this option will result as multiple icons displaying corresponding information about each single element (host) of the certain host group
Area type	This setting is available if “Host group elements” parameter is selected: **Fit to map** - all host group elements are equally placed within the map **Custom size** - manual setting of the map area for all the host group elements to be displayed

Parameter	Description
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 multiline strings can be used in labels. If the type of the map element is “Host group” specifying certain macros has impact on the map view displaying corresponding information about each single host. For example, if {HOST.IP} macro is used, 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 ”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:

Map editing view	Map view
	

Notice how the {HOST.NAME} macro is used. In map editing the macro name in 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.

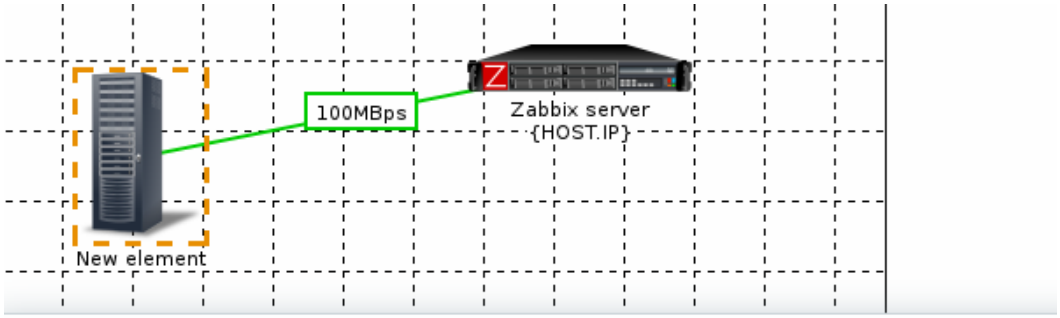
When you configure a link, you set the default link type and color. When you assign triggers to a link, you can assign different link types and colors with these triggers.

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 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



Map element

Type: Host

Label: New element

Label location: Default

*Host: New host × Select

Application: Select

Automatic icon selection

Icons

Default	Server_(96)
Problem	Server_(128)
Maintenance	Server_(24)
Disabled	Default

Coordinates: X 89 Y 127

URLs

NAME	URL

Add

Apply Remove Close

Links

ELEMENT NAME	LINK INDICATORS
Zabbix server	New host: Zabbix agent on New host is unreachable for 5 minutes

Label: 100MBps

Connect to: Zabbix server

Type (OK): Bold line

Color (OK): 00CC00

Link indicators

TRIGGER	TYPE	COLOU
New host: Zabbix agent on New host is unreachable for 5 minutes	Line	DI

Add

Apply Remove Close

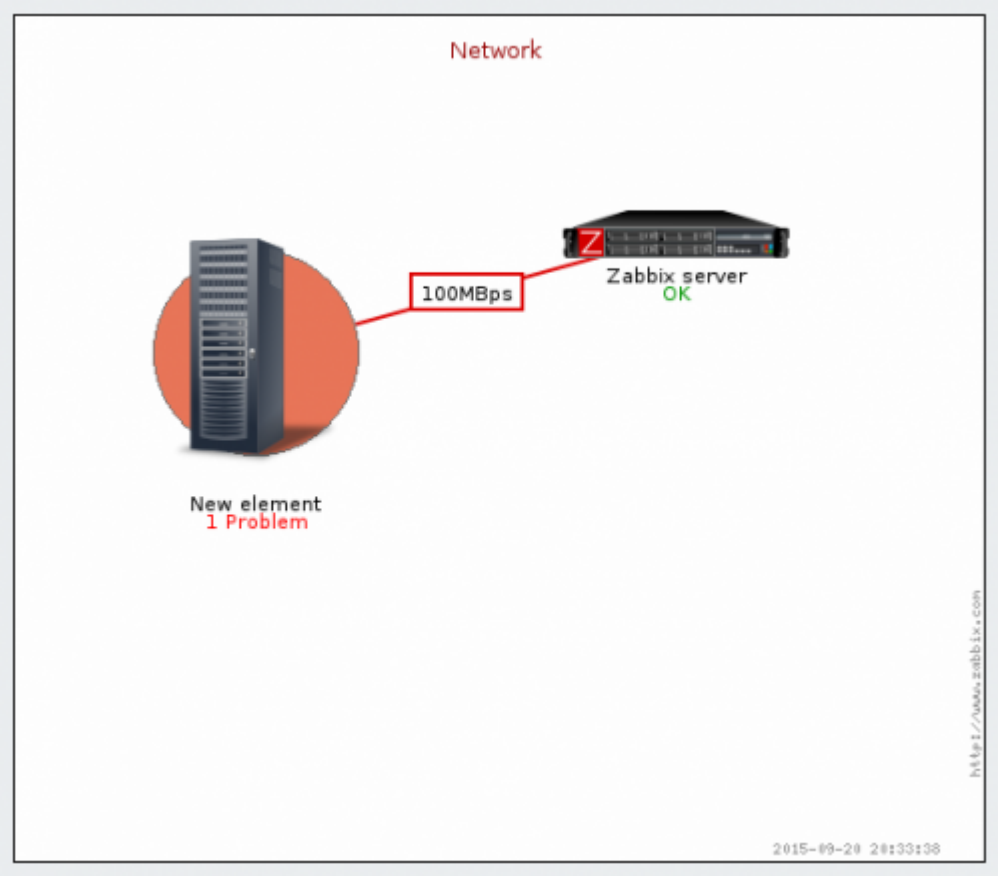
All mandatory input fields are marked with a red asterisk.

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.



Note:
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. *Minimimum trigger severity* and *Show suppressed problem* settings from map configuration affect which problems are taken into account.
2. In case with triggers with multiple problems (multiple problem generation), each problem may have severity that differs from trigger severity (changed manually), may have different tags (due to macros) and may be suppressed.

3 Screens

Overview

On Zabbix screens you can group information from various sources for a quick overview on a single screen. Building the screens is quite easy and intuitive.

Essentially a screen is a table. You choose how many cells per table and what elements to display in the cells. The following elements can be displayed:

- simple graphs
- simple graph prototypes
- user-defined custom graphs
- custom graph prototypes
- maps
- plain text information

- server information (overview)
- host information (overview)
- trigger information (overview)
- host/host group issues (status of problems)
- problems by severity
- data overview
- clock
- history of events
- history of recent actions
- URL (data taken from another location)

Global screens are managed in *Monitoring* → *Screens*, where they can be configured, managed and viewed. They can also be added to the favorites section of *Monitoring* → *Dashboard*.

Host-level screens are configured on template level and then generated for hosts once the template is linked to the hosts.

To configure a screen you must first create it by defining its general properties and then add individual elements in the cells.

All users in Zabbix (including non-admin users) can create screens. Screens have an owner - the user who created them.

Screens can be made public or private. Public screens are visible to all users.

Private screens are visible only to their owner. Private screens can be shared by the owner to other users and user groups. Regular (non-Super admin) users can only share with the groups and users they are member of. Private screens will be visible to their owner and the users the screen is shared with as long as they have read permissions to all screen elements. Admin level users, as long as they have read permissions to all screen elements, can see and edit private screens regardless of being the owner or belonging to the shared user list.

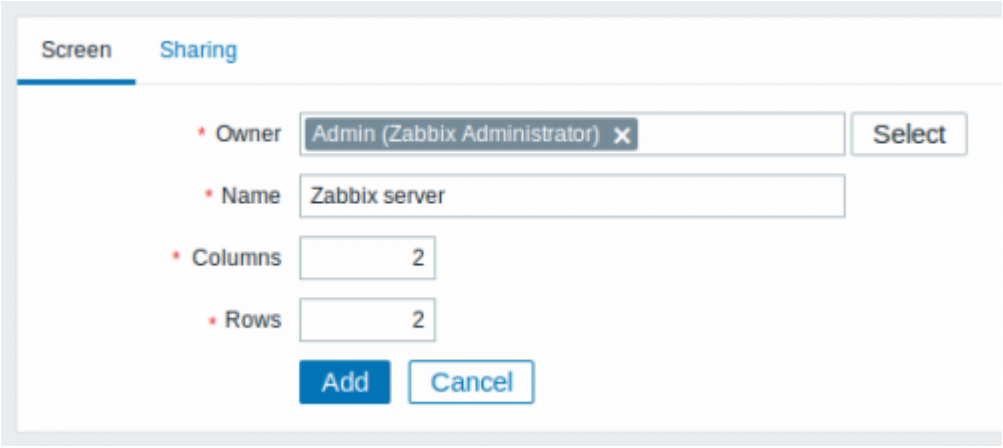
Warning:
For both public and private screens a user must have at least read permissions to all screen elements in order to see the screen. To add an element to a screen a user must also have at least read permission to it.

Creating a screen

To create a screen, do the following:

- Go to *Monitoring* → *Screens*
- Go to the view with all screens
- Click on *Create screen*

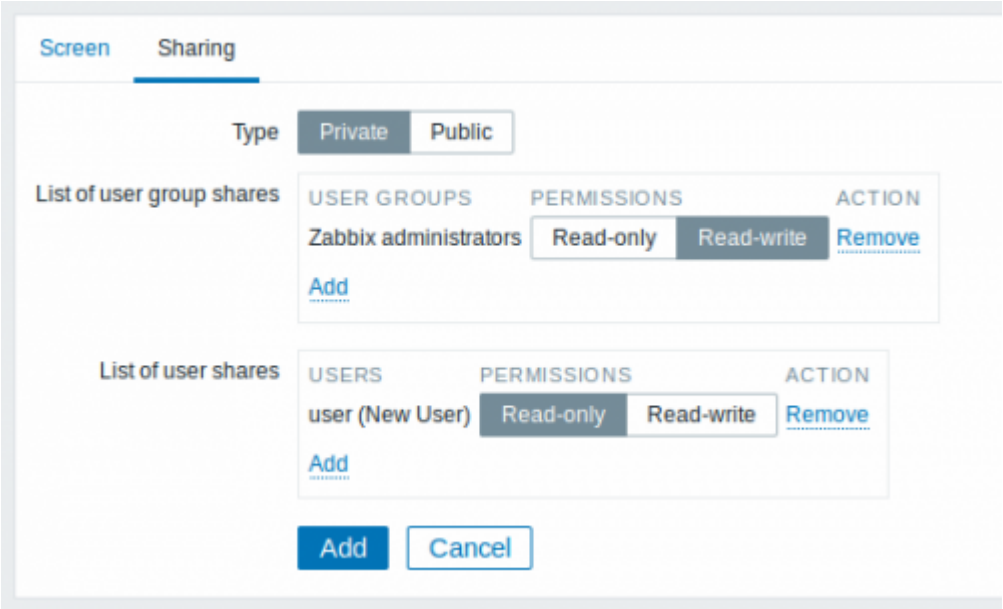
The **Screen** tab contains general screen attributes:



All mandatory input fields are marked with a red asterisk.

Give your screen a unique name and set the number of columns (vertical cells) and rows (horizontal cells).

The **Sharing** tab contains the screen type as well as sharing options (user groups, users) for private screens:



Parameter	Description
Owner	Select the screen owner.
Type	Select screen type: **Private** - screen is visible only to selected user groups and users **Public** - screen is visible to all
List of user group shares	Select user groups that the screen is accessible to. You may allow read-only or read-write access.
List of user shares	Select users that the screen is accessible to. You may allow read-only or read-write access.

Click on *Add* to save the screen.

Adding elements

To add elements to the screen, click on *Constructor* next to the screen name in the list.

On a new screen you probably only see links named *Change*. Clicking those links opens a form whereby you set what to display in each cell.

On an existing screen you click on the existing elements to open the form whereby you set what to display.

All mandatory input fields are marked with a red asterisk.

Screen element attributes:

Parameter	Description
Resource	Information displayed in the cell: **Action log** - history of recent actions **Clock** - digital or analog clock displaying current server or local time **Data overview** - latest data for a group of hosts **Graph** - single custom graph **Graph prototype** - custom graph from low-level discovery rule **History of events** - latest events **Host group issues** - status of triggers filtered by the host group (includes triggers without events, since Zabbix 2.2) **Host info** - high level host related information **Host issues** - status of triggers filtered by the host (includes triggers without events, since Zabbix 2.2) **Map** - single map **Plain text** - plain text data **Simple graph** - single simple graph **Simple graph prototype** - simple graph based on item generated by low-level discovery **System information** - high-level information about Zabbix server **Problems by severity** - displays problems by severity (similar to the Dashboard) **Trigger info** - high level trigger related information **Trigger overview** - status of triggers for a host group **URL** - include content from the specified resource See also more information on configuring each resource.
Horizontal align	Possible values: **Center** **Left** **Right**
Vertical align	Possible values: **Middle** **Top** **Bottom**
Column span	Extend cell to a number of columns, same way as HTML column spanning works.
Row span	Extend cell to a number of rows, same way as HTML row spanning works.

Take note of the '+' and '-' controls on each side of the table.

Clicking on '+' above the table will add a column. Clicking on '-' beneath the table will remove a column.

Clicking on '+' on the left side of the table will add a row. Clicking on '-' on the right side of the table will remove a row.

Attention:
If graph height is set as less than 120 pixels, no trigger will be displayed in the legend.

Dynamic elements

For some of the elements there is an extra option called *Dynamic item*. Checking this box at first does not to seem to change anything.

However, once you go to *Monitoring* → *Screens*, you may realize that now you have an extra multiselect field there for selecting the host. Thus you have a screen where some elements display the same information while others display information depending on the currently selected host.

The benefit of this is that you do not need to create extra screens just because you want to see the same graphs containing data from various hosts.

Dynamic item option is available for several screen elements:

- Graphs (custom graphs)
- Graph prototypes
- Simple graphs

- Simple graph prototypes
- Plain text
- URL

Note:
Clicking on a dynamic graph opens it in full view; although with custom graphs and graph prototypes that is currently supported with the default host only (i.e. with host 'not selected' in the dropdown). When selecting another host in the dropdown, the dynamic graph is created using item data of that host and the resulting graph is not clickable.

Note:
Dynamic URL elements will not be displayed in *Monitoring → Screens*, unless a host is selected. Without a selected host the "No host selected" message will be visible only.

1 Screen elements

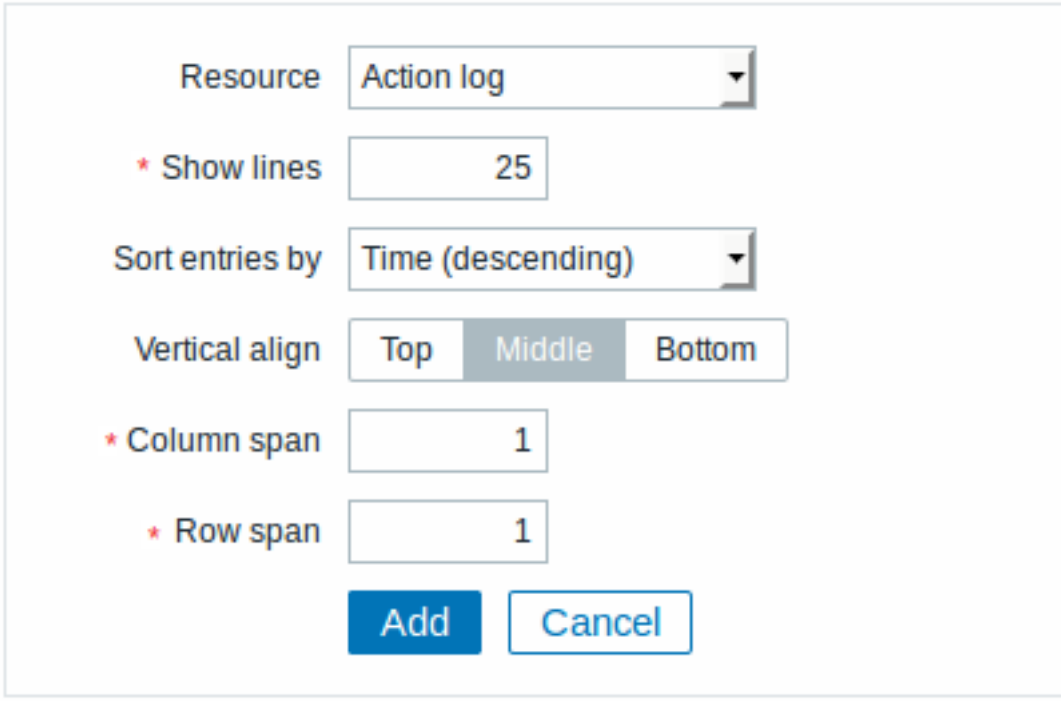
Overview

This section lists available screen elements and provides details for screen element configuration.

1 Action log

In the action log element you can display details of action operations (notifications, remote commands). It replicates information from *Reports → Audit*.

To configure, select *Action log* as resource:



All mandatory input fields are marked with a red asterisk.

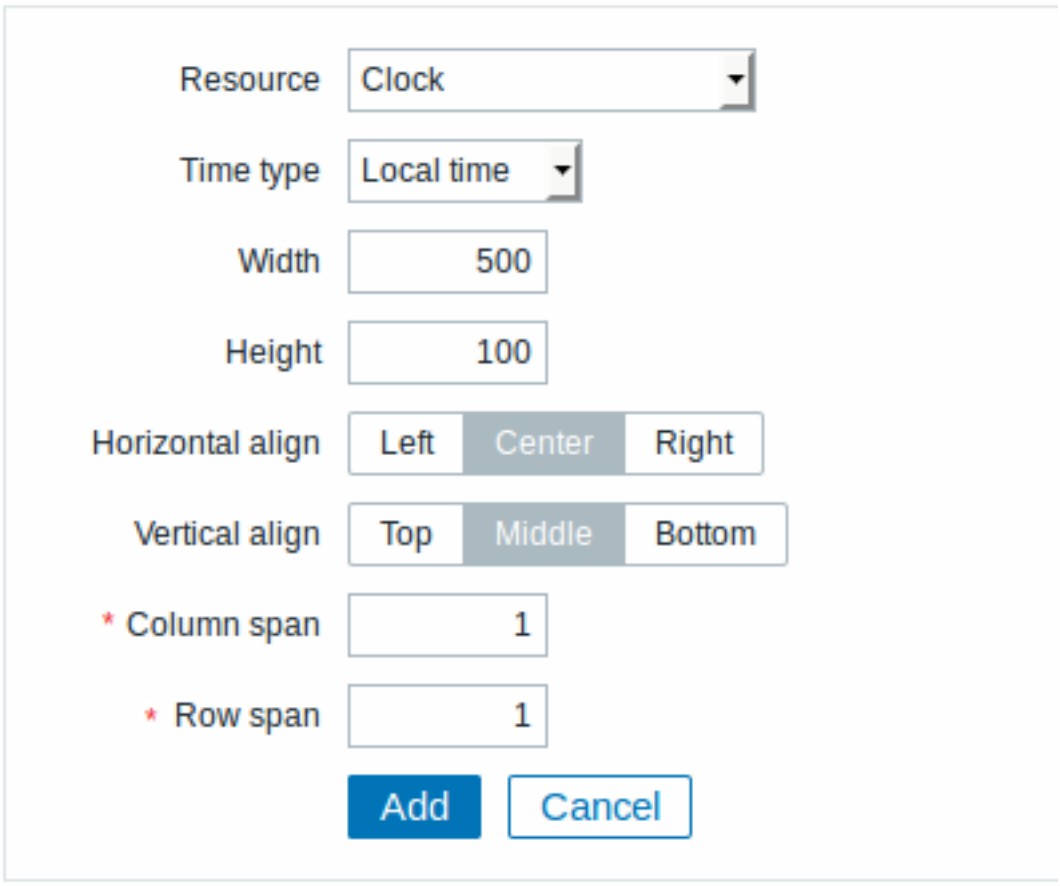
You may set the following specific options:

Show lines	Set how many action log lines will be displayed in the screen cell.
Sort entries by	Sort entries by: **Time** (descending or ascending) **Type** (descending or ascending) **Status** (descending or ascending) **Recipient** (descending or ascending).

2 Clock

In the clock element you may display local, server or specified host time.

To configure, select *Clock* as resource:



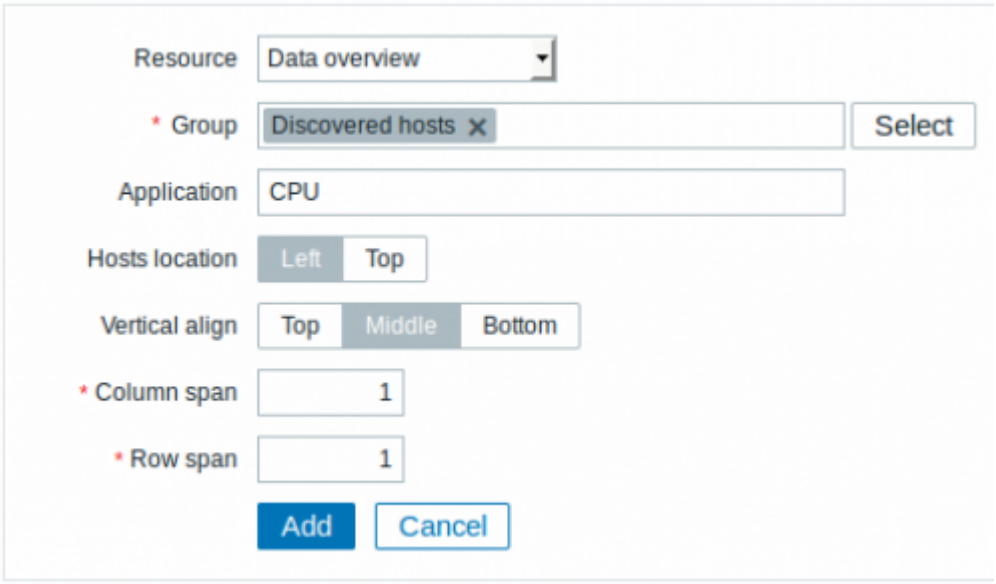
You may set the following specific options:

Time type	Select local, server or specified host time.
Item	Select the item for displaying time. To display host time, use the `system.localtime[local]` item. This item must exist on the host. This field is available only when *Host time* is selected.
Width	Select clock width.
Height	Select clock height.

3 Data overview

In the data overview element you can display the latest data for a group of hosts. It replicates information from *Monitoring* → *Overview* (when *Data* is selected as Type there).

To configure, select *Data overview* as resource:



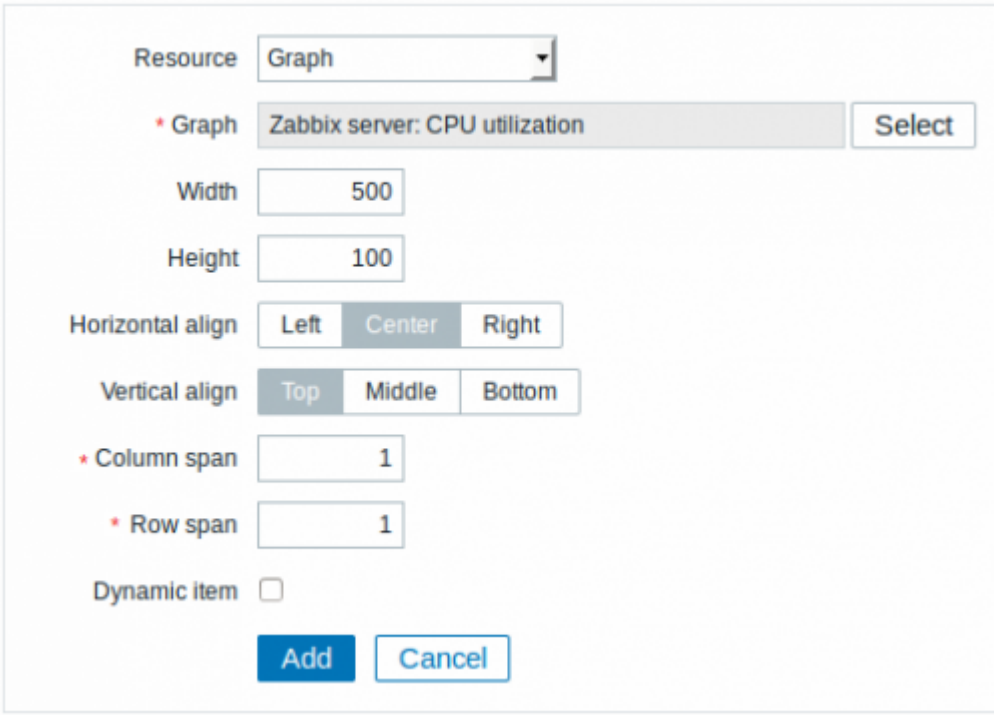
You may set the following specific options:

Group	Select host group.
Application	Enter application name.
Hosts location	Select host location - left or top.

4 Graph

In the graph element you can display a single custom graph.

To configure, select *Graph* as resource:



You may set the following specific options:

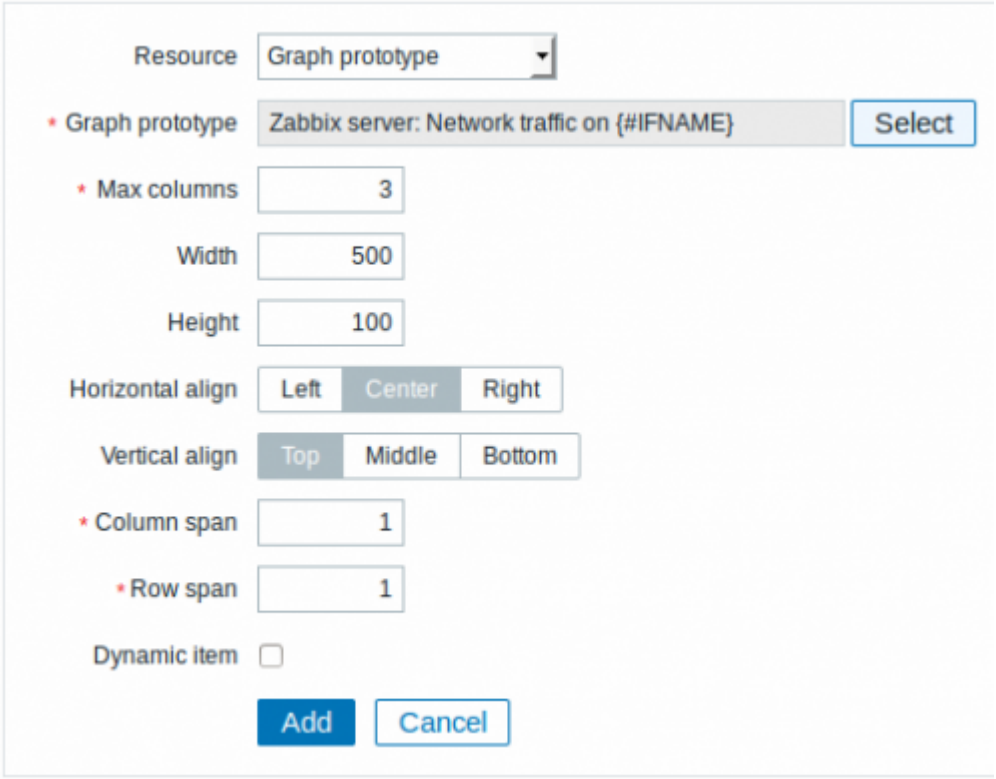
Graph	Select the graph to display.
Width	Select graph width. Note that a line graph may actually take up more space due to legend text.

Height	Select graph height. Note that a line graph may actually take up more space due to legend text.
Dynamic item	Set graph to display different data depending on the selected host.

5 Graph prototype

In the graph prototype element you can display a custom graph from a low-level discovery rule.

To configure, select *Graph prototype* as resource:



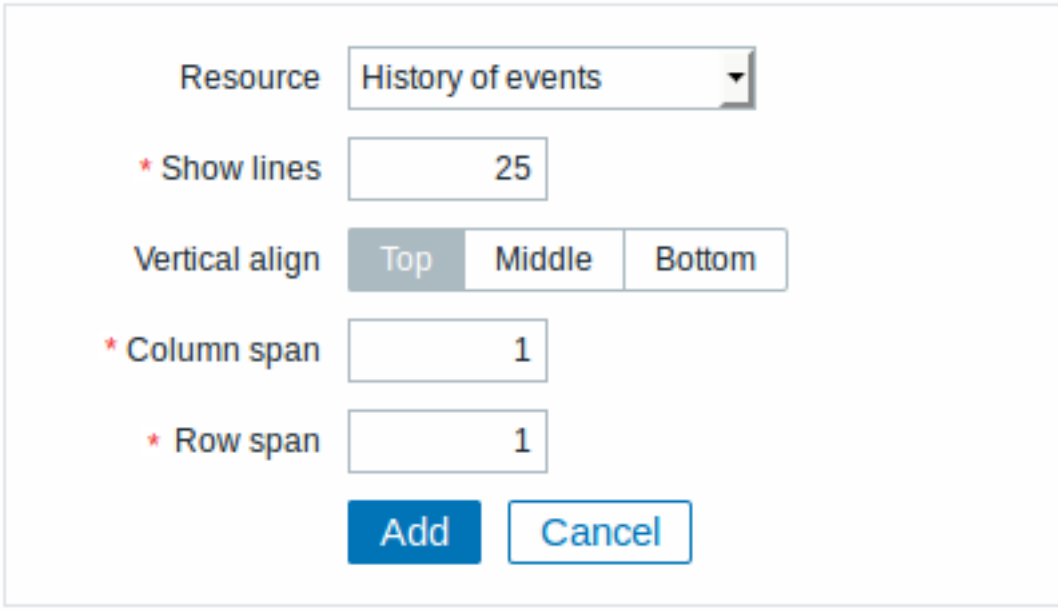
You may set the following specific options:

Graph prototype	Select the graph prototype to display.
Max columns	In how many columns generated graphs should be displayed in the screen cell. Useful when there are many LLD-generated graphs.
Width	Select graph width. Note that a line graph may actually take up more space due to legend text.
Height	Select graph height. Note that a line graph may actually take up more space due to legend text.
Dynamic item	Set graph to display different data depending on the selected host.

6 History of events

In the history of events element you can display latest events.

To configure, select *History of events* as resource:



You may set the following specific option:

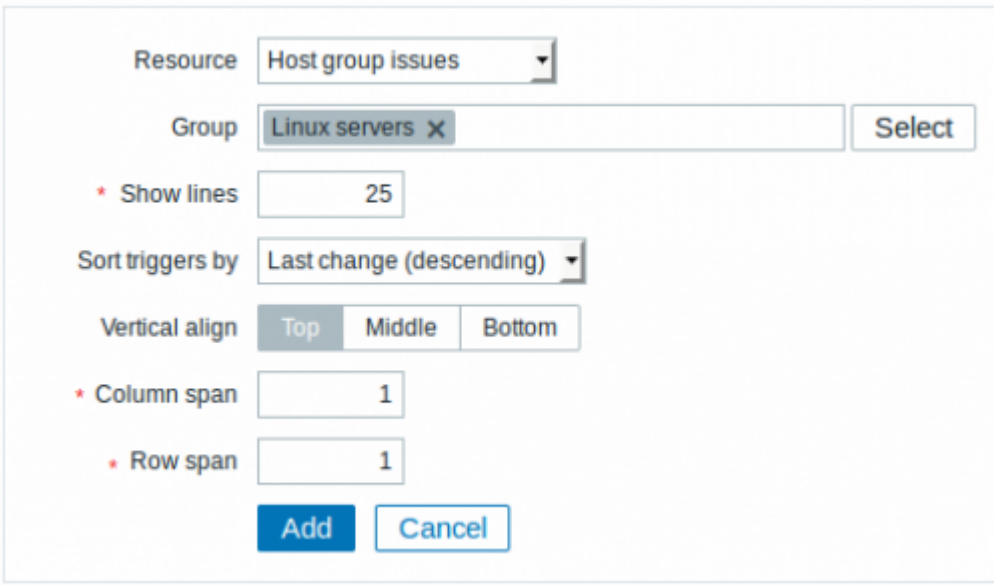
Show lines	Set how many event lines will be displayed in the screen cell.

7 Host group issues

In the host group issue element you can display problem details filtered by the selected host group.

The problem severity color displayed is originally from the underlying trigger, but can be adjusted in the problem update screen.

To configure, select *Host group issues* as resource:



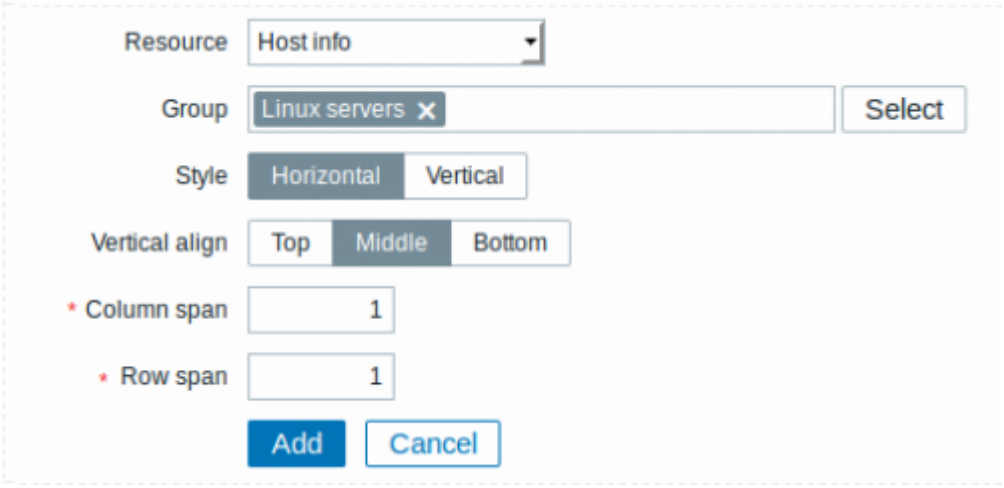
You may set the following specific options:

Group	Select host group.
Show lines	Set how many problem lines will be displayed in the screen cell.
Sort triggers by	Select from the dropdown to sort problems by last change, severity (both descending) or host (ascending).

8 Host info

In the host information element you can display high-level information about host availability.

To configure, select *Host info* as resource:



You may set the following specific options:

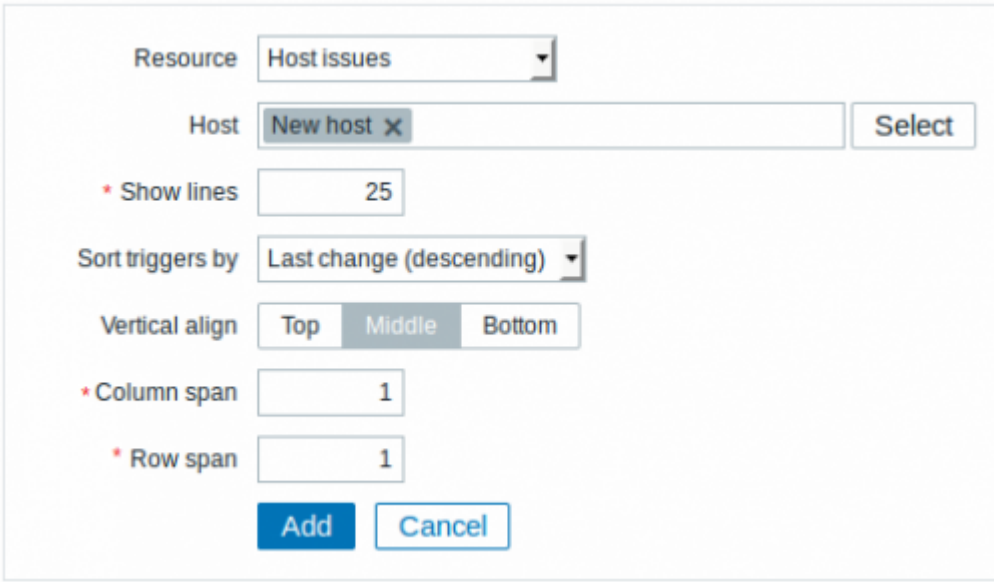
Group	Select host group(s).
Style	Select vertical or horizontal display.

9 Host issues

In the host issue element you can display problem details filtered by the selected host.

The problem severity color displayed is originally from the underlying trigger, but can be adjusted in the problem update screen.

To configure, select *Host issues* as resource:



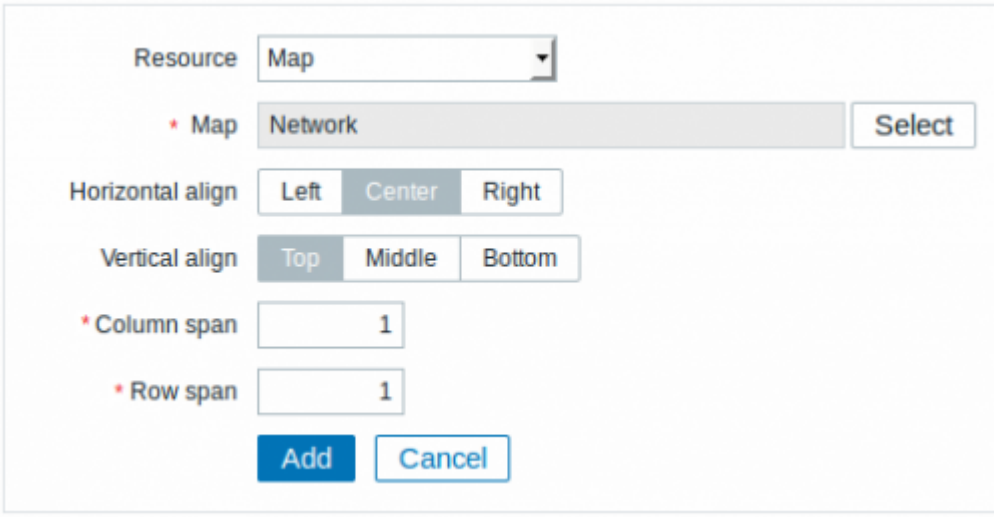
You may set the following specific options:

Host	Select the host.
Show lines	Set how many problem lines will be displayed in the screen cell.
Sort triggers by	Select from the dropdown to sort problems by last change, severity (both descending) or host (ascending).

10 Map

In the map element you can display a configured network map.

To configure, select *Map* as resource:



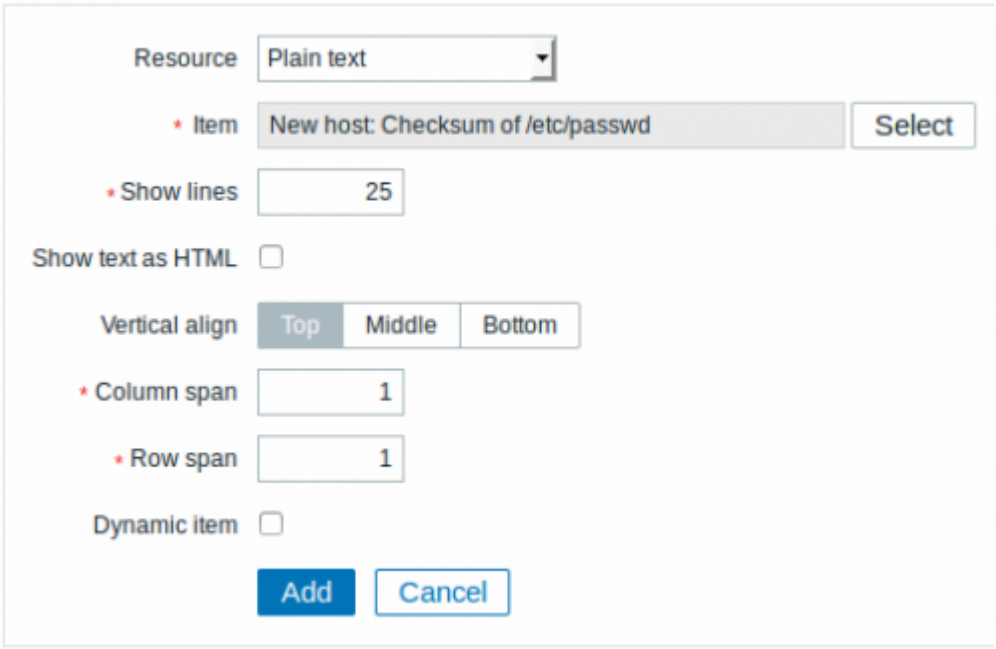
You may set the following specific options:

Map	Select the map to display.

11 Plain text

In the plain text element you can display latest item data in plain text.

To configure, select *Plain text* as resource:



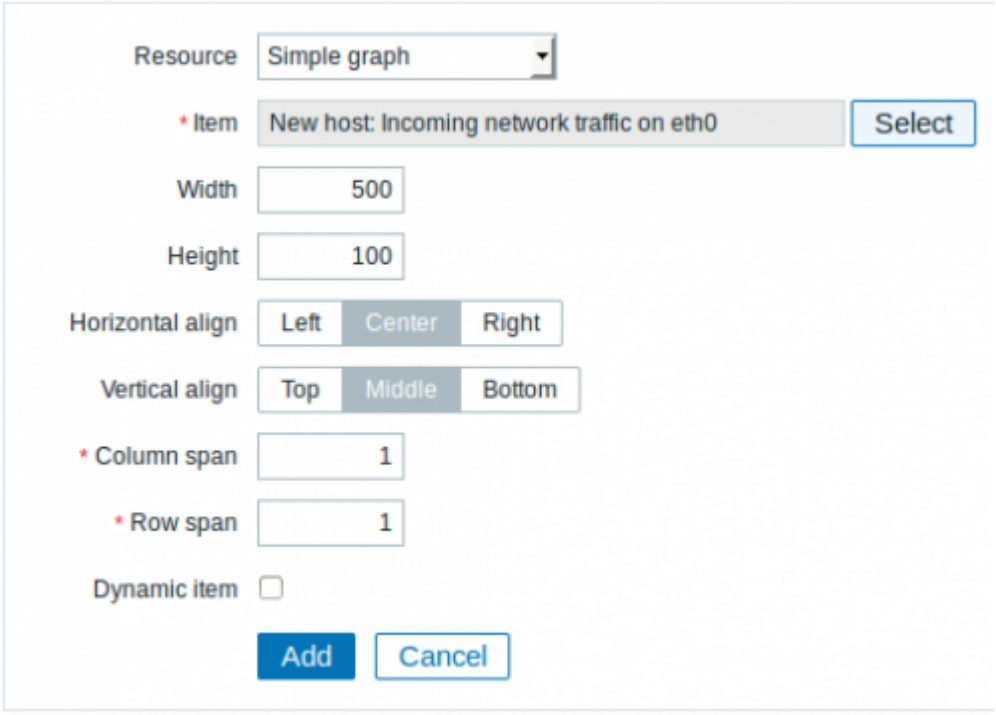
You may set the following specific options:

Item	Select the item.
Show lines	Set how many latest data lines will be displayed in the screen cell.
Show text as HTML	Set to display text as HTML.
Dynamic item	Set to display different data depending on the selected host.

12 Simple graph

In the simple graph element you can display a single simple graph.

To configure, select *Simple graph* as resource:



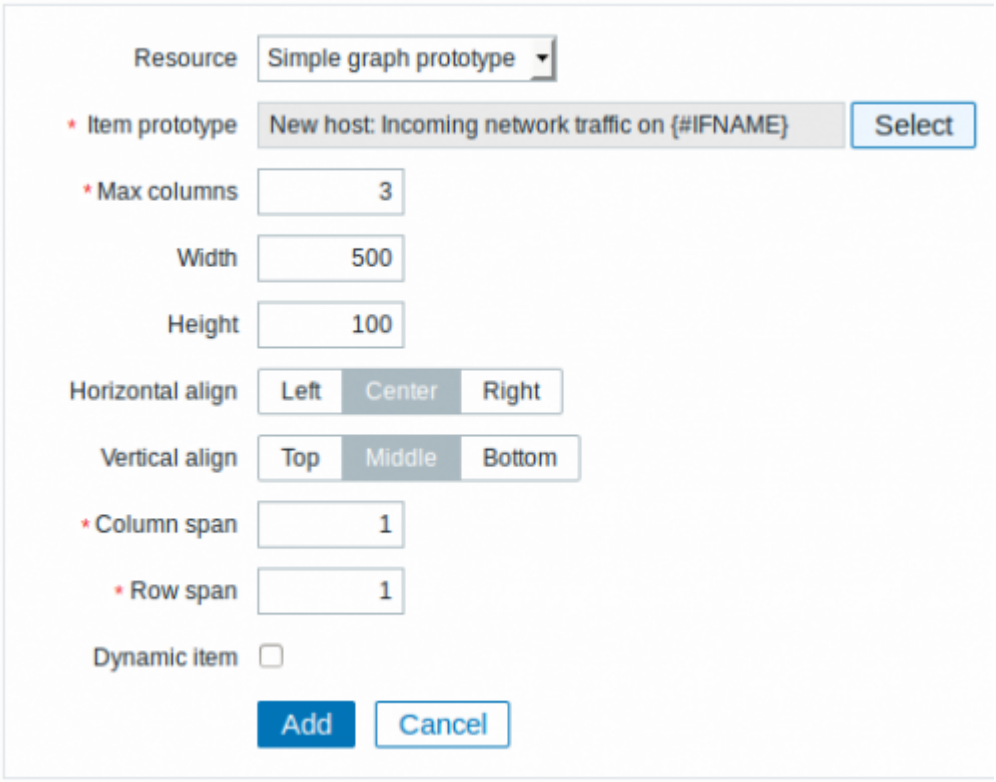
You may set the following specific options:

Item	Select the item for the simple graph.
Width	Select graph width. Note that a line graph may actually take up more space due to legend text.
Height	Select graph height. Note that a line graph may actually take up more space due to legend text.
Dynamic item	Set graph to display different data depending on the selected host.

13 Simple graph prototype

In the simple graph prototype element you can display a simple graph based on an item generated by low-level discovery.

To configure, select *Simple graph prototype* as resource:



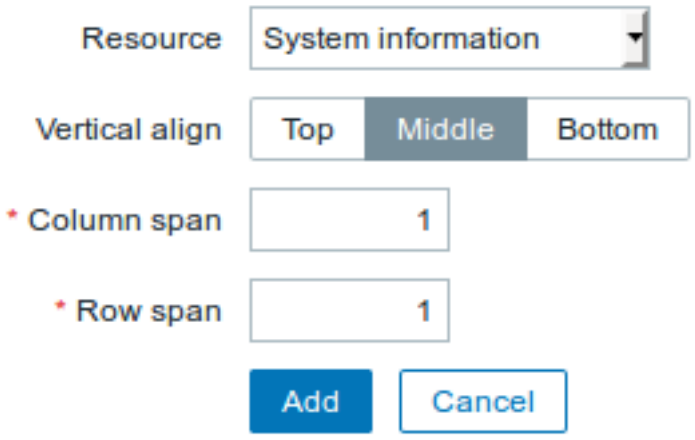
You may set the following specific options:

Item prototype	Select the item prototype for the simple graph.
Max columns	In how many columns generated graphs should be displayed in the screen cell. Useful when there are many LLD-generated graphs.
Width	Select graph width. Note that a line graph may actually take up more space due to legend text.
Height	Select graph height. Note that a line graph may actually take up more space due to legend text.
Dynamic item	Set graph to display different data depending on the selected host.

14 System information

In the system information element you can display high-level Zabbix and Zabbix server information.

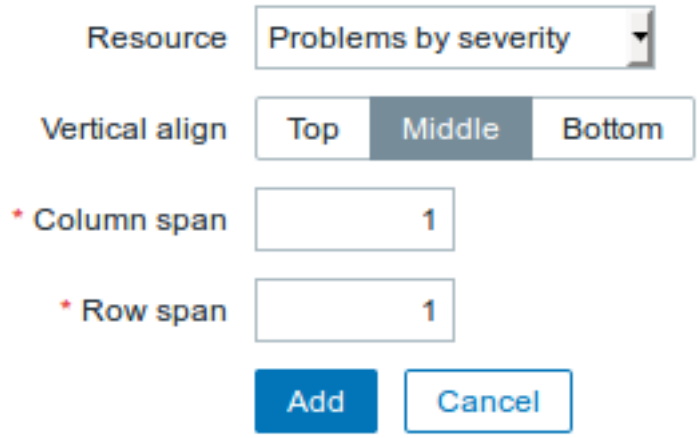
To configure, select *System information* as resource:



15 Problems by severity

In this element you can display problems by severity similarly as in the Dashboard widget.

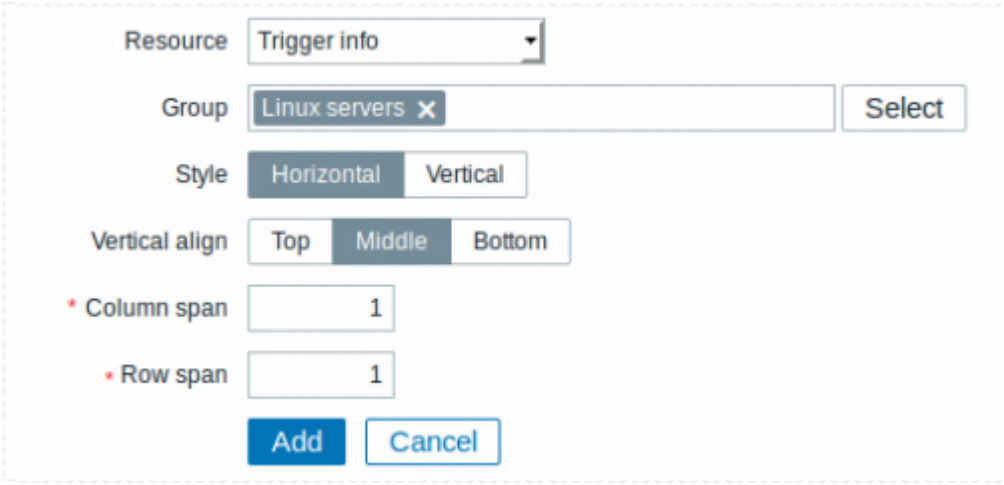
To configure, select *Problems by severity* as resource:



16 Trigger info

In the trigger info element you can display high-level information about trigger states.

To configure, select *Trigger info* as resource:



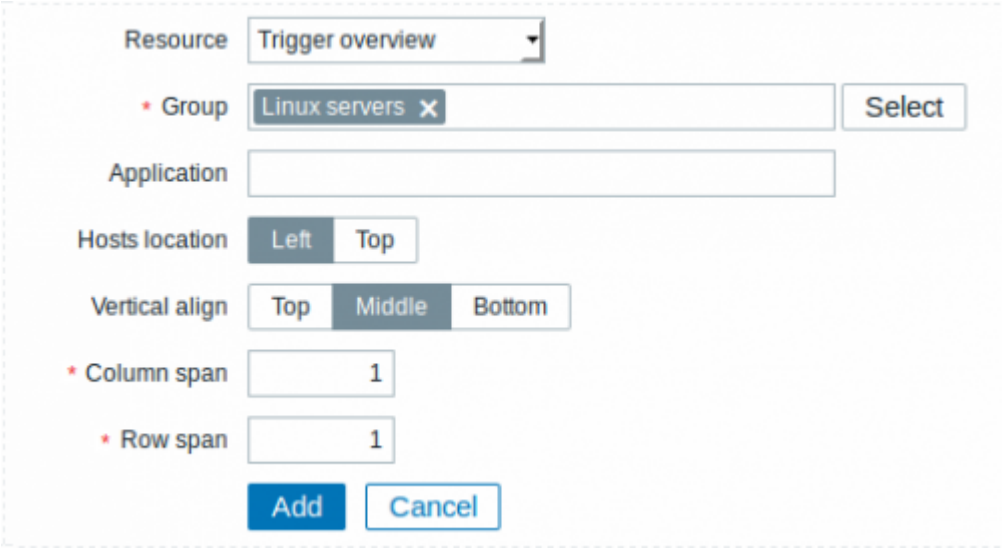
You may set the following specific options:

Group	Select the host group(s).
Style	Select vertical or horizontal display.

17 Trigger overview

In the trigger overview element you can display the trigger states for a group of hosts. It replicates information from *Monitoring* → *Overview* (when *Triggers* is selected as Type there).

To configure, select *Trigger overview* as resource:



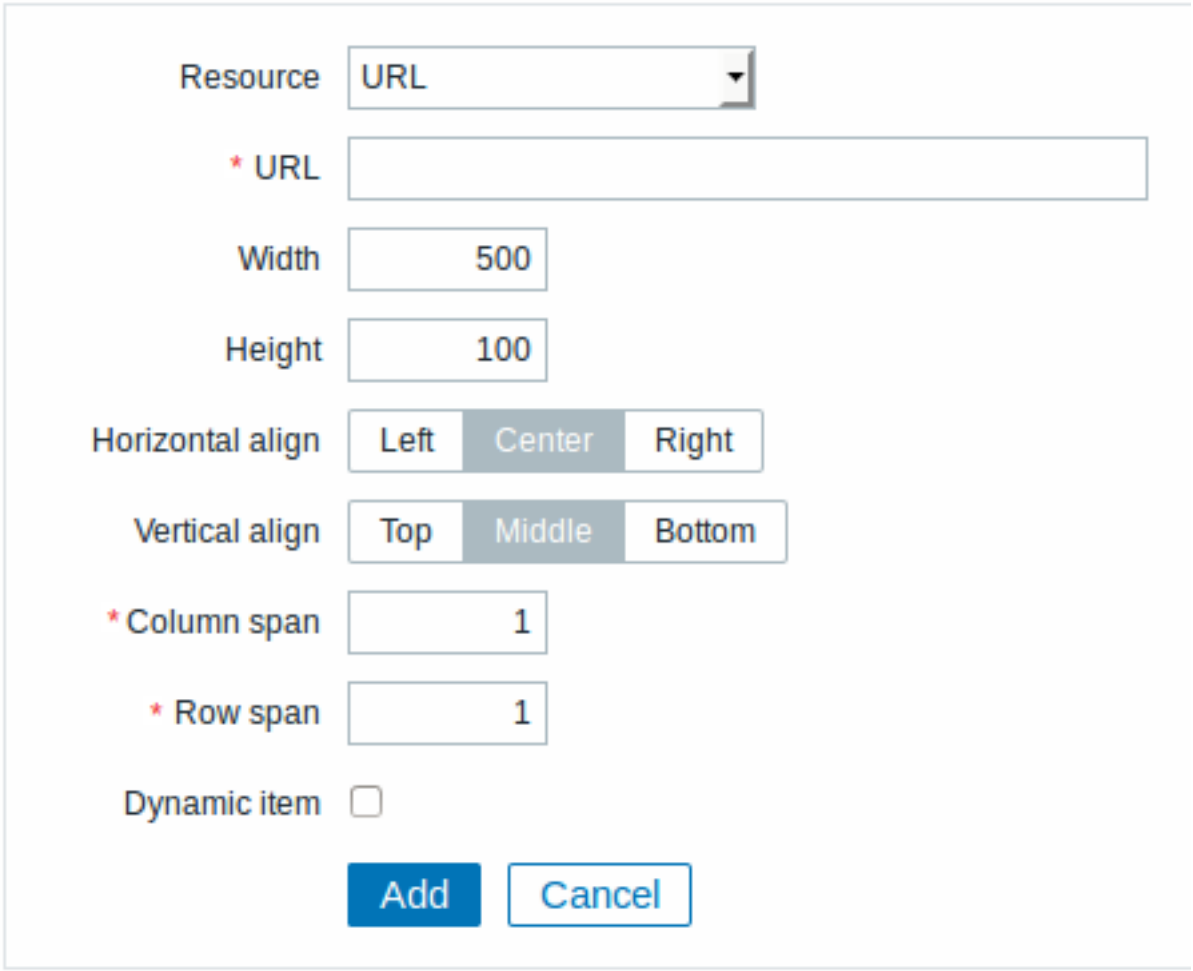
You may set the following specific options:

Group	Select the host group(s).
Application	Enter the application name.
Hosts location	Select host location - left or top.

18 URL

The URL element displays the content retrieved from the specified URL.

To configure, select *URL* as resource:



You may set the following specific options:

URL	Enter the URL to display. External URLs must start with `http://` or `https://`. Since Zabbix 4.4.8, internal URLs support relative paths (for example, `zabbix.php?action=report.status`).
Width	Select window width.
Height	Select window width.
Dynamic item	Set to display different URL content depending on the selected host.

Attention:
Browsers might not load an HTTP page included in a screen (using URL element), if Zabbix frontend is accessed over HTTPS.

4 Slide shows

Overview

In a slide show you can configure that a number of screens are displayed one after another at set intervals.

Sometimes you might want to switch between some configured screens. While that can be done manually, doing that more than once or twice may become very tedious. This is where the slide show function comes to rescue.

All users in Zabbix (including non-admin users) can create slide shows. Slide shows have an owner - the user who created them.

Slide shows can be made public or private. Public slide shows are visible to all users, however, they must have at least read permissions to all slide show elements (screens) to see it. To add a screen to the slide show the user must also have at least read permission to it.

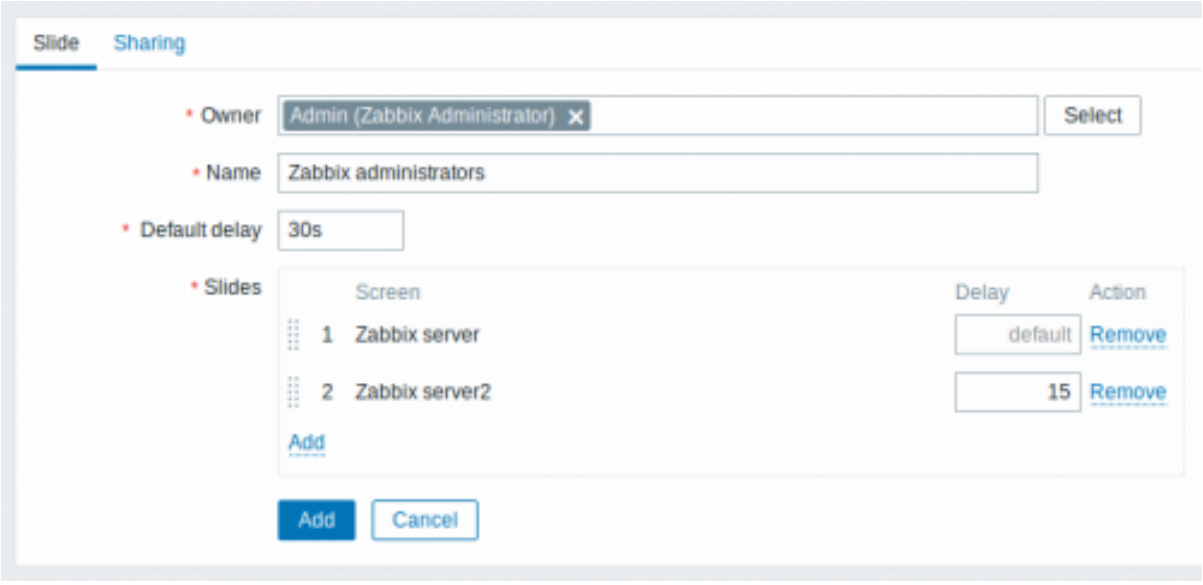
Private slide shows are visible only to their owner. Private slide shows can be shared by the owner to other users and user groups. Regular (non-Super admin) users can only share with the groups and users they are member of. Private slide shows will be visible to their owner and the users the slide show is shared with as long as they have read permissions to all included screens. Admin level users, as long as they have read permissions to all included screens, can see and edit private slide shows regardless of being the owner or belonging to the shared user list.

Configuration

To create a slide show, do the following:

- Go to *Monitoring* → *Screens*
- Select *Slide shows* in the title dropdown
- Click on *Create slide show*

The **Slide** tab contains general slide show attributes:



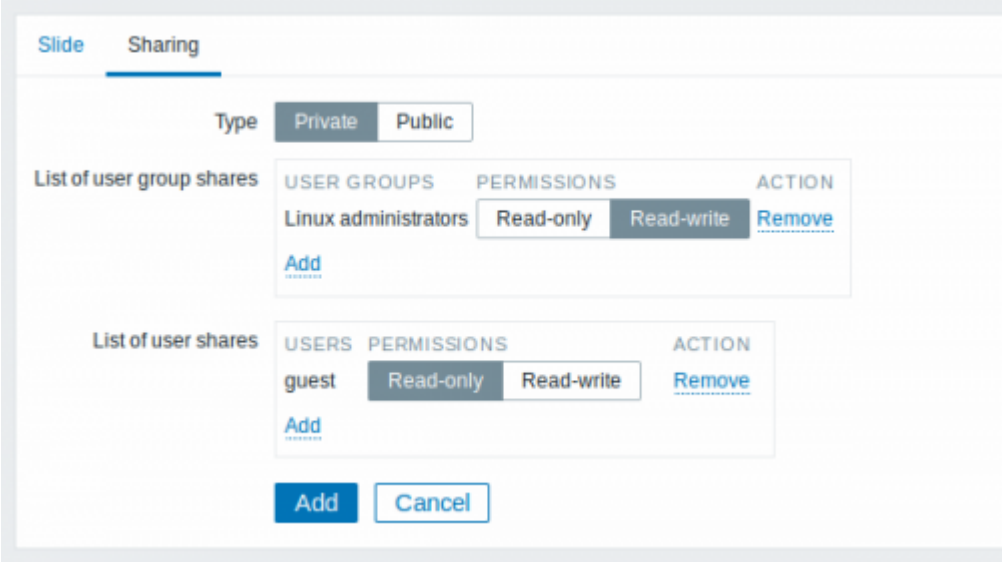
All mandatory input fields are marked with a red asterisk.

Parameter	Description
Owner	Select the slide show owner. Specifying owner is mandatory.
Name	Unique name of the slide show.
Default delay	How long one screen is displayed by default, before rotating to the next. Time suffixes are supported, e.g. 30s, 5m, 2h, 1d.
Slides	List of screens to be rotated. Click on *Add* to select screens. The *Up/Down* arrow before the screen allows to drag a screen up and down in the sort order of display. If you want to display only, say, a single graph in the slide show, create a screen containing just that one graph.
Screen	Screen name.
Delay	A custom value for how long the screen will be displayed, in seconds. If set to 0, the *Default delay* value will be used.
Action	Click on *Remove* to remove a screen from the slide show.

The slide show in this example consists of two screens which will be displayed in the following order:

Zabbix server ⇒ Displayed for 30 seconds ⇒ Zabbix server2 ⇒ Displayed for 15 seconds ⇒ Zabbix server ⇒ Displayed for 30 seconds ⇒ Zabbix server2 ⇒ ...

The **Sharing** tab contains the slide show type as well as sharing options (user groups, users) for private slide shows:



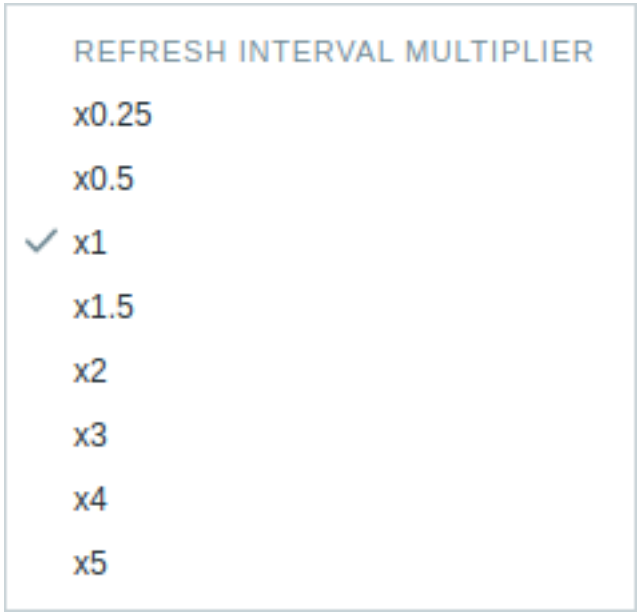
Parameter	Description
Type	Select slide show type: **Private** - slide show is visible only to selected user groups and users **Public** - slide show is visible to all
List of user group shares	Select user groups that the slide show is accessible to. You may allow read-only or read-write access.
List of user shares	Select users that the slide show is accessible to. You may allow read-only or read-write access.

Click on *Add* to save the slide show.

Display

Slide shows that are ready can be viewed in *Monitoring* → *Screens*, then choosing *Slide shows* in the title dropdown and clicking on the slide show name.

With the Menu option next to the dropdown, you can accelerate or slow down the display by choosing a slide delay multiplier:



Attention:
If a delay ends up as being less than 5 seconds (either by having entered a delay less than 5 seconds or by using the slide delay multiplier), a 5-second minimum delay will be used.

5 Host screens

Overview

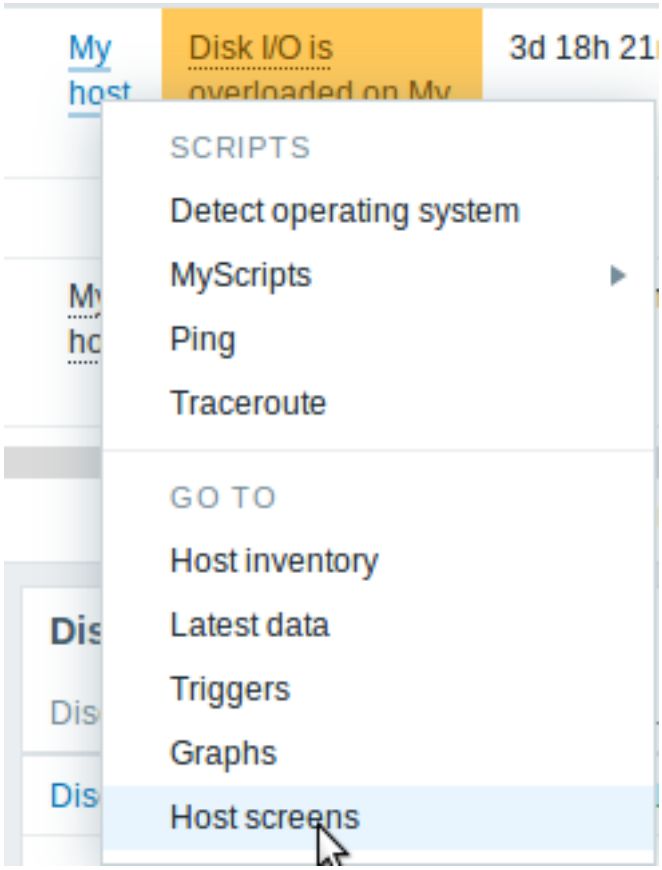
Host screens look similar to global screens, however, host screens display data about the host only. Host screens are configured on the template level and then are generated for a host, once the template is linked to the host.

Host screens *cannot* be configured or directly accessed in the *Monitoring* → *Screens* section, which is reserved for global screens. The ways to access host screens are listed below in this section.

Accessing host screens

Access to host screens is provided:

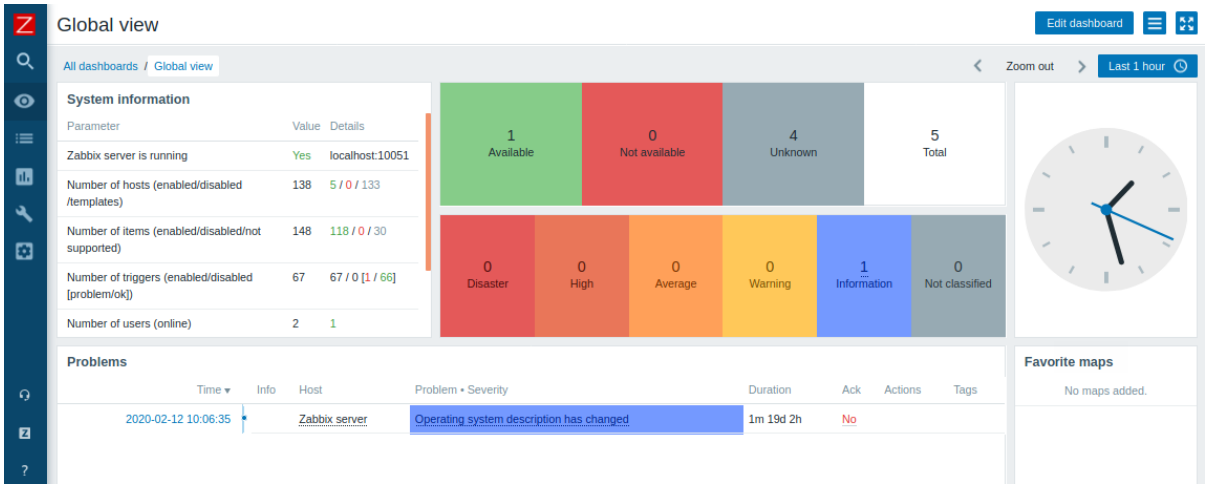
- From the host menu that is available in many frontend locations:
  - click on the host name and then select *Host screens* from the drop-down menu



- When searching for a host name in global search:
  - click on the *Screens* link provided in search results
- When clicking on a host name in *Inventory* → *Hosts*:
  - click on the *Screens* link provided

6 Dashboard

The dashboard and its widgets provide a strong visualization platform with such tools as modern graphs, maps and many more.



7 Templates

Overview

A template is a set of entities that can be conveniently applied to multiple hosts.

The entities may be:

- items

- triggers
- graphs
- applications
- screens
- 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.

Thus, the use of templates is an excellent way of reducing one's workload and streamlining the Zabbix configuration.

Proceed to creating and configuring a template.

8 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 {HOST.CONN} 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.

Note:
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).

Template	Mandatory macros	Additional steps/comments
DELL PowerEdge R720 by HTTP, DELL PowerEdge R740 by HTTP, DELL PowerEdge R820 by HTTP, DELL PowerEdge R840 by HTTP	**{$API.URL}** - Dell iDRAC Redfish API URL in the format `<scheme>://<host>:<port>` (default: <Put your URL here>) **{$API.USER}**, **{$API.PASSWORD}** - Dell iDRAC login credentials (default: not set).	In the Dell iDRAC interface of your server: 1. Enable Redfish API . 2. Create a user for monitoring with read-only permissions.

Template App Apache by HTTP	**{$APACHE.STATUS.HOST}** - the hostname or IP address of Apache status page (default: 127.0.0.1).
	{$APACHE.STATUS.PATH} - the URL path (default: server-status?auto).
	{$APACHE.STATUS.PORT} - the port of Apache status page (default: 80).
	{$APACHE.STATUS.SCHEME} - the request scheme. Supported: http (default), https.
		Apache module mod_status should be set (see Apache documentation for details).
		To check availability, run:
		`httpd -M 2>/dev/null | grep status_module`
		
		Apache configuration example:
		`<Location "/server-status">`
		`SetHandler server-status`
		`Require host example.com`
		`</Location>`

Template App Elasticsearch Cluster by HTTP	**{$ELASTICSEARCH.PORT}** - the port of the Elasticsearch host (default: 9200). **{$ELASTICSEARCH.SCHEME}** - the request scheme. Supported: http (default), https. **{$ELASTICSEARCH.USERNAME}**, **{$ELASTICSEARCH.PASSWORD}** - login credentials, required only if used for Elasticsearch authentication.	-

Template App Etcd by HTTP	**{$ETCD.PORT}**- the port used by Etcd API endpoint (default: 2379).
	{$ETCD.SCHEME} - the request scheme. Supported: http (default), https.
	{$ETCD.USER}**, **{$ETCD.PASSWORD} - login credentials, required only if used for Etcd authentication.
		Metrics are collected from /metrics endpoint; to specify the endpoint's location use `--listen-metrics-urls` flag (see Etcd documentation for details).
		
		To verify, whether Etcd is configured to allow metric collection, run:
		`curl -L http://localhost:2379/metrics`
		
		To check, if Etcd is accessible from Zabbix proxy or Zabbix server run:
		`curl -L http:%%//<etcd_node_adress>:2379/metrics%%`
		
		The template should be added to each node with Etcd.

Template	Mandatory macros	Additional steps/comments
Template App Hadoop by HTTP	**{$HADOOP.CAPACITY_REMAINING.MIN.WARN}** - the Hadoop cluster capacity remaining percent for trigger expression (default: 20). **{$HADOOP.NAMENODE.HOST}** - the Hadoop NameNode host IP address or FQDN (default: NameNode). **{$HADOOP.NAMENODE.PORT}** - the Hadoop NameNode web-UI port (default: 9870). **{$HADOOP.NAMENODE.RESPONSE_TIME.MAX.WARN}** - the Hadoop NameNode API page maximum response time in seconds for trigger expression (default: 10s). **{$HADOOP.RESOURCEMANAGER.HOST}** - the Hadoop ResourceManager host IP address or FQDN (default: ResourceManager). **{$HADOOP.RESOURCEMANAGER.PORT}** - the Hadoop ResourceManager web-UI port (default: 8088). **{$HADOOP.RESOURCEMANAGER.RESPONSE_TIME.MAX.WARN}** - the Hadoop ResourceManager API page maximum response time in seconds for trigger expression (default: 10s).	Metrics are collected by polling the Hadoop API remotely using an HTTP agent and JSONPath preprocessing. Zabbix server (or proxy) execute direct requests to ResourceManager, NodeManagers, NameNode, DataNodes APIs. All metrics are collected at once, thanks to the Zabbix bulk data collection.
Template App HAProxy by HTTP	**{$HAPROXY.STATS.PATH}** - the path of HAProxy Stats page (default: stats). **{$HAPROXY.STATS.PORT}** - the port of the HAProxy Stats host or container (default: 8404). **{$HAPROXY.STATS.SCHEME}** - the request scheme. 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).
Template App 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`
NGINX Plus by HTTP	**{$NGINX.API.ENDPOINT}** - NGINX Plus API URL in the format `<scheme>://<host>:<port>/<location>` (default: ' '). 2. Set the macro {$NGINX.API.ENDPOINT} 3. If required, use other template macros to filter out discovery operations and discover only required zones and upstreams.	1. Enable NGINX Plus API (see NGINX documentation for details).

Template	Mandatory macros	Additional steps/comments	
	Template App 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*

Template	Mandatory macros	Additional steps/comments

Template App 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 %% "" "" ".*"%%`
		`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.

Template DB ClickHouse by HTTP	**{$CLICKHOUSE.PORT}** - the port of ClickHouse HTTP endpoint (default: 8123). **{$CLICKHOUSE.SCHEME}** - the request scheme. Supported: http (default), https. **{$CLICKHOUSE.USER}**, **{$CLICKHOUSE.PASSWORD}** - ClickHouse login credentials (default username: zabbix, password: zabbix_pass). If you don't need authentication, remove headers from HTTP agent type items.	Create a ClickHouse user with a 'web' profile and permission to view databases (see ClickHouse documentation for details). See template's *Readme.md* file for a ready-to-use zabbix.xml file configuration.
Template SAN NetApp AFF A700 by HTTP	**{$URL}** - AFF700 cluster URL address (default: ' ') **{$USERNAME}**, **{$PASSWORD}** - AFF700 login credentials (default: not set).	Create a host for AFF A700 with cluster management IP as the Zabbix agent interface.
Template Tel Asterisk by HTTP	**{$AMI.PORT}** - AMI port number for checking service availability (default: 8088). **{$AMI.SECRET}** - the Asterisk Manager secret (default: zabbix). **{$AMI.URL}** - the Asterisk Manager API URL in the format `<scheme>://<host>:<port>/<prefix>/rawman` (default: http://asterisk:8088/asterisk/rawman). **{$AMI.USERNAME}** - the Asterisk Manager name.	1. Enable the mini-HTTP Server. 2. Add the option `webenabled=yes` to the general section of *manager.conf* file. 3. Create *Asterisk Manager* user in the Asterisk instance.
ZooKeeper by HTTP	**{$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).

> **Note:**
> 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).

Template	Mandatory macros	Additional steps/comments
Template Server Chassis by IPMI	**{$IPMI.USER}**, **{$IPMI.PASSWORD}** - credentials for access to BMC (default: none)	-

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.

> **Note:**
> 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).

Template	Mandatory macros	Additional steps/comments
Apache ActiveMQ by JMX	**{$ACTIVEMQ.PORT}** - port for JMX (default: 1099). **{$ACTIVEMQ.USERNAME}**, **{$ACTIVEMQ.PASSWORD}** - login credentials for JMX (default username: admin, password: activemq).	JMX access to Apache ActiveMQ should be enabled and configured per instructions in the official documentation.
Template DB Apache Cassandra by JMX	**{$CASSANDRA.USER}**, **{$CASSANDRA.PASSWORD}** - Apache Cassandra login credentials (default username: zabbix, password: zabbix).	JMX access to Apache Cassandra should be enabled and configured per instructions in the official documentation.
Apache Kafka by JMX	**{$KAFKA.USER}**, **{$KAFKA.PASSWORD}** - Apache Kafka login credentials (default username: zabbix, password: zabbix).	JMX access to Apache Kafka should be enabled and configured per instructions in the official documentation.
Apache Tomcat by JMX	**{$TOMCAT.USER}**, **{$TOMCAT.PASSWORD}** - Apache Tomcat login credentials; leave blank if Tomcat installation does not require authentication (default: not set).	JMX access to Apache Tomcat should be enabled and configured per instructions in the official documentation (choose the correct version).

Template	Mandatory macros	Additional steps/comments
GridGain by JMX	**{$GRIDGAIN.USER}**, **{$GRIDGAIN.PASSWORD}** - GridGain login credentials (default username: zabbix, password: <secret>).	JMX access to GridGain In-Memory Computing Platform should be enabled and configured per instructions in the documentation.
Ignite by JMX	**{$IGNITE.USER}**, **{$IGNITE.PASSWORD}** - Apache Ignite login credentials (default username: zabbix, password: <secret>).	Enable and configure JMX access to Apache Ignite. 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. Cache and Data Region metrics can be configured as needed - see Ignite documentation for details. See also: Monitoring and Management Using JMX Technology
WildFly Domain by JMX	**{$WILDFLY.JMX.PROTOCOL}** - JMX scheme (default: remote+http) **{$WILDFLY.USER}**, **{$WILDFLY.PASSWORD}** - WildFly login credentials (default username: zabbix, password: zabbix).	1. Enable and configure JMX access to WildFly according to instructions in the official documentation. 2. Copy *jboss-client.jar* from `/(wildfly,EAP,Jboss,AS)/bin/client` into directory `/usr/share/zabbix-java-gateway/lib`. 3. Restart Zabbix Java gateway.
WildFly Server by JMX	**{$WILDFLY.JMX.PROTOCOL}** - JMX scheme (default: remote+http) **{$WILDFLY.USER}**, **{$WILDFLY.PASSWORD}** - WildFly login credentials (default username: zabbix, password: zabbix).	1. Enable and configure JMX access to WildFly according to instructions in the official documentation. 2. Copy *jboss-client.jar* from `/(wildfly,EAP,Jboss,AS)/bin/client` into directory `/usr/share/zabbix-java-gateway/lib`. 3. Restart Zabbix Java gateway.

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.

Note:
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).

Template	Mandatory macros	Additional steps/comments
Template DB MSSQL by ODBC	**{$MSSQL.DSN}** - the system data source name (default: <Put your DSN here>) **{$MSSQL.PORT}** - the TCP port of Microsoft SQL Server (default: 1433) **{$MSSQL.USER},** **{$MSSQL.PASSWORD}** - Microsoft SQL login credentials (default: not set)	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). The "Service's TCP port state" item uses {HOST.CONN} and {$MSSQL.PORT} macros to check the availability of the Microsoft SQL instance.

Template	Mandatory macros	Additional steps/comments
Template DB MySQL by ODBC	**{$MYSQL.DSN}** - the system data source name (default: <Put your DSN here>) **{$MYSQL.USER}, {$MYSQL.PASSWORD}** - MySQL login credentials; password can be blank (default: not set)	To grant required privileges to MySQL user that will be used for monitoring, run: `GRANT USAGE,REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON %% *.* TO '<username>'@'%';%%` See MYSQL documentation for details.

Template	Mandatory macros	Additional steps/comments

Template DB 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:
		`CREATE USER zabbix_mon IDENTIFIED BY <PASSWORD>;`
		`-- Grant access to the zabbix_mon user.`
		`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:
		`[$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=$ORACLE_HOME/lib:/usr/lib64:/usr/lib:$OR`
		`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[bB]ack[0-9._]*$
Result is FALSE: ^NULL[0-9.]*$
Result is FALSE: ^[lL]o[0-9.]*$
Result is FALSE: ^[sS]ystem$
Result is FALSE: ^Nu[0-9.]*$
```

to filter out loopbacks and null interfaces on most systems.

Devices

List of device families for which templates are available:

Template name	Vendor	Device family	Known models	OS	MIBs used	**Tags**
Cisco Catalyst 3750<device model> SNMP	Cisco	Cisco Catalyst 3750	Cisco Catalyst 3750V2-24FS, Cisco Catalyst 3750V2-24PS, Cisco Catalyst 3750V2-24TS, Cisco Catalyst SNMP, Cisco Catalyst SNMP		CISCO-MEMORY-POOL-MIB, IF-MIB, EtherLike-MIB, SNMPv2-MIB, CISCO-PROCESS-MIB, CISCO-ENVMON-MIB, ENTITY-MIB	Certified
Template Net Alcatel Timetra TiMOS SNMP	Alcatel	Alcatel Timetra	ALCATEL SR 7750	TiMOS	TIMETRA-SYSTEM-MIB,TIMETRA-CHASSIS-MIB	Certified
Template Net Brocade FC SNMP	Brocade	Brocade FC switches	Brocade 300 SAN Switch-	-	SW-MIB,ENTITY-MIB	Performance, Fault

Template name	Vendor	Device family	Known models	OS	MIBs used	Tags
Template Net Brocade_Foundry Stackable SNMP	Brocade	Brocade ICX	Brocade ICX6610, Brocade ICX7250-48, Brocade ICX7450-48F		FOUNDRY-SN-AGENT-MIB, FOUNDRY-SN-STACKING-MIB	Certified
Template Net Brocade_Foundry Nonstackable SNMP	Brocade Foundry	Brocade MLX, Foundry	Brocade MLXe, Foundry FLS648, Foundry FWSX424		FOUNDRY-SN-AGENT-MIB	Performance, Fault
Template Net Cisco IOS SNMP	Cisco	Cisco IOS ver > 12.2 3.5	Cisco C2950	IOS	CISCO-PROCESS-MIB,CISCO-MEMORY-POOL-MIB,CISCO-ENVMON-MIB	Certified
Template Net Cisco releases later than 12.0_3_T and prior to 12.2_3.5_ SNMP	Cisco	Cisco IOS > 12.0 3 T and 12.2 3.5	-	IOS	CISCO-PROCESS-MIB,CISCO-MEMORY-POOL-MIB,CISCO-ENVMON-MIB	Certified
Template Net Cisco releases prior to 12.0_3_T SNMP	Cisco	Cisco IOS 12.0 3 T	-	IOS	OLD-CISCO-CPU-MIB,CISCO-MEMORY-POOL-MIB	Certified
Template Net D-Link DES_DGS Switch SNMP	D-Link	DES/DGX switches	D-Link DES-xxxx/DGS-xxxx,DLINK DGS-3420-26SC	-	DLINK-AGENT-MIB,EQUIPMENT-MIB,ENTITY-MIB	Certified
Template Net D-Link DES 7200 SNMP	D-Link	DES-7xxx	D-Link DES 7206	-	ENTITY-MIB,MY-SYSTEM-MIB,MY-PROCESS-MIB,MY-MEMORY-MIB	Performance Fault Interfaces
Template Net Dell Force S-Series SNMP	Dell	Dell Force S-Series	S4810		F10-S-SERIES-CHASSIS-MIB	Certified

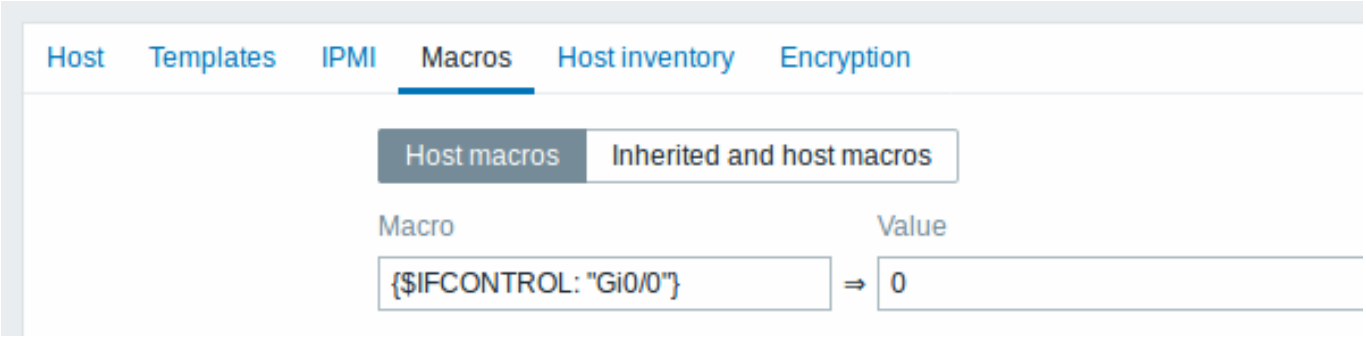
Template name	Vendor	Device family	Known models	OS	MIBs used	Tags
Template Net Extreme Exos SNMP	Extreme	Extreme EXOS	X670V-48x	EXOS	EXTREME-SYSTEM-MIB,EXTREME-SOFTWARE-MONITOR-MIB	Certified
Template Net Huawei VRP SNMP	Huawei	Huawei VRP	S2352P-EI	-	ENTITY-MIB,HUAWEI-ENTITY-EXTENT-MIB	Certified
Template Net Intel_Qlogic Infiniband SNMP	Intel/QLogic	Intel/QLogic Infiniband devices	Infiniband 12300		ICS-CHASSIS-MIB	Fault Inventory
Template Net Juniper SNMP	Juniper	MX,SRX,EX models	Juniper MX240, Juniper EX4200-24F	JunOS	JUNIPER-MIB	Certified
Template Net Mellanox SNMP	Mellanox	Mellanox Infiniband devices	SX1036	MLNX-OS	HOST-RESOURCES-MIB,ENTITY-MIB,ENTITY-SENSOR-MIB,MELLANOX-MIB	Certified
Template Net Mikrotik SNMP	Mikrotik	Mikrotik RouterOS devices	Mikrotik CCR1016-12G, Mikrotik RB2011UAS-2HnD, Mikrotik 912UAG-5HPnD, Mikrotik 941-2nD, Mikrotik 951G-2HnD, Mikrotik 1100AHx2	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	Certified
Template Net QTech QSW SNMP	QTech	Qtech devices	Qtech QSW-2800-28T	-	QTECH-MIB,ENTITY-MIB	Performance Inventory
Template Net Ubiquiti AirOS SNMP	Ubiquiti	Ubiquiti AirOS wireless devices	NanoBridge,NanoStation,Lit	AirOS	HOST-RESOURCES-MIB,IEEE802dot11-MIB	Performance
Template Net HP Comware HH3C SNMP	HP	HP (H3C) Comware	HP A5500-24G-4SFP HI Switch		HH3C-ENTITY-EXT-MIB,ENTITY-MIB	Certified

Template name	Vendor	Device family	Known models	OS	MIBs used	Tags
Template Net HP Enterprise Switch SNMP	HP	HP Enterprise Switch	HP ProCurve J4900B Switch 2626, HP J9728A 2920-48G Switch		STATISTICS-MIB,NETSWITCH-MIB,HP-ICF-CHASSIS,ENTITY-MIB,SEMI-MIB	Certified
Template Net TP-LINK SNMP	TP-LINK	TP-LINK	T2600G-28TS v2.0		TPLINK-SYSMONITOR-MIB,TPLINK-SYSINFO-MIB	Performance Inventory
Template Net Netgear Fastpath SNMP	Netgear	Netgear Fastpath	M5300-28G		FASTPATH-SWITCHING-MIB,FASTPATH-BOXSERVICES-PRIVATE-MIB	Fault Inventory

Template design

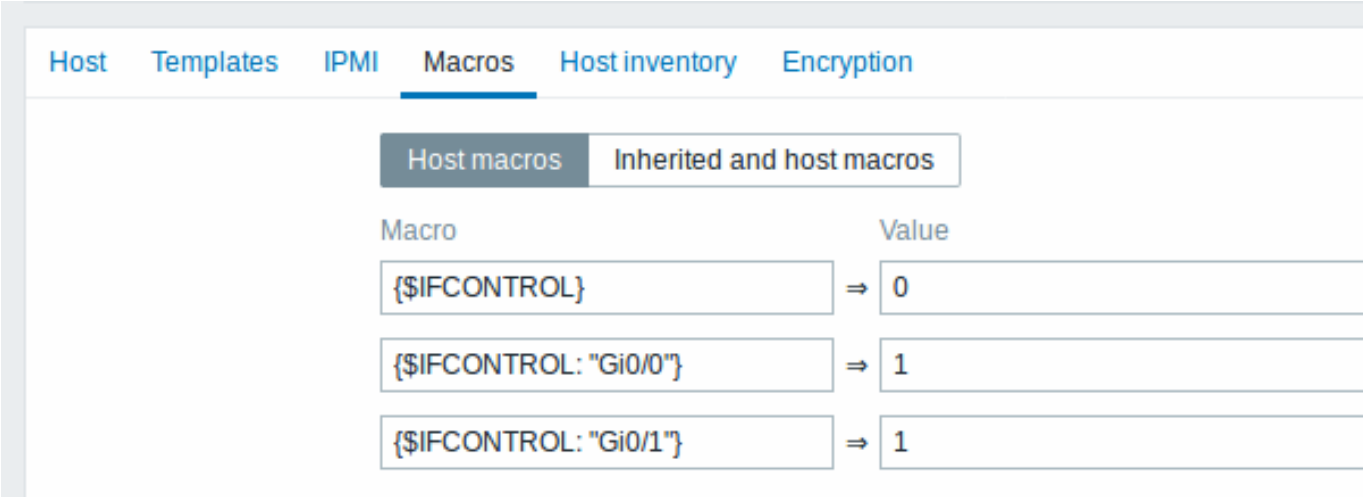
Templates were designed with the following in mind:

- 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 no want to monitor this condition for a specific interface create a user macro with context with the value 0. For example:



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 t
```



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 .xml 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.

Attention:
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.

Note:
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).

Template name	Mandatory macros	Additional steps/comments
SMART by Zabbix agent 2 / SMART by Zabbix agent 2 active	-	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.

Template name	Mandatory macros	Additional steps/comments
Systemd by Zabbix agent 2	-	No specific configuration is required.
Template App Ceph by Zabbix agent 2	**{$CEPH.API.KEY}** - the API key (default: `zabbix_pass`). Required, if {$CEPH.CONNSTRING} is a URI. Must be empty, if {$CEPH.CONNSTRING} is a session name. **{$CEPH.CONNSTRING}** - connection string; can be a session name or a URI defined in the following format: <protocol(host:port)>. For URI only HTTPS schema is supported. Examples: Prod, https://localhost:8003 (default) **{$CEPH.USER}** - user to be used for monitoring (default:`zabbix`). Required, if {$CEPH.CONNSTRING} is a URI. Must be empty, if {$CEPH.CONNSTRING} is a session name.	Works with *Ceph* plugin; named sessions are supported. 1. Configure the Ceph RESTful Module according to documentation. 2. Make sure a RESTful API endpoint is available for connection.
Template App Docker	-	Works with *Docker* plugin; named sessions are not supported. To set path to Docker API endpoint edit Plugins.Docker.Endpoint parameter in the agent 2 configuration file (default: `Plugins.Docker.Endpoint=unix:///var/run/docker.sock`) To test availability, run: `zabbix_get -s docker-host -k docker.info`
Template App Memcached	**{$MEMCACHED.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:11211. Examples: tcp://127.0.0.1:11211, tcp://localhost, unix:/var/run/memcached.sock.	Works with *Memcached* plugin; named sessions are supported. To test availability, run: `zabbix_get -s memcached-host -k memcached.ping`

Template name	Mandatory macros	Additional steps/comments
MongoDB cluster by Zabbix agent 2	**{$MONGODB.CONNSTRING}** - connection string in the URI format; password is not used (default: tcp://localhost:27017). Can be a session name or a URI defined in the following format: %% <protocol(host:port)>%% For URI only TCP scheme is supported. Examples: MongoDB1, tcp://172.16.0.10 **{$MONGODB.USER}**, **{$MONGODB.PASSWORD}** - MongoDB credentials (default: none). If not set and {$MONGODB.CONNSTRING} is a URI, parameters from the configuration file will be used. Must be empty, if {$MONGODB.CONNSTRING} is a session name.	Works with *MongoDB* plugin; named sessions are supported. For MongoDB configuration instructions, see plugins. To test availability, run: `zabbix_get -s mongos.node -k 'mongodb.ping["{$MONGODB.CONNSTRING}","{$MONGODB.USE`
MongoDB node by Zabbix agent 2	**{$MONGODB.CONNSTRING}** - 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: %% <protocol(host:port)>%% For URI only TCP scheme is supported. Examples: MongoDB1, tcp://172.16.0.10 **{$MONGODB.USER}**, **{$MONGODB.PASSWORD}** - MongoDB credentials (default: none). If not set and {$MONGODB.CONNSTRING} is a URI, parameters from the configuration file will be used. Must be empty, if {$MONGODB.CONNSTRING} is a session name.	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["{$MONGODB.CONNSTRING}","{$MONGODB.USE`

Template name	Mandatory macros	Additional steps/comments
Template DB MySQL by Zabbix agent 2	**{$MYSQL.DSN}** - the system data source name of the MySQL instance (default: <Put your DSN>). Can be a session name or a URI defined in the following format: %% <protocol(host:port or /path/to/socket)/>%% For URI only TCP and Unix schemas are supported. Examples: MySQL1, tcp://localhost:3306, tcp://172.16.0.10, unix:/var/run/mysql.sock **{$MYSQL.USER}**, **{$MYSQL.PASSWORD}** - MySQL credentials (default: none). Required, if {$MYSQL.DSN} is a URI. Must be empty, if {$MYSQL.DSN} is a session name.	Works with *MySQL* plugin; named sessions are supported. To grant required privileges to a MySQL user that will be used for monitoring, run: `GRANT USAGE,REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON *.* TO '<username>'@'%';` See MySQL documentation for information about user privileges and Unix sockets.

Template name	Mandatory macros	Additional steps/comments

Template DB 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 (default)
	
	**{$ORACLE.SERVICE}** - Oracle Service name (default: ORA). Required, if {$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;`

Template DB PostgreSQL by Zabbix agent 2	**{$PG.URI}** - connection string; can be a session name or a URI defined in the following format: %% <protocol(host:port or /path/to/socket)/>%%. For URI only TCP and Unix schemas are supported.
	Examples: Postgres1, tcp://localhost:5432 (default), tcp://172.16.0.10
	{$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).

Template name	Mandatory macros	Additional steps/comments
Template DB Redis	**{$REDIS.CONN.URI}** - connection string in the URI format; port is optional; password is not used. If not set, the plugin's default value is used: tcp://localhost:6379	Works with *Redis* plugin; named sessions are supported. To test availability, run: `zabbix_get -s redis-master -k redis.ping`
Website certificate by Zabbix agent 2	** {$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.

Note:
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).

Template name	Mandatory macros	Additional steps/comments
Microsoft Exchange Server 2016 by Zabbix agent/Microsoft Exchange Server 2016 by Zabbix agent active		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.

Template App 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:
		`httpd -M 2>/dev/null | grep status_module`
		
		Apache configuration example:
		`<Location "/server-status">`
		`SetHandler server-status`
		`Require host example.com`
		`</Location>`

Template App 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).

Template name	Mandatory macros	Additional steps/comments
Template App IIS by Zabbix agent / Template App IIS by Zabbix agent active	**{$IIS.PORT}** - the port IIS Server listens on (default: 80) **{$IIS.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.
Template App Nginx by Zabbix agent	**{$NGINX.STUB_STATUS.HOST}** - the hostname or IP address of Nginx stub_status host or container (default: localhost) **{$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`

Template App 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`

Template App 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.

Template name	Mandatory macros	Additional steps/comments

Template DB MySQL	**{$MYSQL.HOST}** - the hostname or IP address of MySQL host or container (default: 127.0.0.1 (since 5.0.27)/localhost (before 5.0.27))
	{$MYSQL.PORT} - the database service port (default: 3306)
		1. If necessary, add the path to the mysql and mysqladmin 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>’ ”

Template DB 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>`

9 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

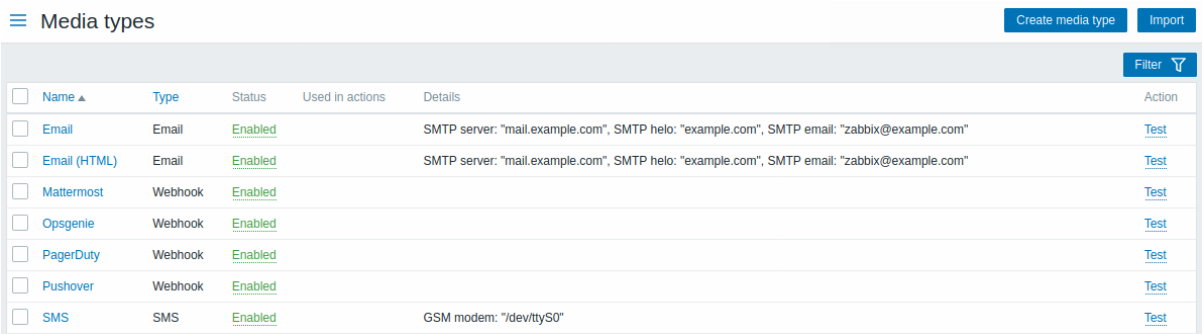
Overview

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*.



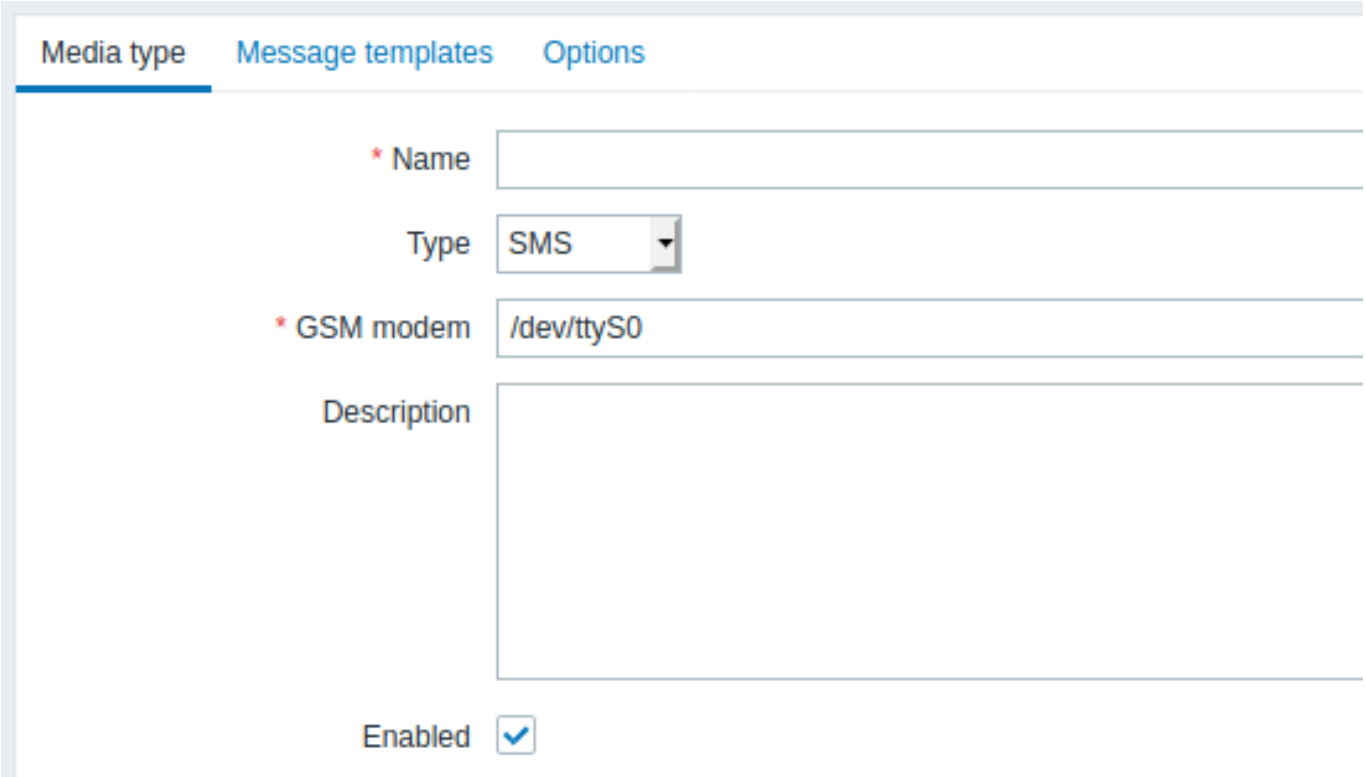
Some media types come pre-defined in the default dataset. You just need to finetune their parameters to get them working.

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.



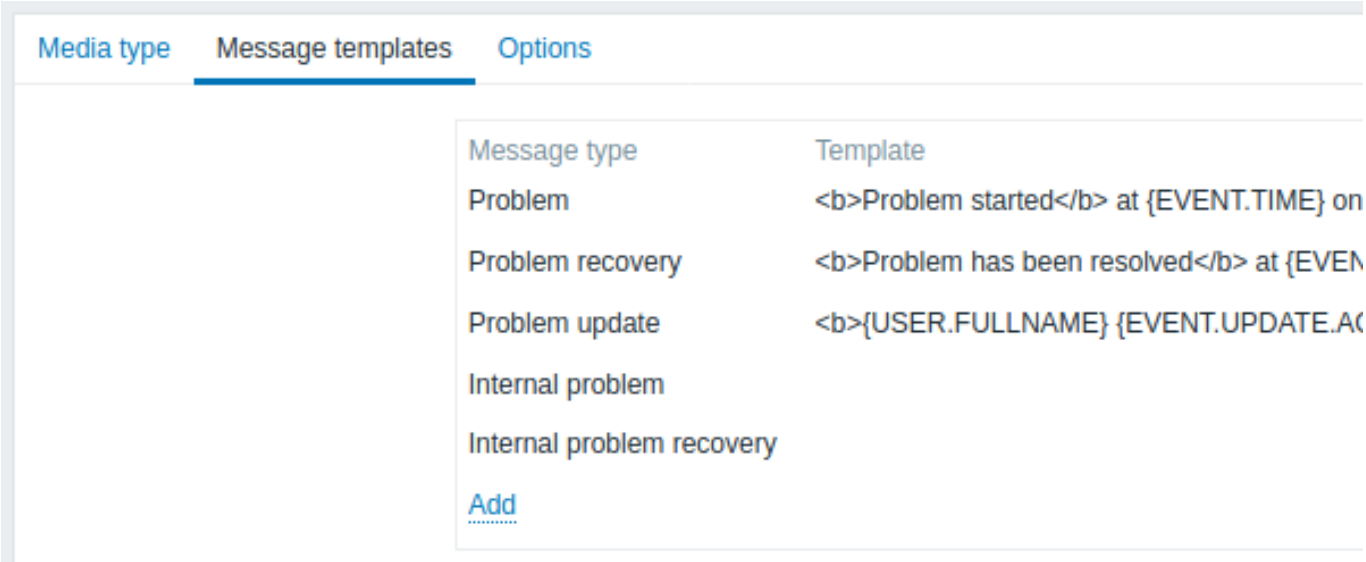
In the **Media type** tab the common general attributes are:

Parameter	Description
Name	Name of the media type.
Type	Select the type of media.
Description	Enter a description.
Enabled	Mark the checkbox to enable the media type.

See the individual pages of media types for media-specific parameters.

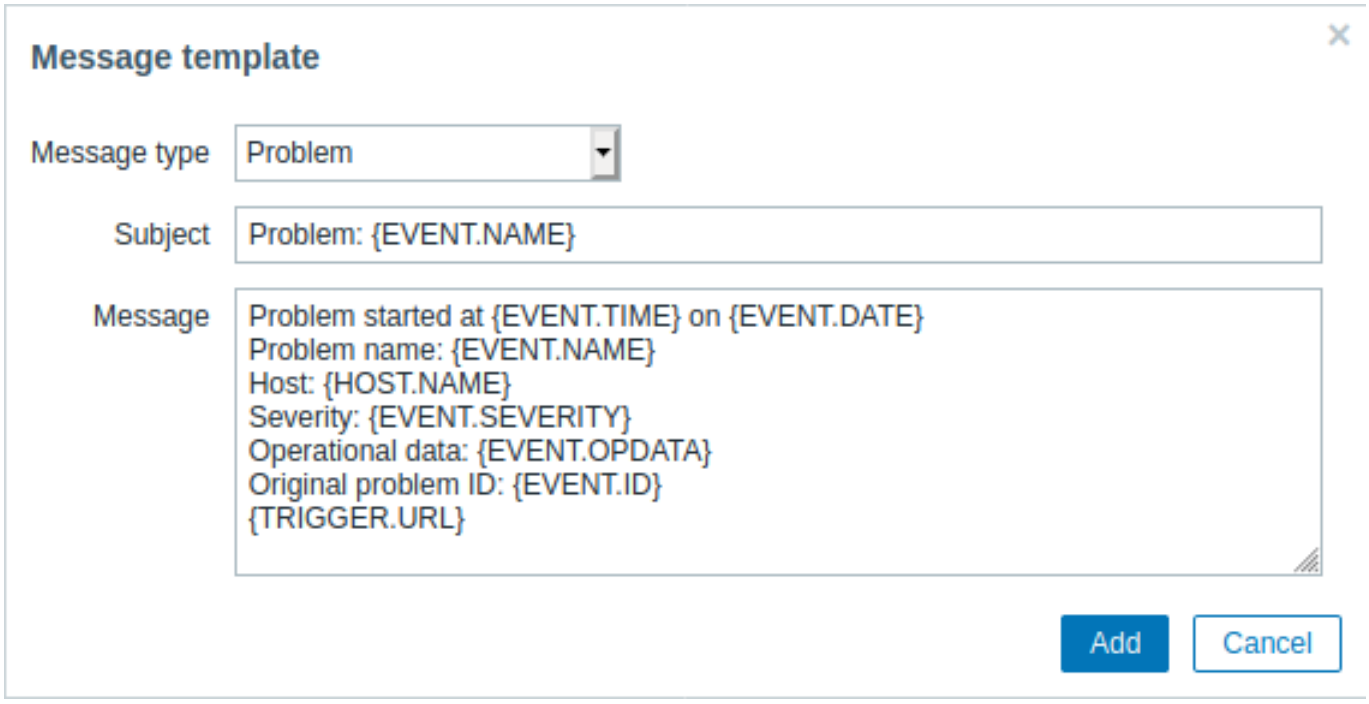
The **Message templates** tab allows to set default notification messages for all or some of the following event types:

- Problem
- Problem recovery
- Problem update
- Discovery
- Autoregistration
- Internal problem
- Internal problem recovery



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:

Parameter	Description
Message type	Type of event for which default message will be defined. Only one default message can be defined for each event type.
Subject	Subject of the default message. The subject may contain macros. It is limited to 255 characters. Subject is not available for SMS media type.
Message	The default message. The message may contain macros. It is limited to certain amount of characters depending on the type of database (see Sending messages for more information).

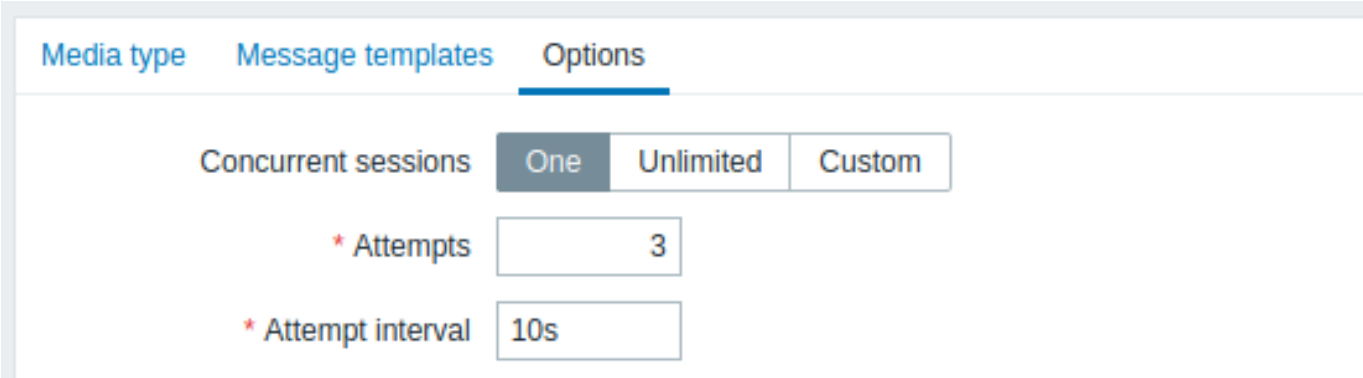
To make changes to an existing message template: In the *Actions* column click on Edit to edit the template or click on Remove to delete the message template.

A notification message template can also be defined for each action individually (see action operations for details). Custom messages in actions can override message templates configured for a media type.

Warning:
Message templates must be defined for all media types, including webhooks or custom alert scripts that are not using default messages for notifications. For example, an action "Send message to Pushover webhook" will fail to send problem notifications, if problem message has not been defined for the Pushover webhook.

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 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; 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

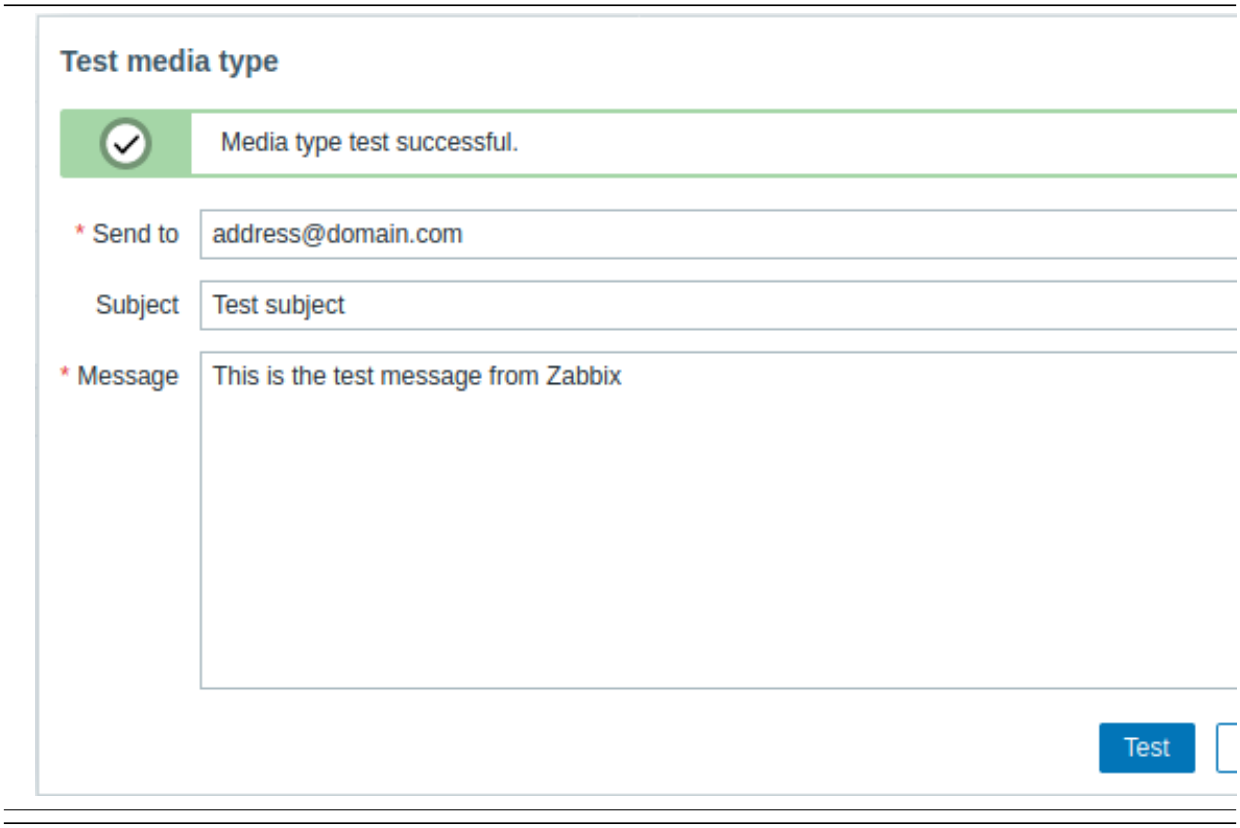
It is possible to test if a configured media type works.

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 recipient address in the *Send to* field and specify test notification subject (optional) and message.
- Send a test message by clicking on *Test*

Test success or failure message will be displayed in the same window:

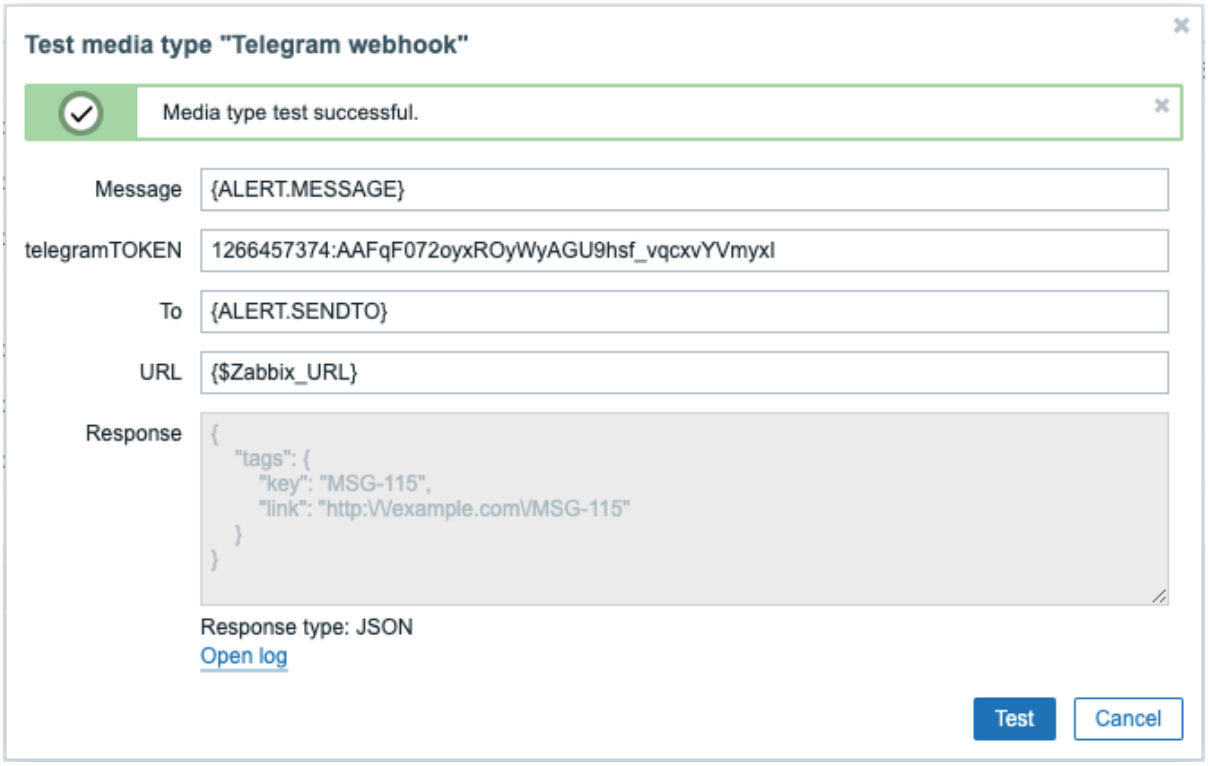


**Webhook **

To test a webhook media type:

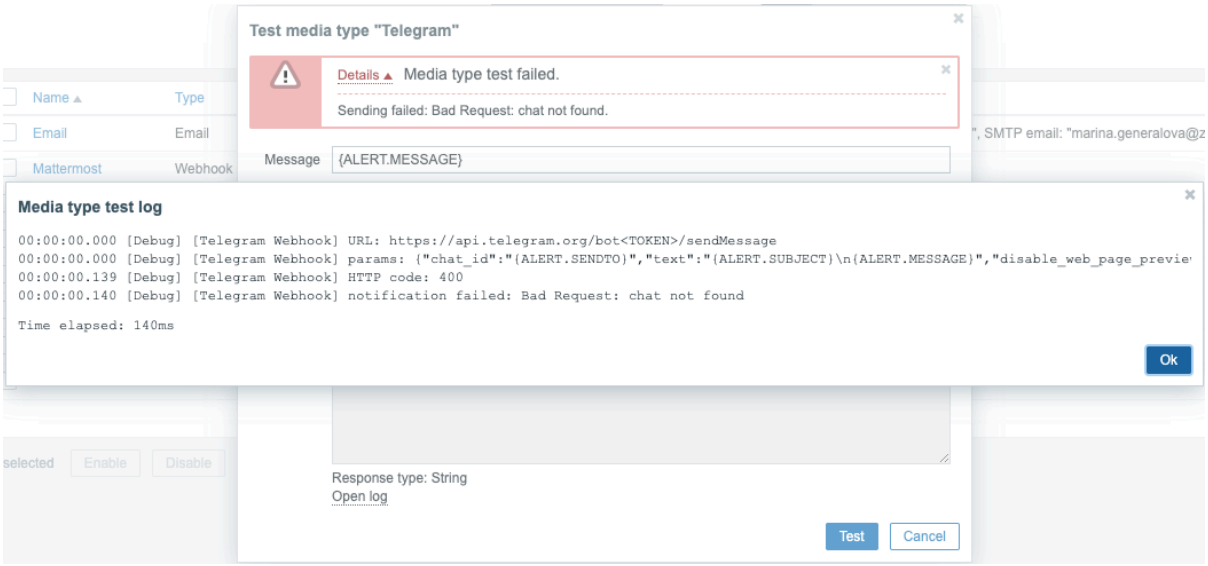
- Locate the relevant webhook in the list of media types
- Click on *Test* in the last column of the list (a testing window will open)
- Edit the webhook parameter values, if needed
- Click on *Test*

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

User will receive media type notifications, only if a media (e-mail address / phone number / webhook user ID / etc) for this media type is defined in the user profile. For example, the action sending messages to user "Admin" using webhook "X" will always fail to send messages if webhook "X" media is not defined in the user profile.

To define user media:

- Go to *Administration* → *Users*
- Open the user properties form
- In the Media tab, click on Add

User media attributes:

Parameter	Description
Type	The drop-down list contains names of all configured media types.
Send to	Provide required contact information where to send messages. 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@nycdatacenter.com>* and *manager@nycdatacenter.com* are both valid formats. Incorrect examples: *John Doe zabbix@company.com*, *%%"Zabbix\@\<H(comment)Q\>" 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). See the Time period specification page for description of the format.
Use if severity	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.
Status	Status of the user media. **Enabled** - is in use. **Disabled** - is not being used.

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.

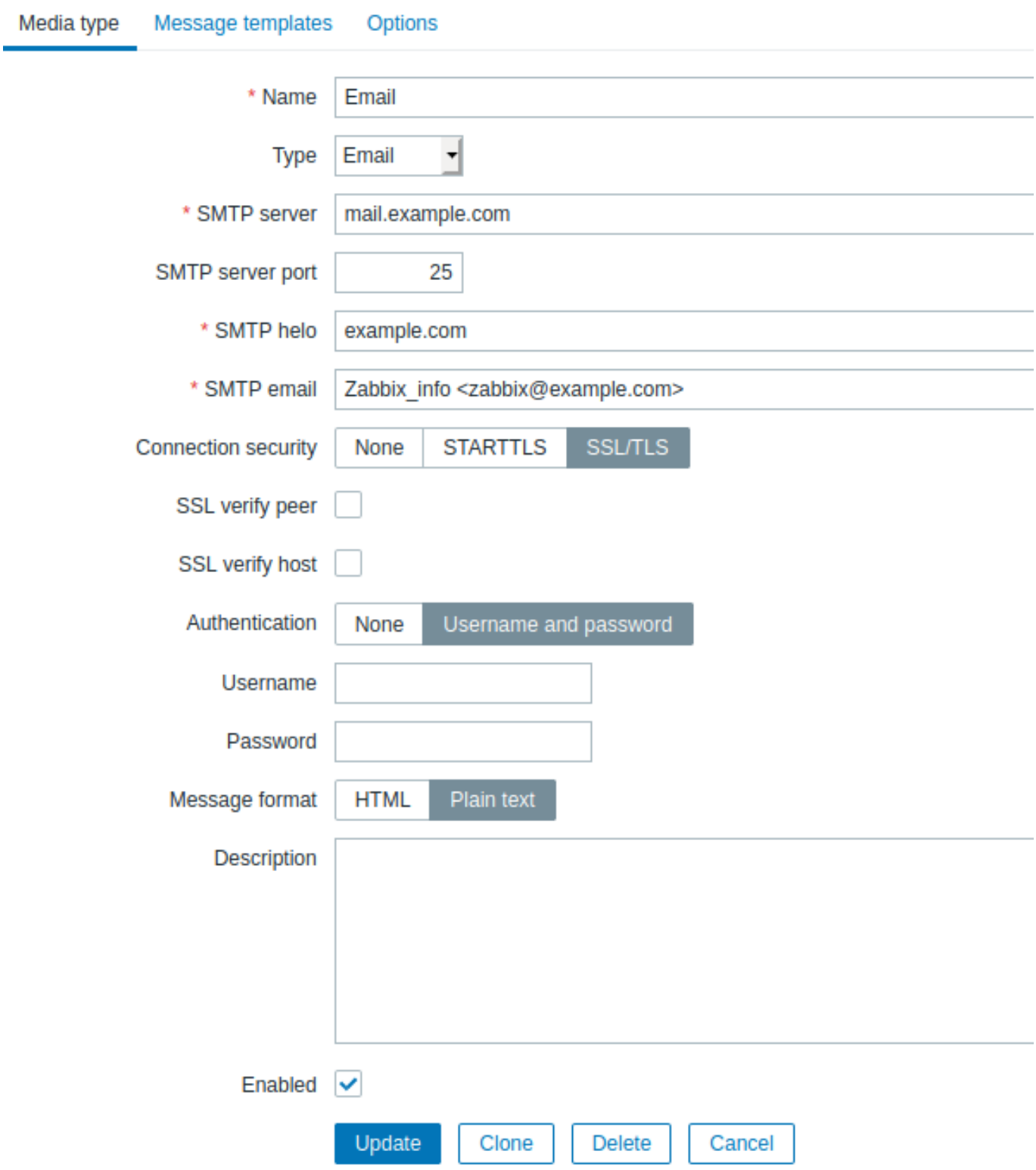
> **Note:**
> Multiple notifications for single event will be grouped together on the same email thread.

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:

Parameter	Description
SMTP server	Set an SMTP server to handle outgoing messages.
SMTP server port	Set the SMTP server port to handle outgoing messages. This option is supported *starting with Zabbix 3.0*.
SMTP helo	Set a correct SMTP helo value, normally a domain name.

Parameter	Description
SMTP email	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 <zabbix@company.com>* 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: *zabbix@company.com* (only email address, no need to use angle brackets) *Zabbix_info <zabbix@company.com>* (display name and email address in angle brackets) *∑Ω-monitoring <zabbix@company.com>* (UTF-8 characters in display name) Invalid examples: *Zabbix HQ zabbix@company.com* (display name present but no angle brackets around email address) *”Zabbix\@\<H(comment)Q\>” <zabbix@company.com>* (although valid by RFC 5322, quoted pairs and comments are not supported in Zabbix emails)
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 ”SSLCALocation” 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 (since 3.4.2) **Username and password** - implies ”AUTH=*” leaving the choice of authentication mechanism to cURL (until 3.4.2) **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

Attention:
To enable SMTP authentication options, Zabbix server must be both compiled with the `--with-libcurl` compilation option (with cURL 7.20.0 or higher) and use the `libcurl-full` packages during runtime.

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:

Parameter	Description
GSM modem	Set the serial device name of the GSM modem.

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 alertscripts

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 must be located in the directory specified in the server configuration file `AlertScriptsPath` parameter.

Here is an example alert script:

```
#####!/bin/bash

to=$1
subject=$2
body=$3

cat <<EOF | mail -s "$subject" "$to"
$body
EOF
```

Attention:
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.

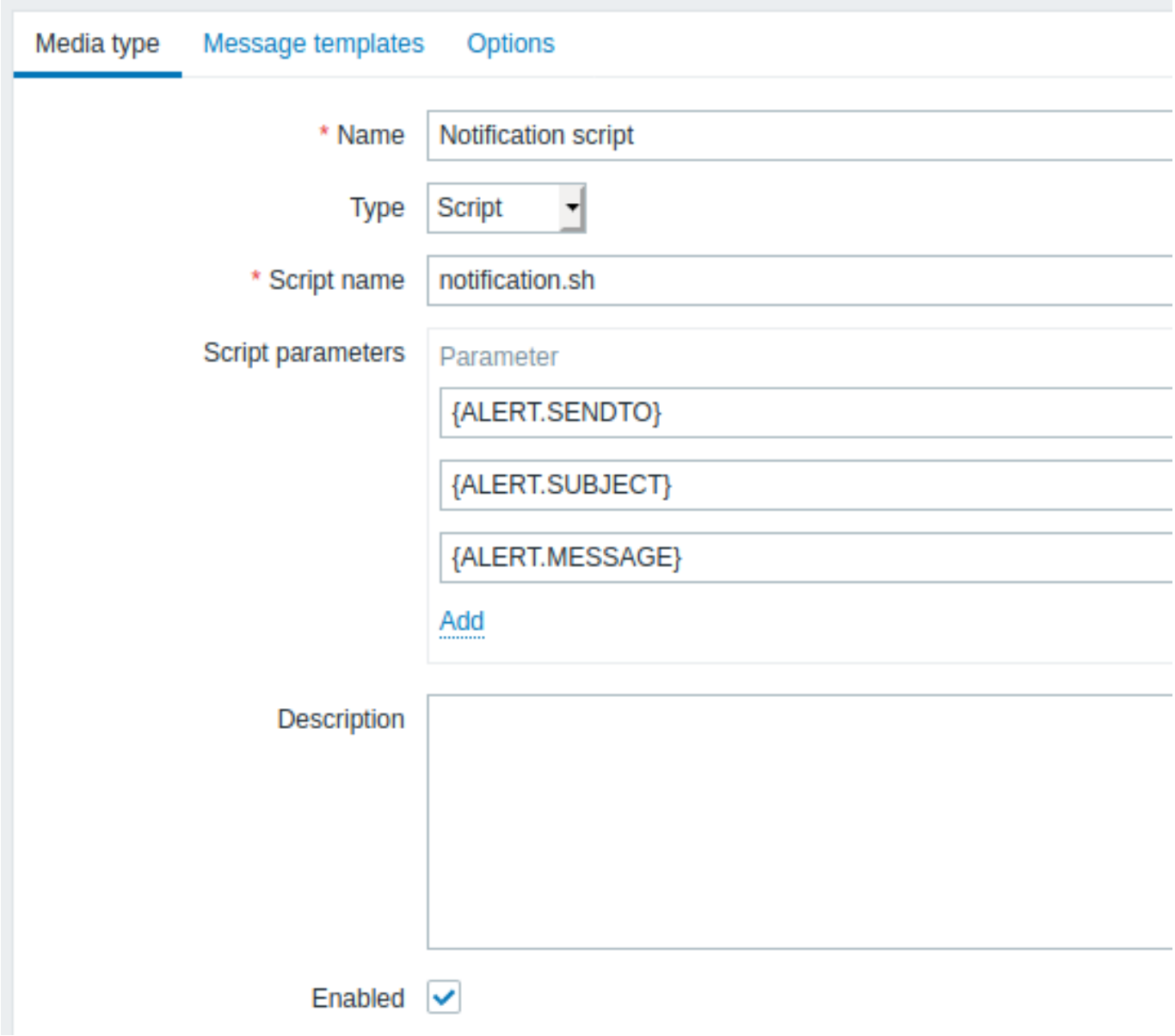
Environment variables are not preserved or created for the script, so they should be handled explicitly.

Configuration

To configure custom alertscripts 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:

Parameter	Description
Script name	Enter the name of the script file (e.g., notification.sh) that is located in the directory specified in the server configuration file `AlertScriptsPath` parameter.
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.

Warning:
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.

Attention:
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
- 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

Note:
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

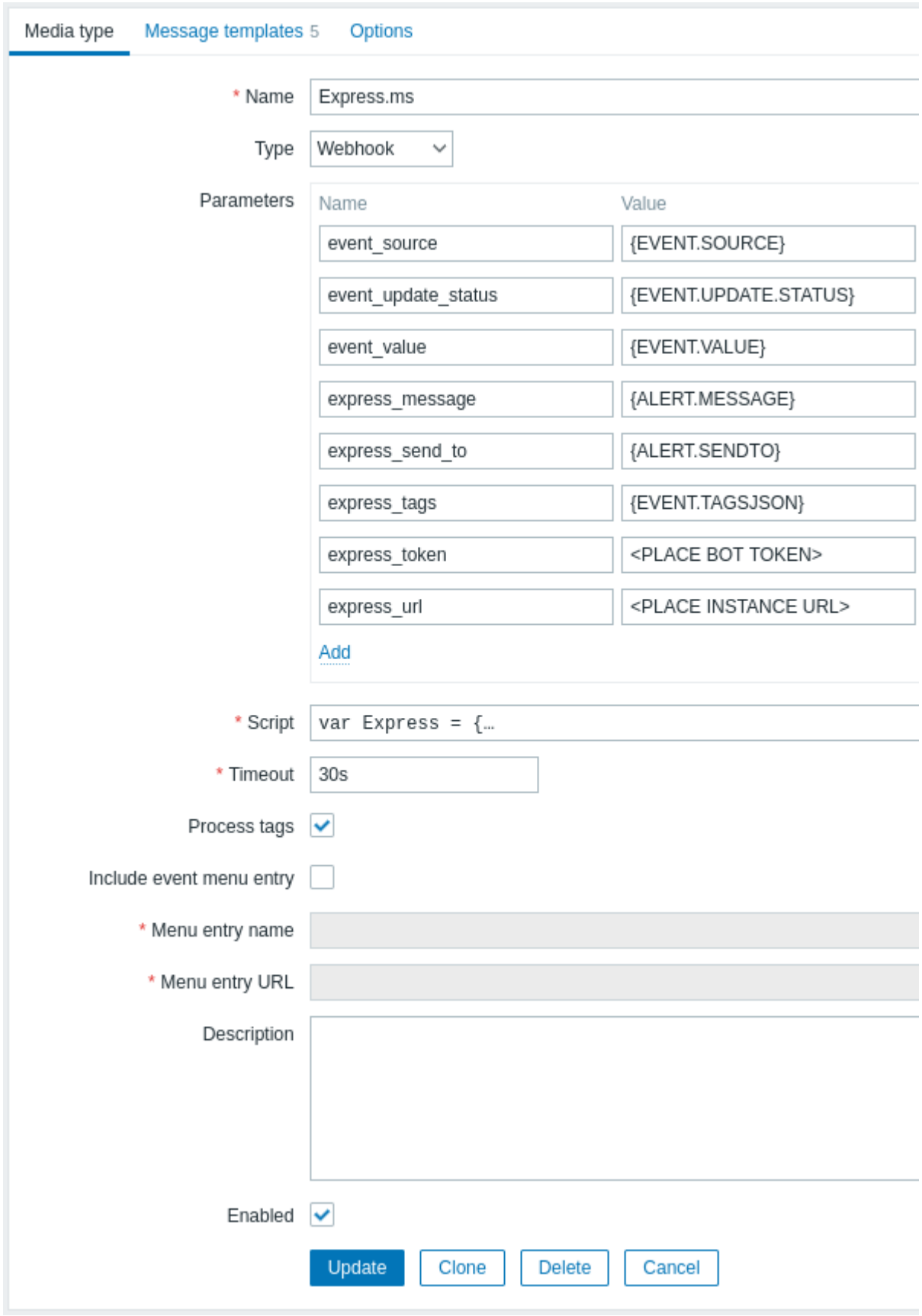
To start using a webhook integration:

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:

Parameter	Description
Parameters	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:<empty>, HTTPProxy:<empty>, To:{ALERT.SENDTO}, Subject:{ALERT.SUBJECT}, Message:{ALERT.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).
Script	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. *Note*: Using local variables instead of global ones is recommended to make sure that each script operates on its own data and that there are no collisions between simultaneous calls (see known issues). See also: Webhook development guidelines, Webhook script examples, Additional JavaScript objects.
Timeout	JavaScript execution timeout (1-60s, default 30s). Time suffixes are supported, e.g. 30s, 1m.
Process tags	Mark the checkbox to process returned JSON property values as tags. These tags are added to any existing problem tags. Note that when using webhook tags, the webhook must return a JSON object containing at least an empty tags object: `var result = {tags: {}};` Examples of tags that can be returned: *Jira ID: PROD-1234*, *Responsible: John Smith*, *Processed:<no value>*

Parameter	Description
Include event menu entry	Mark the checkbox to include an entry in the event menu linking to a created external ticket. An entry will be included for each webhook that is enabled and has this checkbox marked. Note that if the *Menu entry name* and *Menu entry URL* parameters contain any {EVENT.TAGS.<tag name>} macros, an entry will be included only if these macros can be resolved (that is, the event has these tags defined). If marked, the webhook should not be used for sending notifications to different users (consider creating a dedicated user instead) and should not be used in multiple alert actions for a single problem event.
Menu entry name	Specify the menu entry name. {EVENT.TAGS.<tag name>} macro is supported. This field is only mandatory if *Include event menu entry* is marked.
Menu entry URL	Specify the underlying URL of the menu entry. {EVENT.TAGS.<tag name>} macro is supported. This field is only mandatory if *Include event menu entry* is marked.

See common media type parameters for details on how to configure default messages and alert processing options.

Warning:
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 an alias *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:

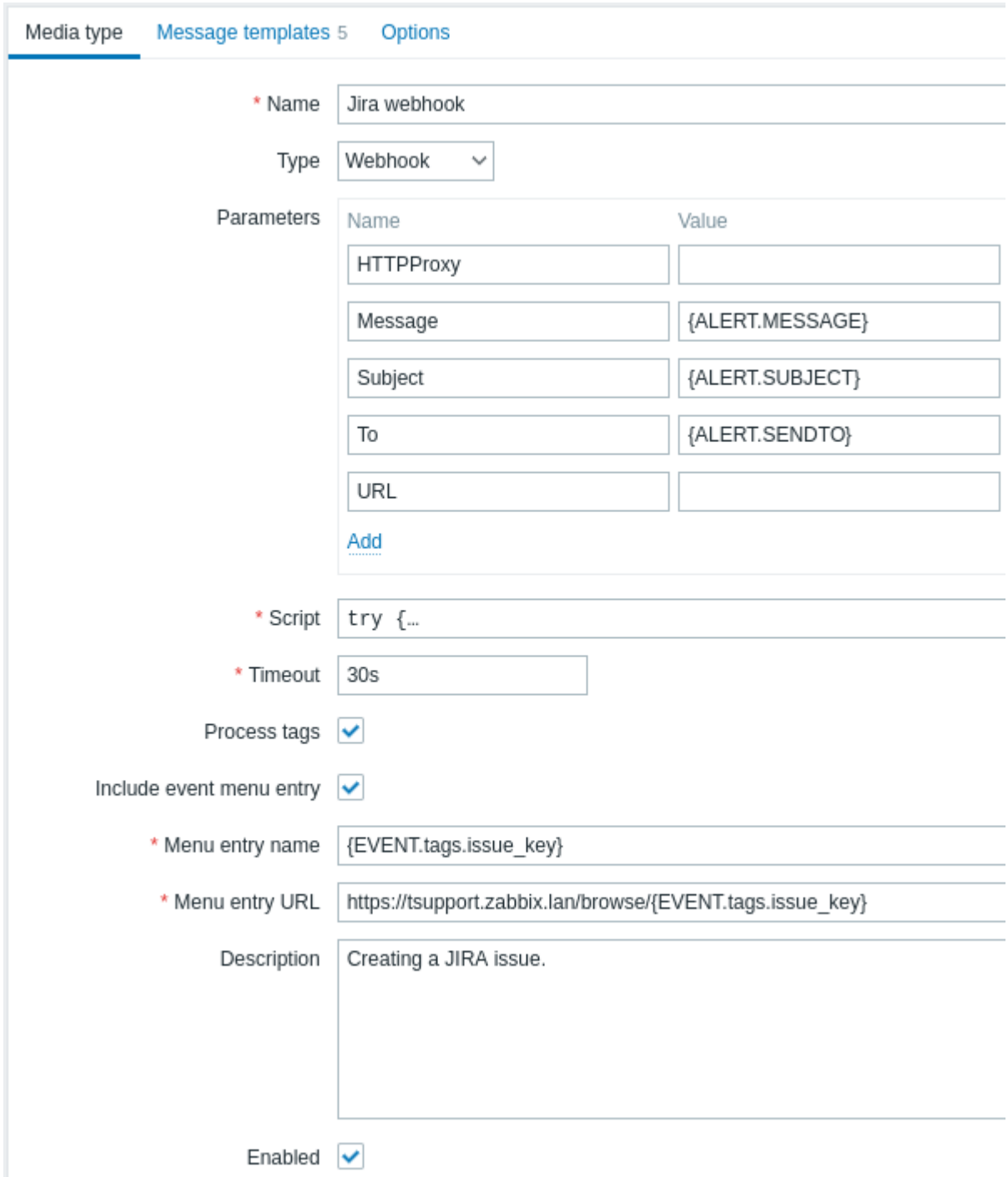
- If a webhook uses webhook tags to store ticket\message ID and handle update\resolve operations, avoid using the same webhook in multiple alert actions for a single problem event. If {EVENT.TAGS.<tag name>} exists and gets updated in the webhook, its resulting value will be undefined. To avoid this, use a new tag name in the webhook for storing updated values. This applies to Jira, Jira Service Desk, Mattermost, Opsgenie, OTRS, Redmine, ServiceNow, Slack, Zammad, and Zendesk webhooks provided by Zabbix and to most webhooks utilizing the *Include event menu entry* option. Note, however, that a single webhook can be used in multiple operations or escalation steps of the same action, as well as in different actions that will not be triggered by the same problem event due to different conditions.
- When using a webhook in actions for internal events, ensure to mark the *Custom message* checkbox and define a custom message in the action operation configuration. 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)



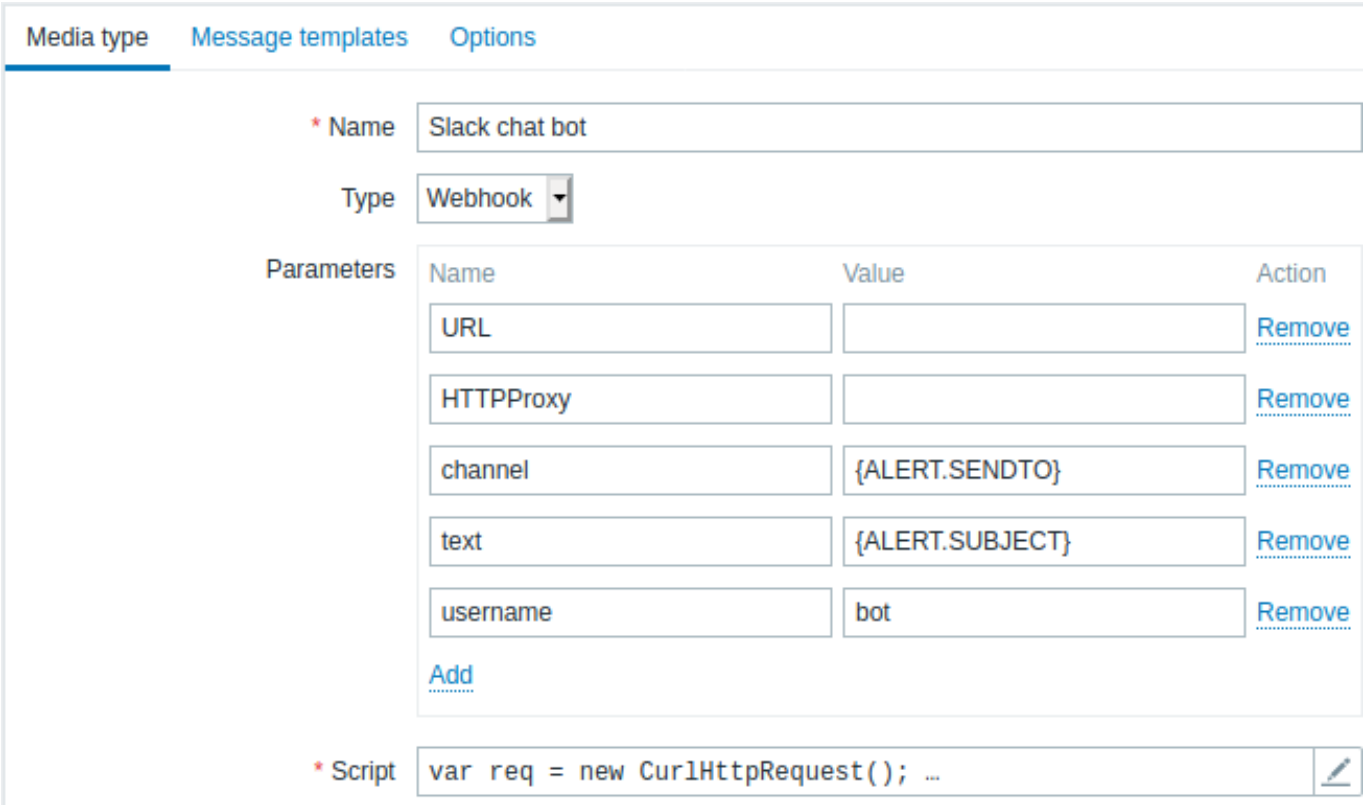
This script will create a JIRA issue and return some info on the created issue.

```
try {
    Zabbix.Log(4, '[ Jira webhook ] Started with params: ' + value);

    var result = {
            'tags': {
                'endpoint': 'jira'
            }
        },
        params = JSON.parse(value),
        req = new CurlHttpRequest(),
        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://jira.example.com/rest/api/2/issue/',
        JSON.stringify({"fields": fields})
    );

    if (req.Status() != 201) {
        throw 'Response code: ' + req.Status();
    }

    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.



```
try {
    var params = JSON.parse(value),
        req = new CurlHttpRequest(),
        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.Status();
        }
        else {
            throw 'Request success, but response parsing failed.';
        }
    }

    if (req.Status() !== 200 || !response.ok || response.ok === 'false') {
        throw response.error;
    }

    return 'OK';
}
catch (error) {
    Zabbix.Log(3, '[ Slack 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:

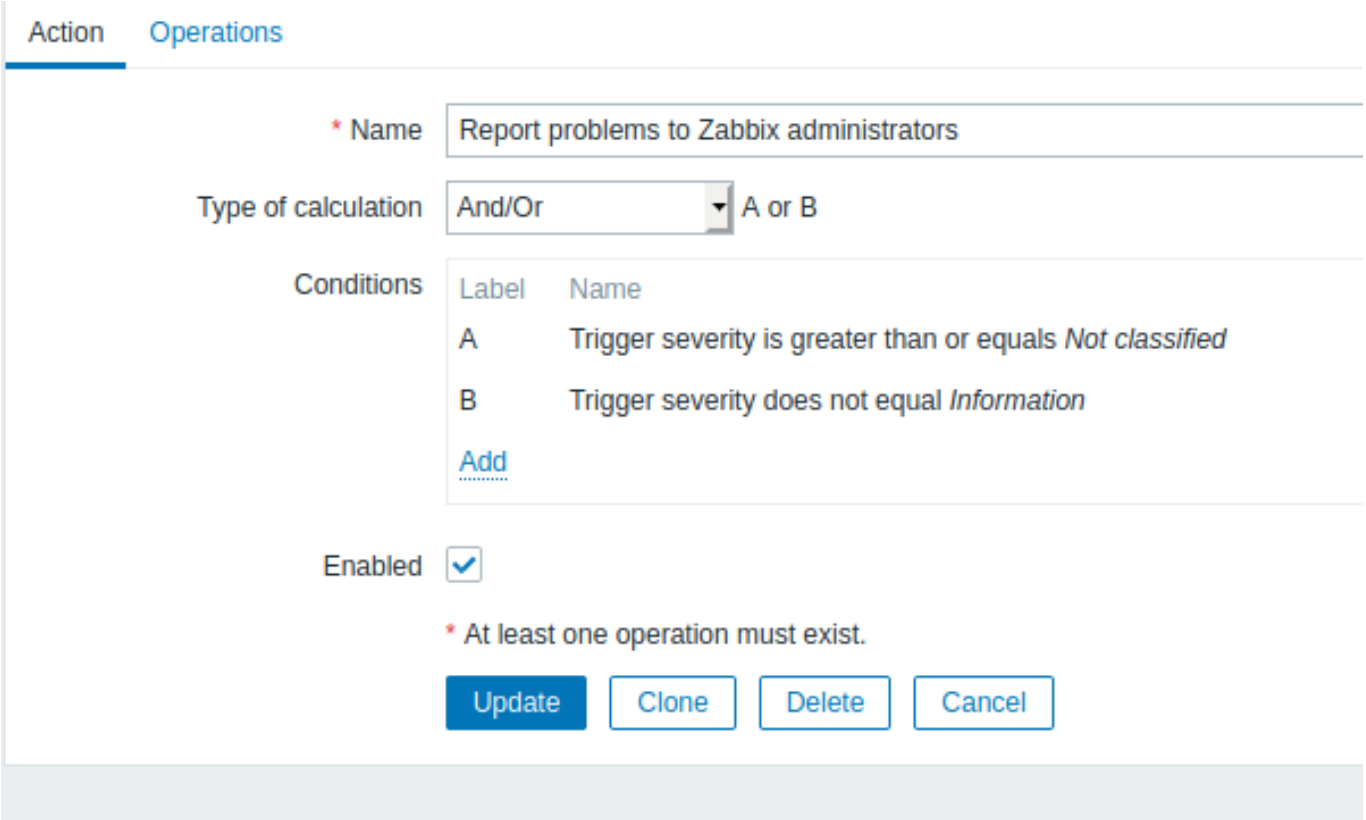
- Trigger actions - for events when trigger status changes from *OK* to *PROBLEM* and back
- Discovery actions - for events when network discovery takes place
- Autoregistration actions - for events when new active agents auto-register (or host metadata changes for registered ones)
- Internal actions - 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*
- Select action type (Trigger, Discovery, Autoregistration, Internal) from the title dropdown
- Click on *Create action*
- Name the action
- Choose conditions upon which operations are carried out
- Choose the operations to carry out

General action attributes:



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Unique action name.
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. If no conditions are configured, the action will run for every event that corresponds to the action type being configured.
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 set for trigger-based actions:

Condition type	Supported operators	Description
Application	equals contains does not contain	Specify an application or an application to exclude. **equals** - event belongs to a trigger of the item that is linked to the specified application. **contains** - event belongs to a trigger of the item that is linked to an application containing the string. **does not contain** - event belongs to a trigger of the item that is linked to an application not containing the string.
Host group	equals does not equal	Specify host groups or host groups to exclude. **equals** - event belongs to this host group. **does not equal** - event does not belong to this host group. Specifying a parent host group implicitly selects all nested host groups. To specify the parent group only, all nested groups have to be additionally set with the **does not equal** operator.
Template	equals does not equal	Specify templates or templates to exclude. **equals** - event belongs to a trigger inherited from this template. **does not equal** - event does not belong to a trigger inherited from this template.
Host	equals does not equal	Specify hosts or hosts to exclude. **equals** - event belongs to this host. **does not equal** - event does not belong to this host.
Tag name	equals does not equal contains does not contain	Specify event tag or event tag to exclude. **equals** - event has this tag **does not equal** - event does not have this tag **contains** - event has a tag containing this string **does not contain** - event does not have a tag containing this string

Condition type	Supported operators	Description
Tag value	equals does not equal contains does not contain	Specify event tag and value combination or tag and value combination to exclude. **equals** - event has this tag and value **does not equal** - event does not have this tag and value **contains** - event has a tag and value containing these strings **does not contain** - event does not have a tag and value containing these strings
Trigger	equals does not equal	Specify triggers or triggers to exclude. **equals** - event is generated by this trigger. **does not equal** - event is generated by any other trigger, except this one.
Trigger name	contains does not contain	Specify a string in the trigger name or a string to exclude. **contains** - event is generated by a trigger, containing this string in the name. **does not contain** - this string cannot be found in the trigger name. *Note*: Entered value will be compared to trigger name with all macros expanded.
Trigger severity	equals does not equal is greater than or equals is less than or equals	Specify trigger severity. **equals** - equal to trigger severity **does not equal** - not equal to trigger severity **is greater than or equals** - more or equal to trigger severity **is less than or equals** - less or equal to trigger severity
Time period	in not in	Specify a time period or a time period to exclude. **in** - event time is within the time period. **not in** - event time is not within the time period. See the time period specification page for description of the format. User macros are supported, since Zabbix 3.4.0.

Condition type	Supported operators	Description
Problem is suppressed	no yes	Specify if the problem is suppressed (not shown) because of host maintenance. **no** - problem is not suppressed. **yes** - problem is suppressed.

Discovery actions

The following conditions can be set for discovery-based events:

Condition type	Supported operators	Description
Host IP	equals does not equal	Specify an IP address range or a range to exclude for a discovered host. **equals** - host IP is in the range. **does not equal** - host IP is not in the range. It may have the following formats: Single IP: 192.168.1.33 Range of IP addresses: 192.168.1-10.1-254 IP mask: 192.168.4.0/24 List: 192.168.1.1-254, 192.168.2.1-100, 192.168.2.200, 192.168.4.0/24 Support for spaces in the list format is provided since Zabbix 3.0.0.
Service type	equals does not equal	Specify a service type of a discovered service or a service type to exclude. **equals** - matches the discovered service. **does not equal** - does not match the discovered service. 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)*.
Service port	equals does not equal	Specify a TCP port range of a discovered service or a range to exclude. **equals** - service port is in the range. **does not equal** - service port is not in the range.

Condition type	Supported operators	Description
Discovery rule	equals does not equal	Specify a discovery rule or a discovery rule to exclude. **equals** - using this discovery rule. **does not equal** - using any other discovery rule, except this one.
Discovery check	equals does not equal	Specify a discovery check or a discovery check to exclude. **equals** - using this discovery check. **does not equal** - using any other discovery check, except this one.
Discovery object	equals	Specify the discovered object. **equals** - equal to discovered object (a device or a service).
Discovery status	equals	**Up** - matches 'Host Up' and 'Service Up' events **Down** - matches 'Host Down' and 'Service Down' events **Discovered** - matches 'Host Discovered' and 'Service Discovered' events **Lost** - matches 'Host Lost' and 'Service Lost' events
Uptime/Downtime	is greater than or equals is less than or equals	Uptime for 'Host Up' and 'Service Up' events. Downtime for 'Host Down' and 'Service Down' events. **is greater than or equals** - is more or equal to. Parameter is given in seconds. **is less than or equals** - is less or equal to. Parameter is given in seconds.

Condition type	Supported operators	Description
Received value	equals does not equal is greater than or equals is less than or equals contains does not contain	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). **equals** - equal to the value. **does not equal** - not equal to the value. **is greater than or equals** - more or equal to the value. **is less than or equals** - less or equal to the value. **contains** - contains the substring. Parameter is given as a string. **does not contain** - does not contain the substring. Parameter is given as a string.
Proxy	equals does not equal	Specify a proxy or a proxy to exclude. **equals** - using this proxy. **does not equal** - using any other proxy except this one.

Note:
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 set for actions based on active agent autoregistration:

Condition type	Supported operators	Description
Host metadata	contains does not contain matches does not match	Specify host metadata or host metadata to exclude. **contains** - host metadata contains the string. **does not contain** - host metadata does not contain the string. Host metadata can be specified in an agent configuration file. **matches** - host metadata matches regular expression. **does not match** - host metadata does not match regular expression.

Condition type	Supported operators	Description
Host name	contains does not contain matches does not match	Specify a host name or a host name to exclude. **contains** - host name contains the string. **does not contain** - host name does not contain the string. **matches** - host name matches regular expression. **does not match** - host name does not match regular expression.
Proxy	equals does not equal	Specify a proxy or a proxy to exclude. **equals** - using this proxy. **does not equal** - using any other proxy except this one.

Internal event actions

The following conditions can be set for actions based on internal events:

Condition type	Supported operators	Description
Application	equals contains does not contain	Specify an application or an application to exclude. **equals** - event belongs to an item that is linked to the specified application. **contains** - event belongs to an item that is linked to an application containing the string. **does not contain** - event belongs to an item that is linked to an application not containing the string.
Event type	equals	**Item in ”not supported” state** - matches events where an item goes from a ’normal’ to ’not supported’ state **Low-level discovery rule in ”not supported” state** - matches events where a low-level discovery rule goes from a ’normal’ to ’not supported’ state **Trigger in ”unknown” state** - matches events where a trigger goes from a ’normal’ to ’unknown’ state
Host group	equals does not equal	Specify host groups or host groups to exclude. **equals** - event belongs to this host group. **does not equal** - event does not belong to this host group.

Condition type	Supported operators	Description
Template	equals does not equal	Specify templates or templates to exclude. **equals** - event belongs to an item/trigger/low-level discovery rule inherited from this template. **does not equal** - event does not belong to an item/trigger/low-level discovery rule inherited from this template.
Host	equals does not equal	Specify hosts or hosts to exclude. **equals** - event belongs to this host. **does not equal** - event does not belong to this host.

Type of calculation

The following options of calculating conditions are available:

- **And** - all conditions must be met

Note that using "And" calculation is disallowed between several triggers when they are selected as a `Trigger=` condition. Actions can only be executed based on the event of one trigger.

- **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 (including IPMI)

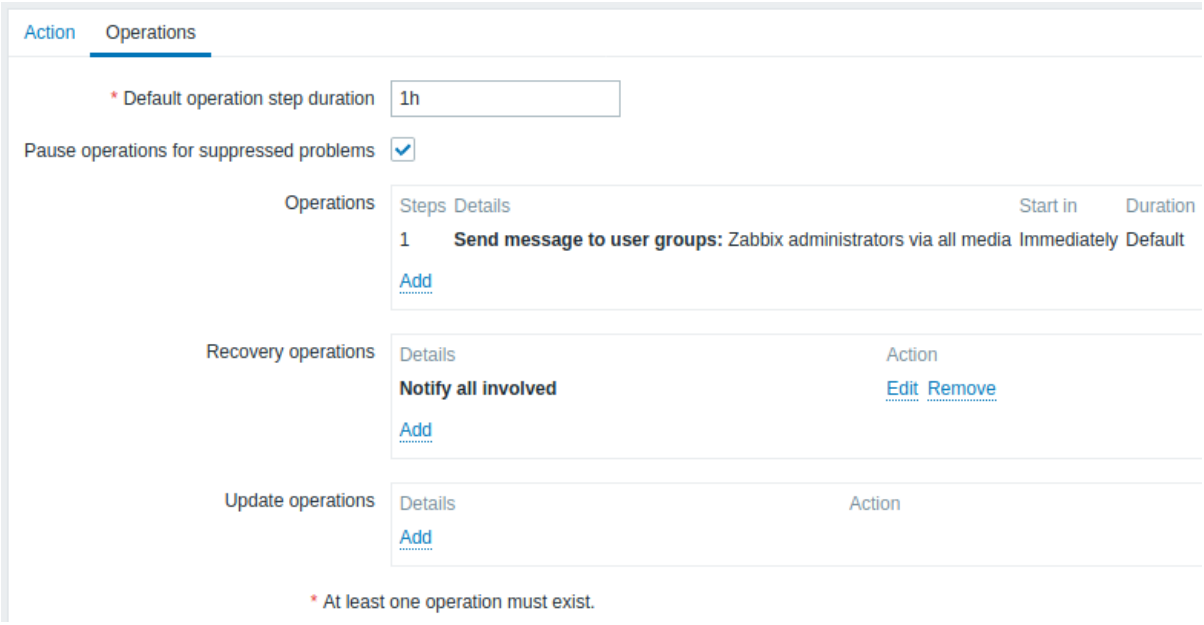
Attention:
Zabbix server does not create alerts if access to the host is explicitly "denied" for the user defined as action operation recipient or if the user has no rights defined to the host at all.

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.



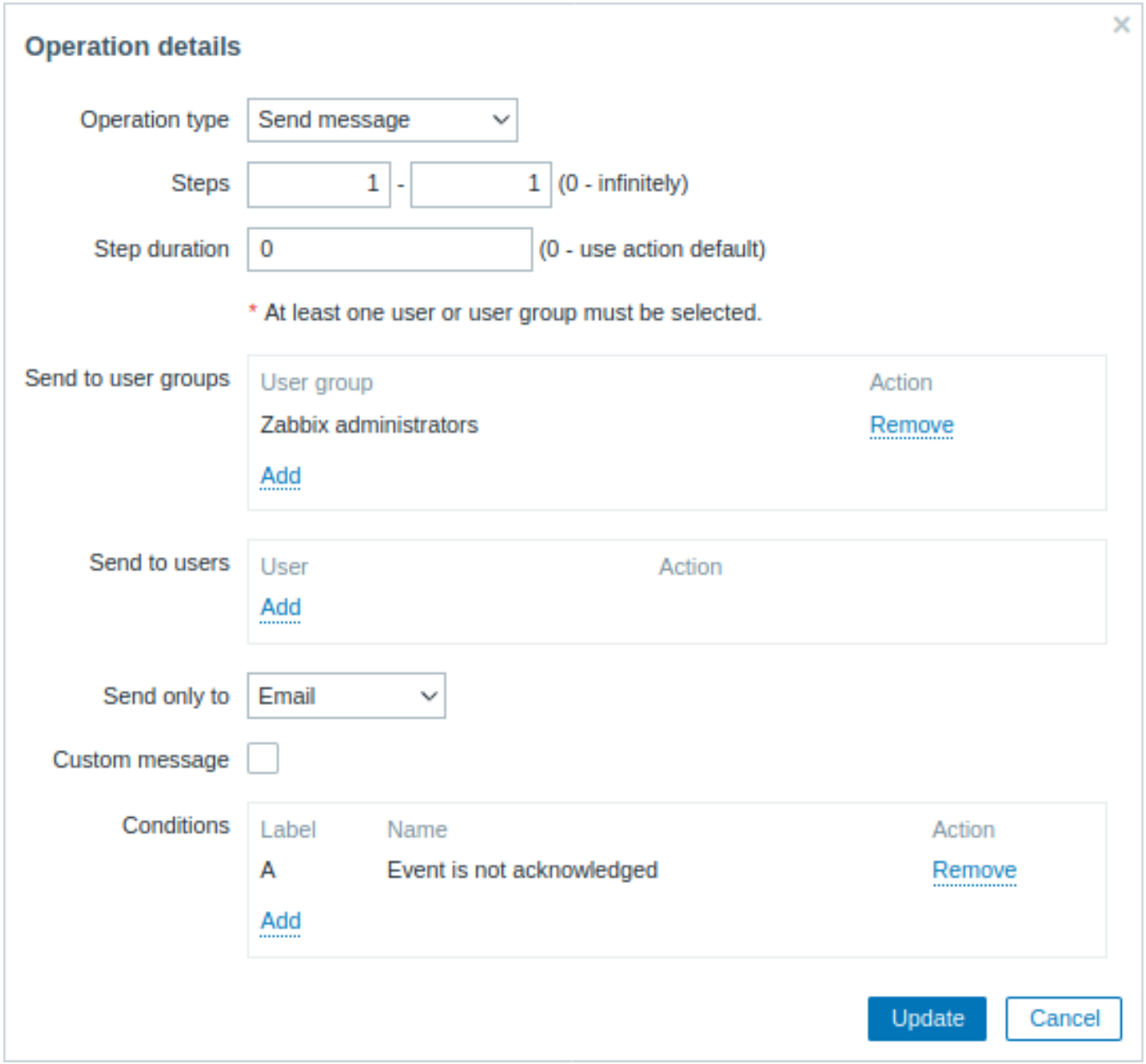
To configure details of a new operation, click on Add in the Operations block. To edit an existing operation, click on Edit next to the operation. A popup window will open where you can edit the operation step details.

All mandatory input fields are marked with a red asterisk.

General operation attributes:

Parameter	Description
Default operation step duration	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.
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 supported since Zabbix 3.2.0.
Operations	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 alias) 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. **Action** - links for editing and removing an operation are displayed.
Recovery operations	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 alias) of a notification recipient. **Action** - links for editing and removing an operation are displayed.
Update operations	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 alias) of a notification recipient. **Action** - links for editing and removing an operation are displayed.

Operation details



Parameter	Description
Operation type	Two operation types are available for all events: **Send message** - send message to user **Remote command** - execute a remote command More operations are available for discovery and autoregistration based events (see above).
Steps	Select the step(s) to assign the operation to in an escalation schedule: **From** - execute starting with this step **To** - execute until this step (0=infinity, execution will not be limited)
Step duration	Custom duration for these steps (0=use default step duration). Time suffixes are supported, e.g. 60s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. Several operations can be assigned to the same step. If these operations have different step duration defined, the shortest one is taken into account and applied to the step.

Parameter		Description
	Operation type: send message	
	Send to user groups	Click on *Add* to select user groups to send the message to. The user group must have at least "read" permissions to the host in order to be notified.
	Send to users	Click on *Add* to select users to send the message to. The user must have at least "read" permissions to the host in order to be notified.
	Send only to	Send message to all defined media types or a selected one only.
	Custom message	If selected, the custom message can be configured. For notifications about internal events via webhooks, custom message is mandatory.
	Subject	Subject of the custom message. The subject may contain macros. It is limited to 255 characters.
	Message	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).
	Operation type: remote command	
	Target list	Select targets to execute the command on: **Current host** - 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. **Host** - select host(s) to execute the command on. **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 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.
	Type	Select the command type: **IPMI** - execute an IPMI command **Custom script** - execute a custom set of commands **SSH** - execute an SSH command **Telnet** - execute a Telnet command **Global script** - execute one of the global scripts defined in *Administration→Scripts*.

Parameter		Description
	Execute on	Execute a custom script on: **Zabbix agent** - the script will be executed by Zabbix agent on the host **Zabbix server (proxy)** - the script will be executed by Zabbix server or proxy - depending on whether the host is monitored by server or proxy **Zabbix server** - the script will be executed by Zabbix server only To execute scripts on the agent, system.run items must be allowed. To execute scripts on proxy, it must be configured (*EnableRemoteCommands* parameter enabled) to allow remote commands from the server. This field is available if 'Custom script' is selected as *Type*.
	Commands	Enter the command(s). Supported macros will be resolved based on the trigger expression that caused the event. For example, host macros will resolve to the hosts of the trigger expression (and not of the target list).
Conditions		Condition for performing the operation: **Not ack** - only when the event is unacknowledged **Ack** - only when the event is acknowledged.

When done, click on *Add* to add 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

Warning:
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

Attention:
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.

Attention:
Since ORACLE alert message parameters field has a length limitation, message parameters can not be longer than 2048 characters (exceeding characters will be truncated).

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 (also in agent configuration before Zabbix 5.0.2)

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):

Database	Limit in characters	Limit in bytes
MySQL	65535	65535
Oracle Database	2048	4000
PostgreSQL	65535	not limited
SQLite (only Zabbix proxy)	65535	not limited

Configuration

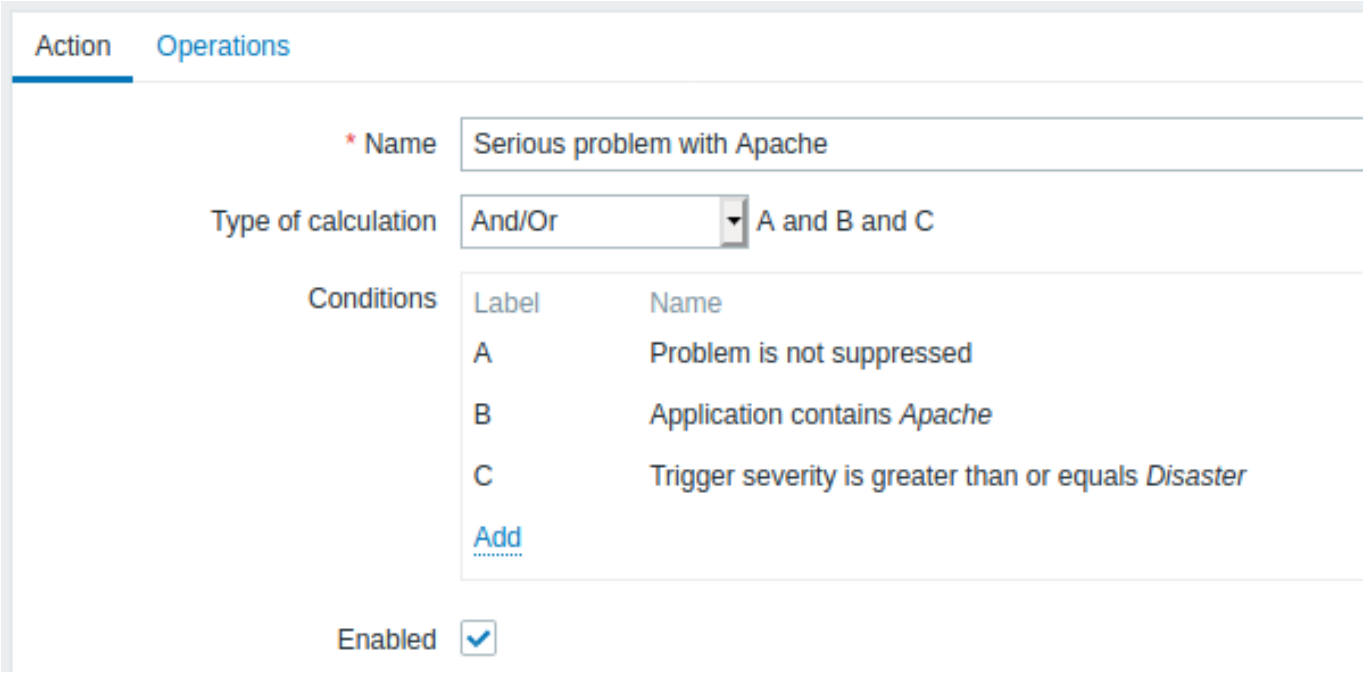
Those remote commands that are executed on Zabbix agent (custom scripts) must be first enabled in the agent configuration.

Make sure that the `AllowKey=system.run[*]` parameter is added. Restart agent daemon if changing this parameter.

Attention:
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:



All mandatory input fields are marked with a red asterisk.

- In the *Operations* tab, select the **Remote command** operation type
- Select the remote command type (IPMI, Custom script, SSH, Telnet, Global script)
- If *Custom script* type is selected, choose the way how custom script will be executed (by Zabbix agent, Zabbix server (proxy) or Zabbix server only)
- Enter the remote command

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*).

Attention:
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**.

Note:
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:

```
# visudo
```

Example lines that could be used in *sudoers* file:

```
# 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
```

Note:
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:

```
<command> [<value>]
```

where

- <command> - one of IPMI commands without spaces
- <value> - 'on', 'off' or any unsigned integer. <value> is an optional parameter.

Examples

Example 1

Restart of Windows on certain condition.

In order to automatically restart Windows upon a problem detected by Zabbix, define the following actions:

PARAMETER	Description
Operation type	'Remote command'
Type	'Custom script'
Command	c:\windows\system32\shutdown.exe -r -f

Example 2

Restart the host by using IPMI control.

PARAMETER	Description
Operation type	'Remote command'
Type	'IPMI'
Command	reset

Example 3

Power off the host by using IPMI control.

PARAMETER	Description
Operation type	'Remote command'
Type	'IPMI'
Command	power off

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.

> **Note:**
> 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.

> **Attention:**
> If performing discovery by a proxy, currently hostname lookup still takes place on Zabbix server.

> **Attention:**
> 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: {zabbix.zabbix.com:system.cpu.load[,avg1].last()}
```

When you receive the message, the message will be replaced by something like:

```
Processor load is: 1.45
```

Example 3

Message:

```
Latest value: {{HOST.HOST}:{ITEM.KEY}.last()}
MAX for 15 minutes: {{HOST.HOST}:{ITEM.KEY}.max(900)}
MIN for 15 minutes: {{HOST.HOST}:{ITEM.KEY}.min(900)}
```

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: {Myhost:system.cpu.load[percpu,avg1].last()}>5 or {Myotherhost:system.cpu.load[percpu,

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:

Event ID: 21874
Event value: 1
Event status: PROBLEM
Event time: 13:04:30
Event date: 2018.01.02
Event age: 5m
Event acknowledgment: Yes
Event update history: 2018.01.02 13:05:51 "John Smith (Admin)"
Actions: acknowledged.

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
```

Attention:
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.

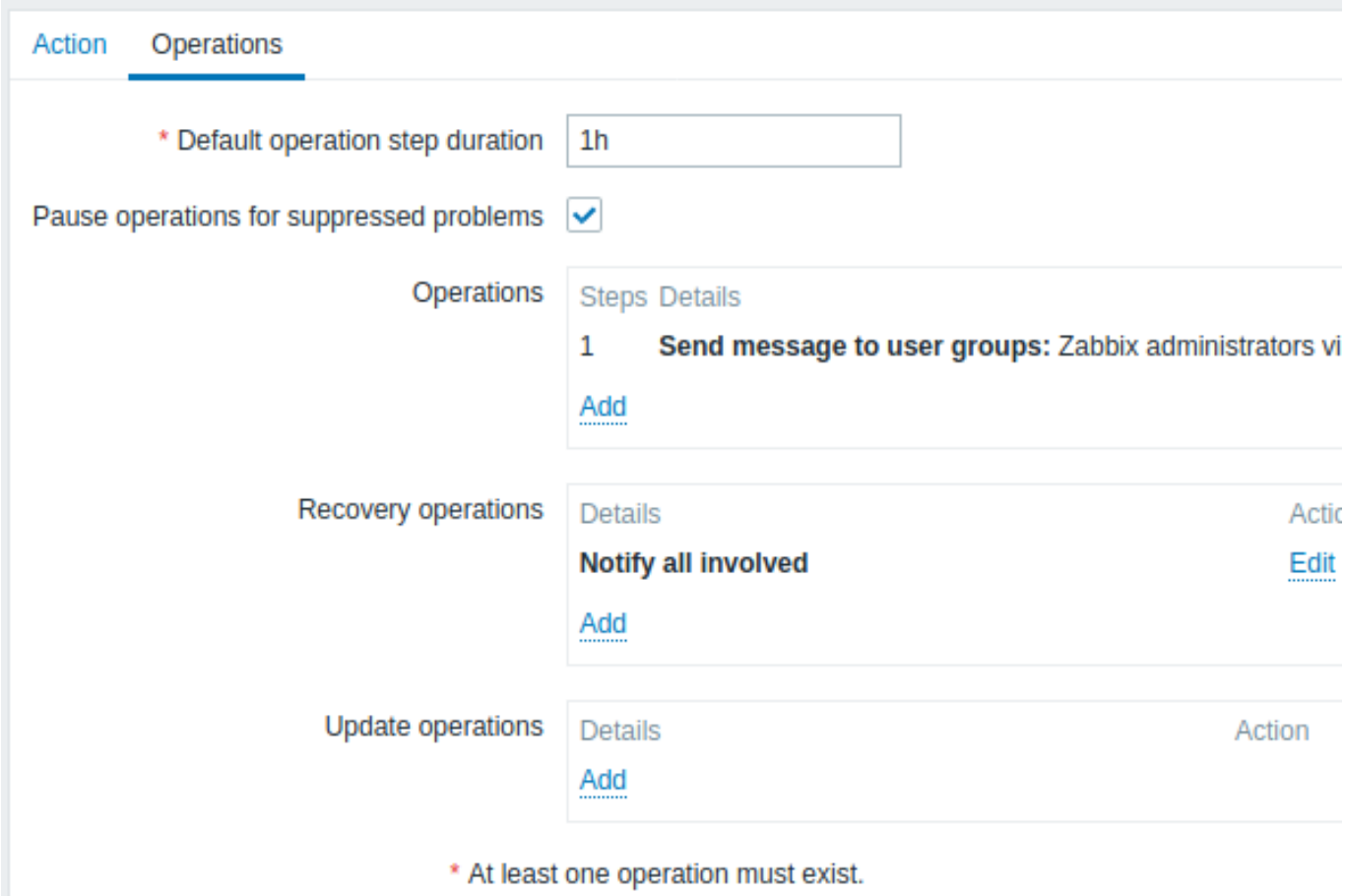
Use cases

Some use cases for recovery operations are as follows:

1. Notify on a recovery all users that were notified on the problem:
   - Select *Notify all involved* as operation type.
2. Have multiple operations upon recovery: send a notification and execute a remote command:
   - Add operation types for sending a message and executing a command.
3. 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.

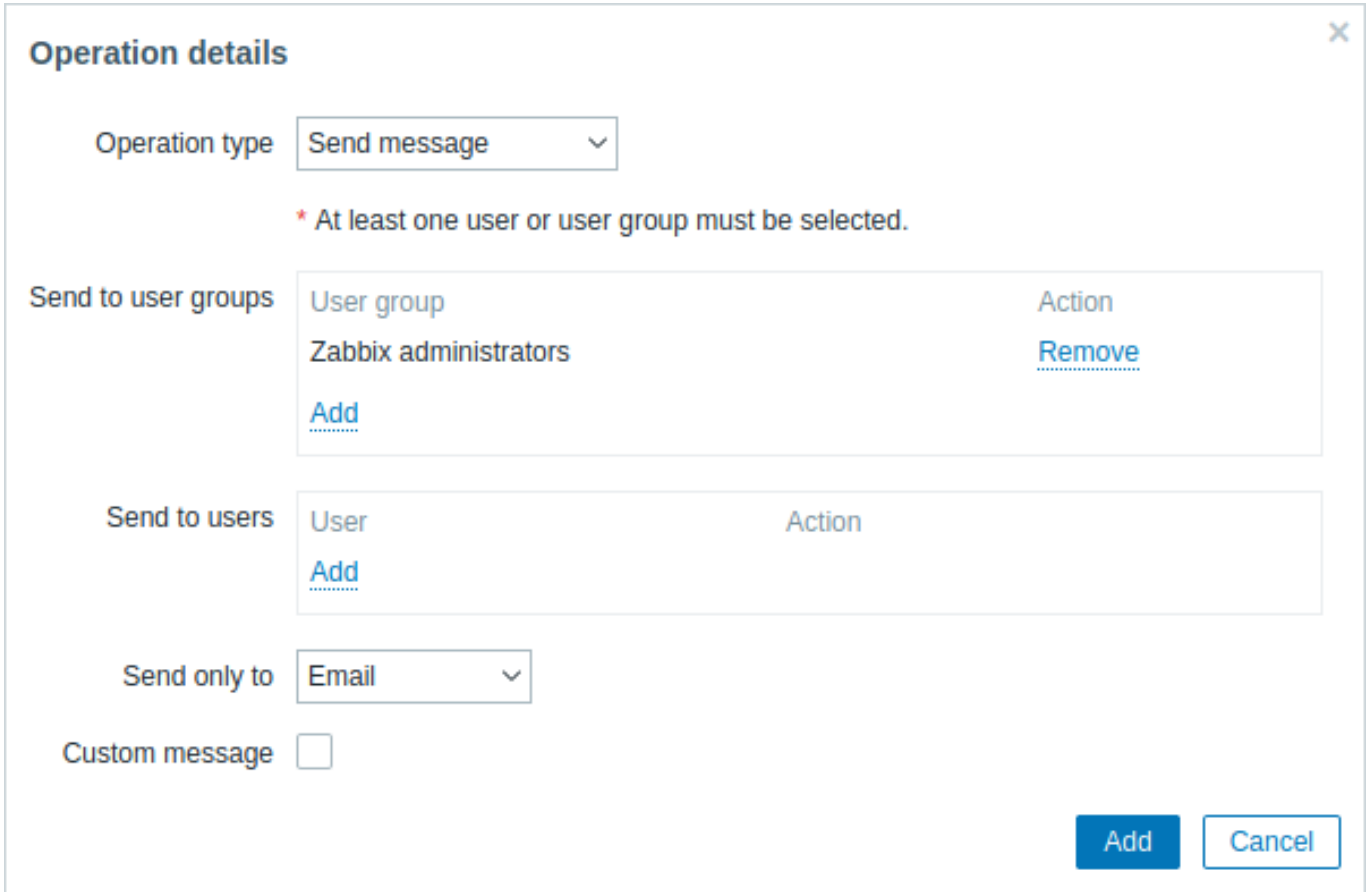
Configuring a recovery operation

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



Parameter		Description
Operation type		Three operation types are available for recovery events: **Send message** - send recovery message to specified user **Remote command** - execute a remote command **Notify all involved** - send recovery message to all users who were notified on the problem event 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	
	Send to user groups	Click on *Add* to select user groups to send the recovery message to. The user group must have at least ”read” permissions to the host in order to be notified.
	Send to users	Click on *Add* to select users to send the recovery message to. The user must have at least ”read” permissions to the host in order to be notified.
	Send only to	Send default recovery message to all defined media types or a selected one only.
	Custom message	If selected, a custom message can be defined.
	Subject	Subject of the custom message. The subject may contain macros.

Parameter		Description
	Message	The custom message. The message may contain macros.
	Operation type: remote command	
	Target list	Select targets to execute the command on: **Current host** - 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. **Host** - select host(s) to execute the command on. **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.
	Type	Select the command type: **IPMI** - execute an IPMI command **Custom script** - execute a custom set of commands **SSH** - execute an SSH command **Telnet** - execute a Telnet command **Global script** - execute one of the global scripts defined in *Administration→Scripts*.
	Execute on	Execute a custom script on: **Zabbix agent** - the script will be executed by Zabbix agent on the host **Zabbix server (proxy)** - the script will be executed by Zabbix server or proxy - depending on whether the host is monitored by server or proxy **Zabbix server** - the script will be executed by Zabbix server only To execute scripts on the agent, it must be configured to allow remote commands from the server. This field is available if 'Custom script' is selected as *Type*.
	Commands	Enter the command(s). Supported macros will be resolved based on the trigger expression that caused the event. For example, host macros will resolve to the hosts of the trigger expression (and not of the target list).
	Operation type: notify all involved	

Parameter		Description
	Custom message	If selected, a custom message can be defined.
	Subject	Subject of the custom message. The subject may contain macros.
	Message	The custom message. The message may contain macros.

All mandatory input fields are marked with a red asterisk. When done, click on *Add* to add operation to the list of *Recovery operations*.

4 Update operations

Overview

Update operations allow you to be notified when problems are updated by other users, i.e.:

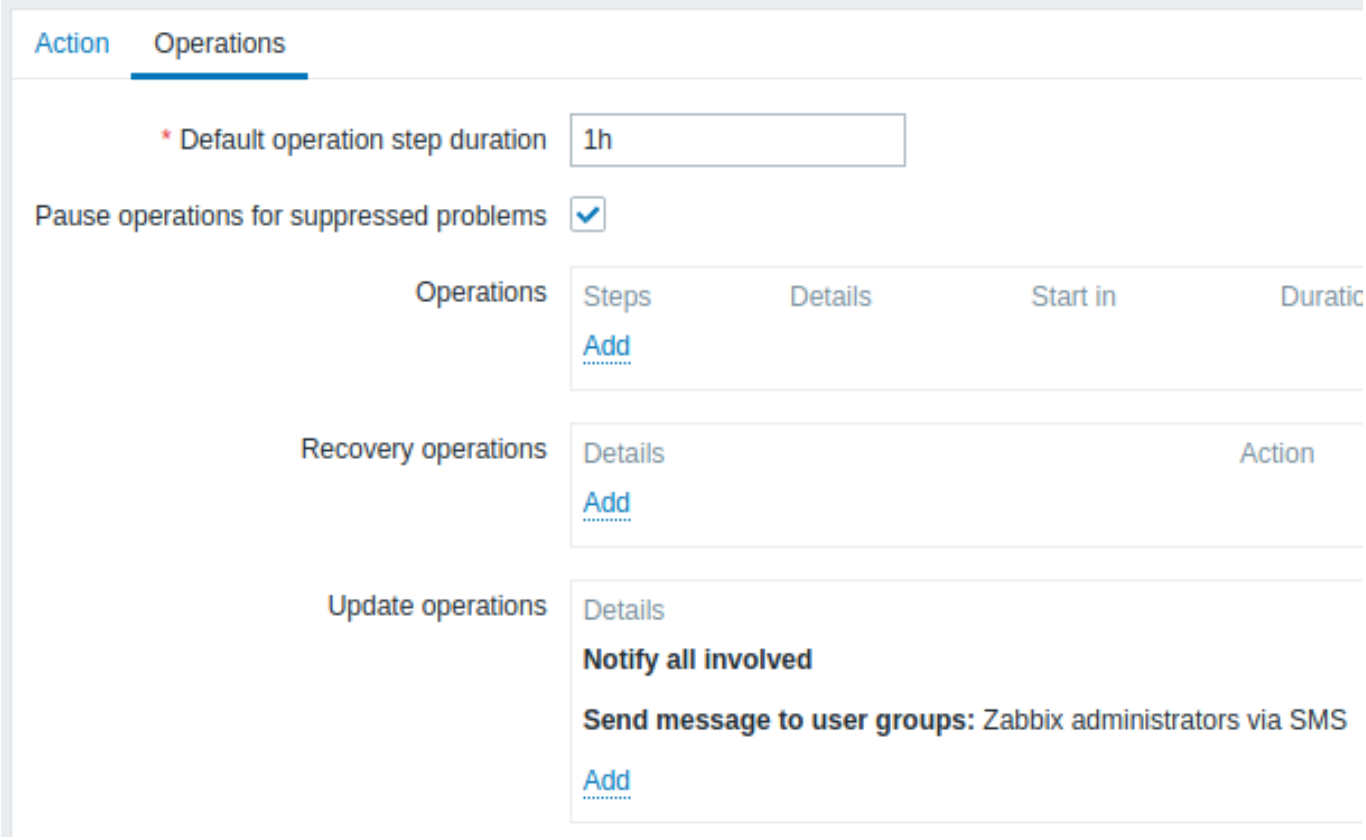
- commented upon
- acknowledged
- severity changed
- closed (manually)

Update operations are available in actions with the event source as *Triggers*.

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.

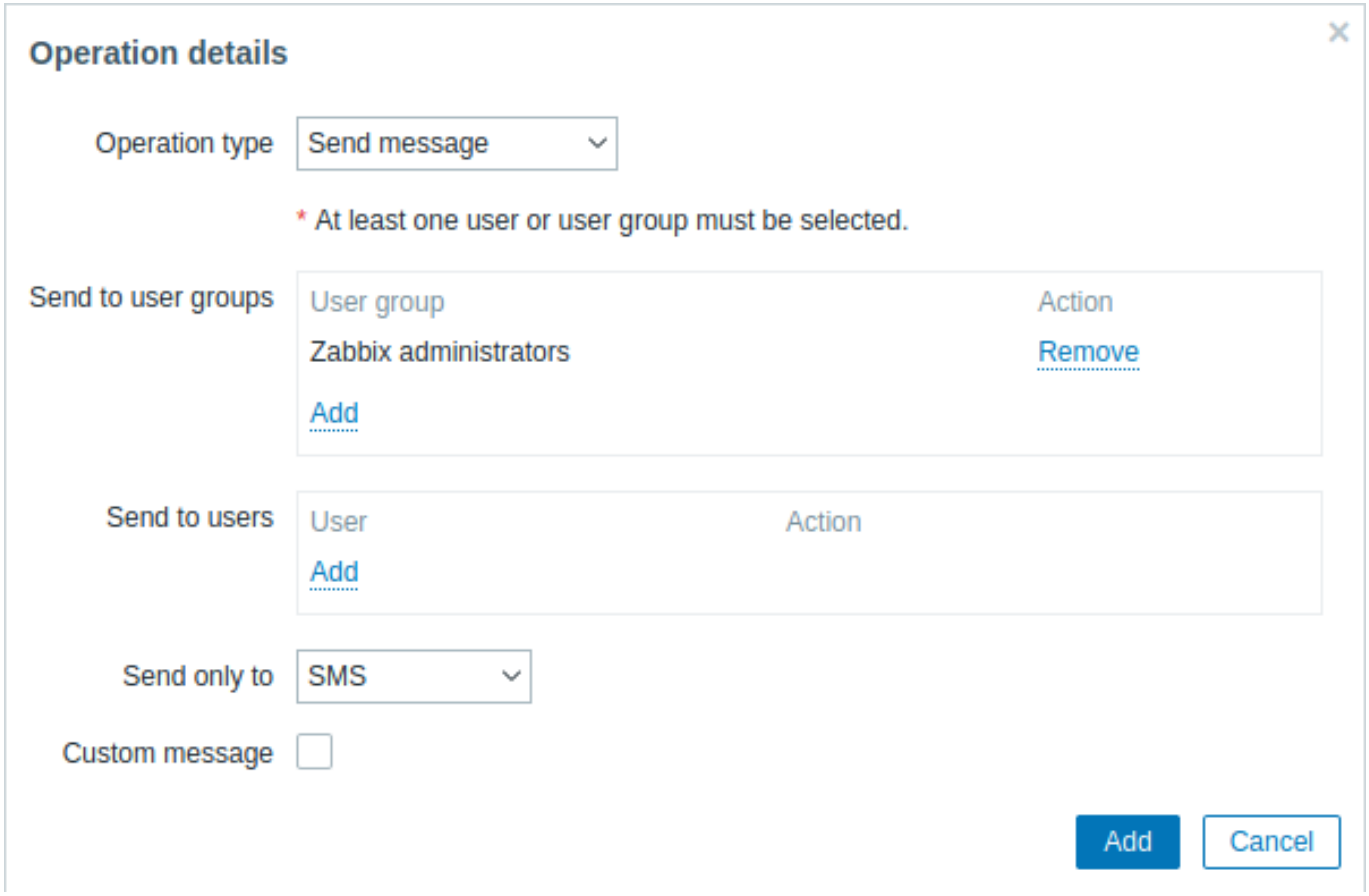
Configuring an update operation

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 operation details


Operation details
Operation type
Send message
* At least one user or user group must be selected.
Send to user groups
User group
Action
Zabbix administrators
Remove
Add
Send to users
User
Action
Add
Send only to
SMS
Custom message
Add
Cancel

Operation type	Three operation types are available for update operations: **Send message** - send update message to specified user when event is updated, for example, acknowledged **Remote command** - execute a remote command when event is updated, for example, acknowledged **Notify all involved** - send notification message to all users who received notification about the problem appearing and/or have updated the problem event. If the same recipient with unchanged default subject/message is defined in several operation types, duplicate notifications are not sent. The person who updates a problem does not receive notification about their own update.
Operation type: send message	
Send to user groups	Click on *Add* to select user groups to send the update message to. The user group must have at least ”read” permissions to the host in order to be notified.
Send to users	Click on *Add* to select users to send the update message to. The user must have at least ”read” permissions to the host in order to be notified.
Send only to	Send update message to all defined media types or a selected one only.

	Custom message	If selected, the custom message can be defined.
	Subject	Subject of the custom message. The subject may contain macros.
	Message	The custom message. The message may contain macros.
	Operation type: remote command	

Target list	Select targets to execute the command on: **Current host** - 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. **Host** - select host(s) to execute the command on. **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.

Type	Select the command type: **IPMI** - execute an IPMI command **Custom script** - execute a custom set of commands **SSH** - execute an SSH command **Telnet** - execute a Telnet command **Global script** - execute one of the global scripts defined in *Administration→Scripts*.
Execute on	Execute a custom script on: **Zabbix agent** - the script will be executed by Zabbix agent on the host **Zabbix server (proxy)** - the script will be executed by Zabbix server or proxy - depending on whether the host is monitored by server or proxy **Zabbix server** - the script will be executed by Zabbix server only To execute scripts on the agent, it must be configured to allow remote commands from the server. This field is available if 'Custom script' is selected as *Type*.
Commands	Enter the command(s). Supported macros will be resolved based on the trigger expression that caused the event. For example, host macros will resolve to the hosts of the trigger expression (and not of the target list).
Operation type: notify all involved	

	Default media type	Users who update a problem but have not received notifications about the problem appearing will receive notifications about further updates on the selected default media type - Email or SMS. This field is available since Zabbix 3.4.2.
	Custom message	If selected, the custom message can be defined.
	Subject	Subject of the custom message. The subject may contain macros.
	Message	The custom message. The message may contain macros.

All mandatory input fields are marked with a red asterisk. When done, click on *Add* to add operation to the list of *Update 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.

Situation	Behavior
The host in question goes into maintenance after the initial problem notification is sent	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.
*The time period defined in the **Time period** action condition ends after the initial notification is sent*	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.

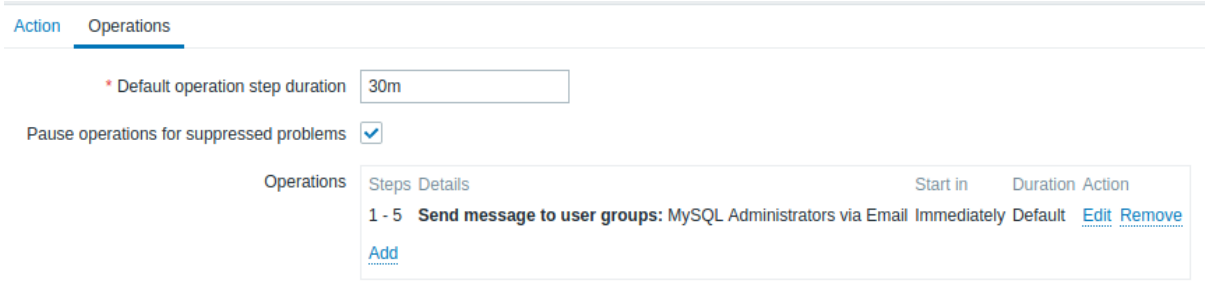
Situation	Behavior
A problem starts during maintenance and continues (is not resolved) after maintenance ends	Depending on the *Pause operations for suppressed problems* setting in action configuration, all escalation steps are executed either from the moment maintenance ends or immediately.
A problem starts during a no-data maintenance and continues (is not resolved) after maintenance ends	It must wait for the trigger to fire, before all escalation steps are executed.
Different escalations follow in close succession and overlap	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.
During an escalation in progress (like a message being sent), based on any type of event: - the action is disabled Based on trigger event: - the trigger is disabled - the host or item is disabled Based on internal event about triggers: - the trigger is disabled Based on internal event about items/low-level discovery rules: - the item is disabled - the host is disabled	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 '<Action name>' 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). Note that the *Escalation canceled* message is also sent if operations are finished, but recovery operations are configured and are not executed yet.
During an escalation in progress (like a message being sent) the action is deleted	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`

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).

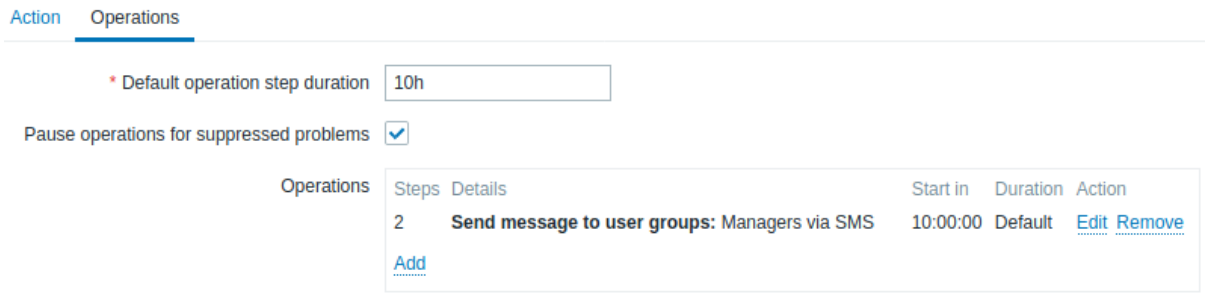
If the problem is resolved and a recovery message is configured, it will be sent to those who received at least one problem message within this escalation scenario.

Note:
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'



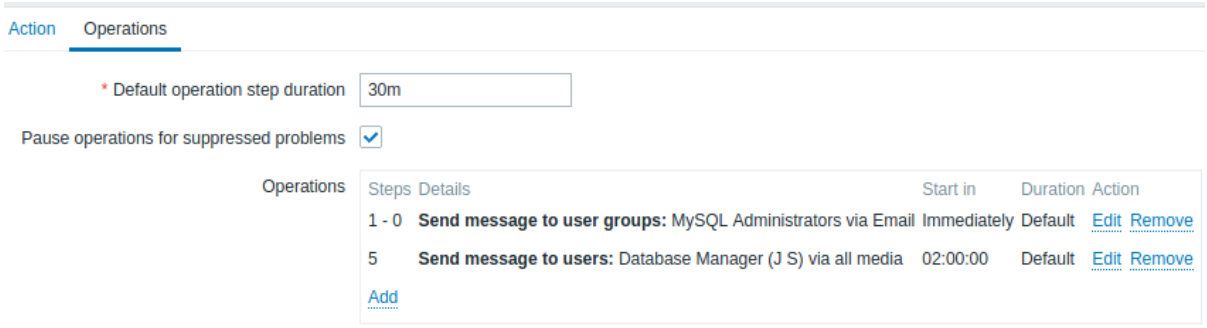
A notification will only be sent at Step 2 of the escalation scenario, or 10 hours after the problem starts.

You can customize the message text to something like 'The problem is more than 10 hours old'.

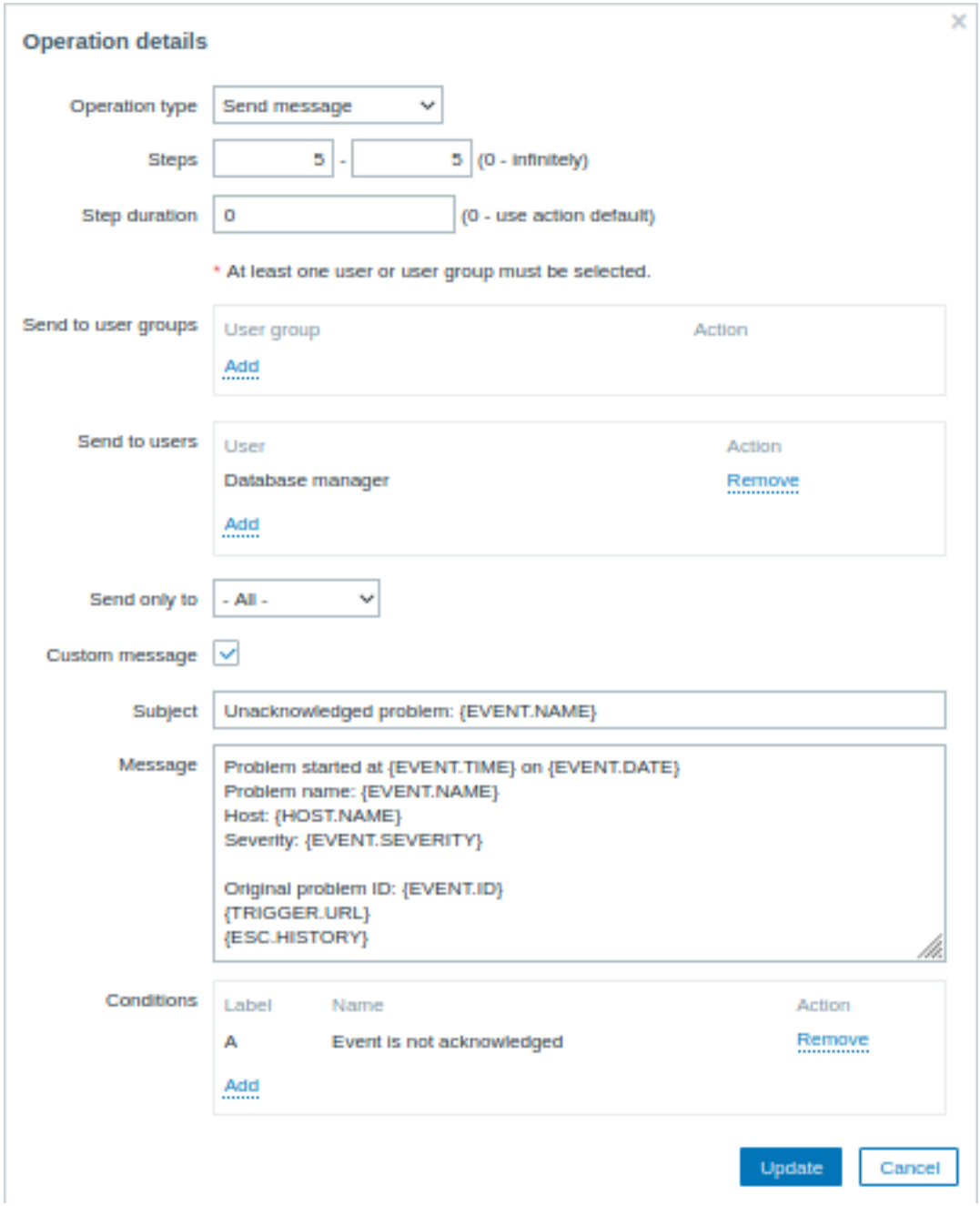
Example 3

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:



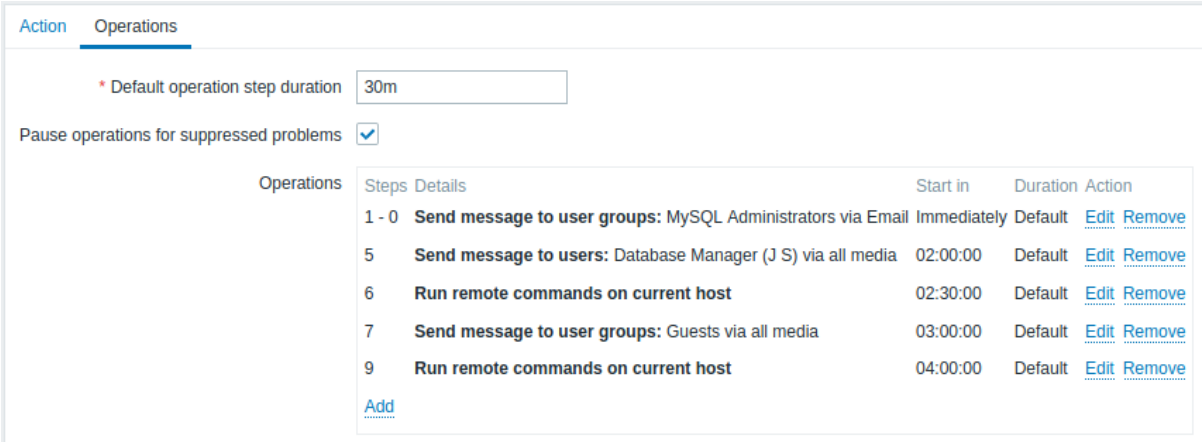
Note the use of {ESC.HISTORY} macro in the customized message. The macro will contain information about all previously executed steps on this escalation, such as notifications sent and commands executed.

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.

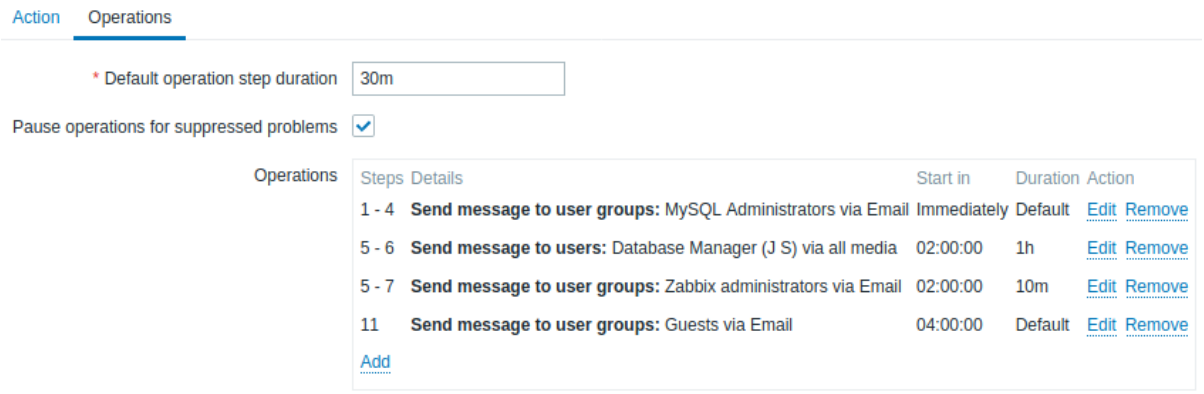
If the problem still exists, after another 30 minutes Zabbix will send a message to all guest users.

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.

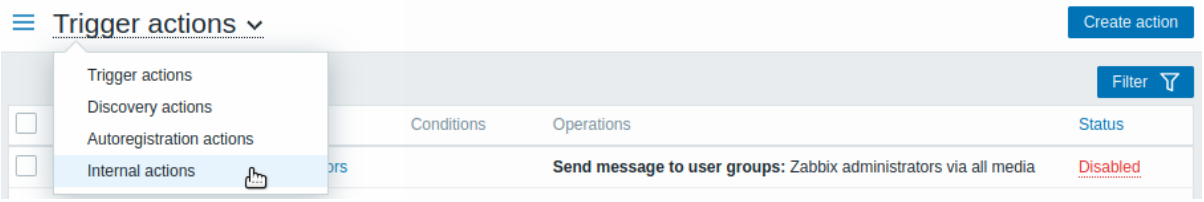
Step 1

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.

Attention:
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 page title dropdown.

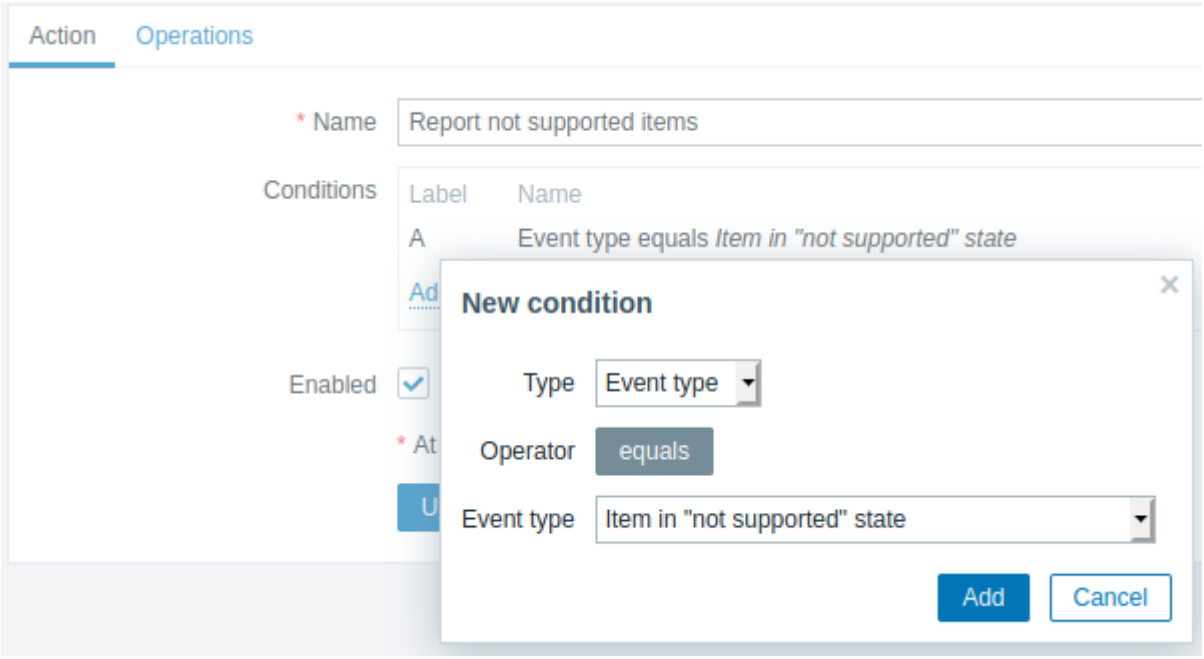


Click on *Create action* to the right to open an action configuration form.

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.

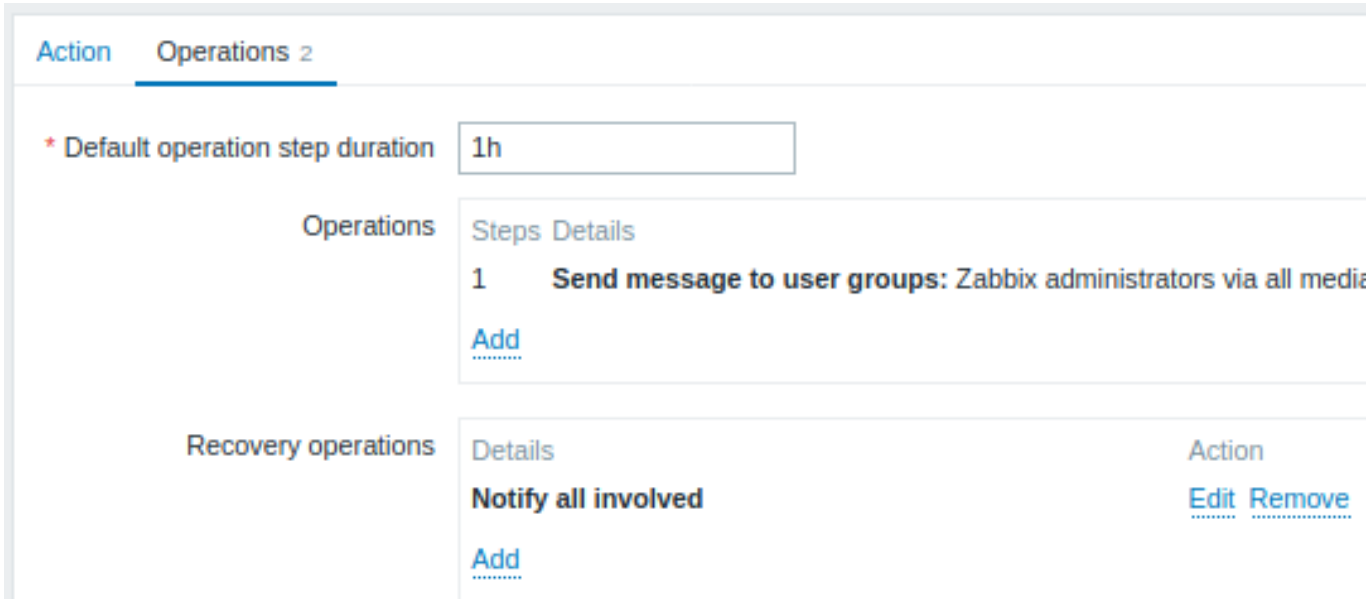


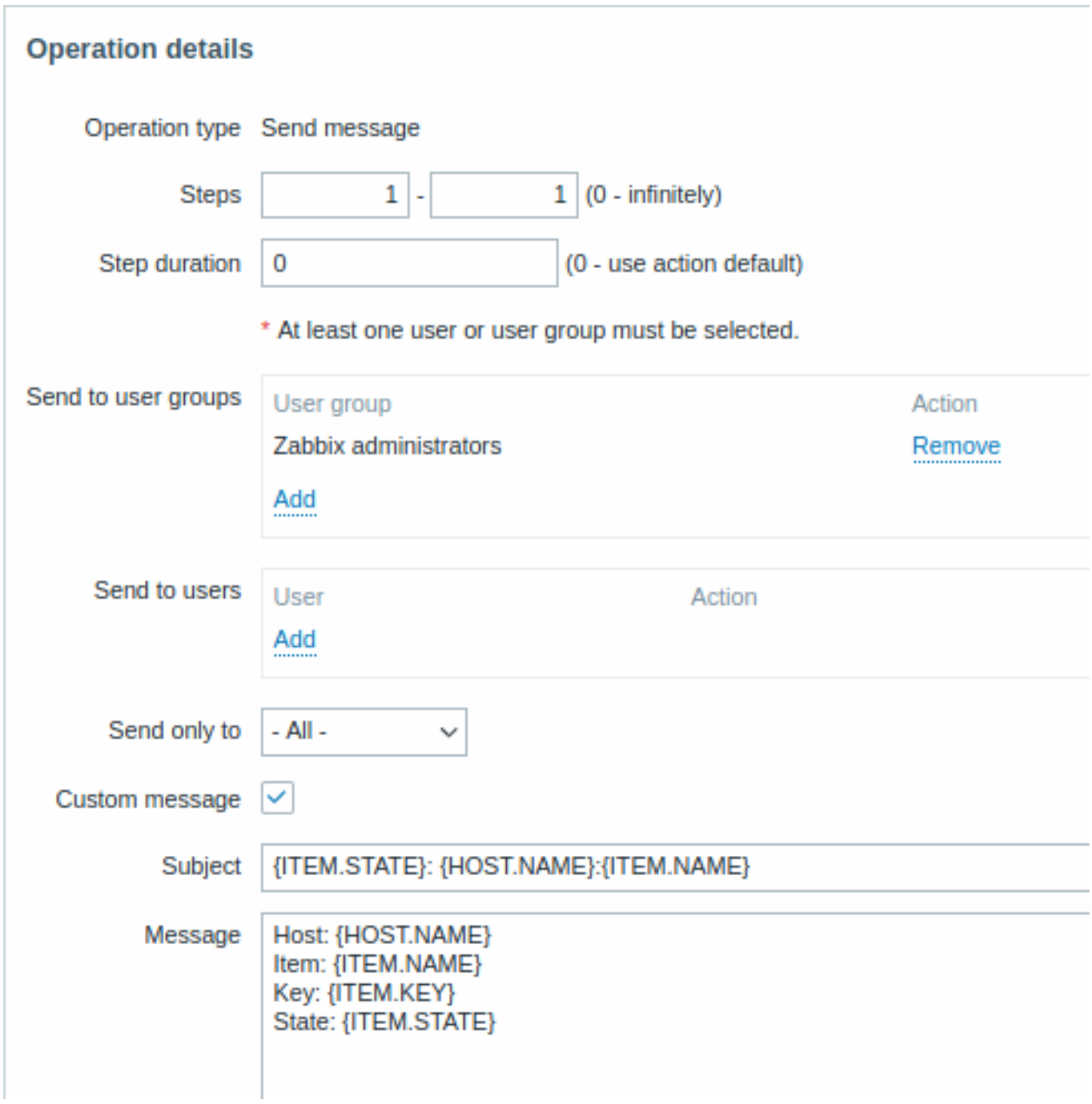
Don't forget to click on *Add* to actually list the condition in the *Conditions* block.

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.





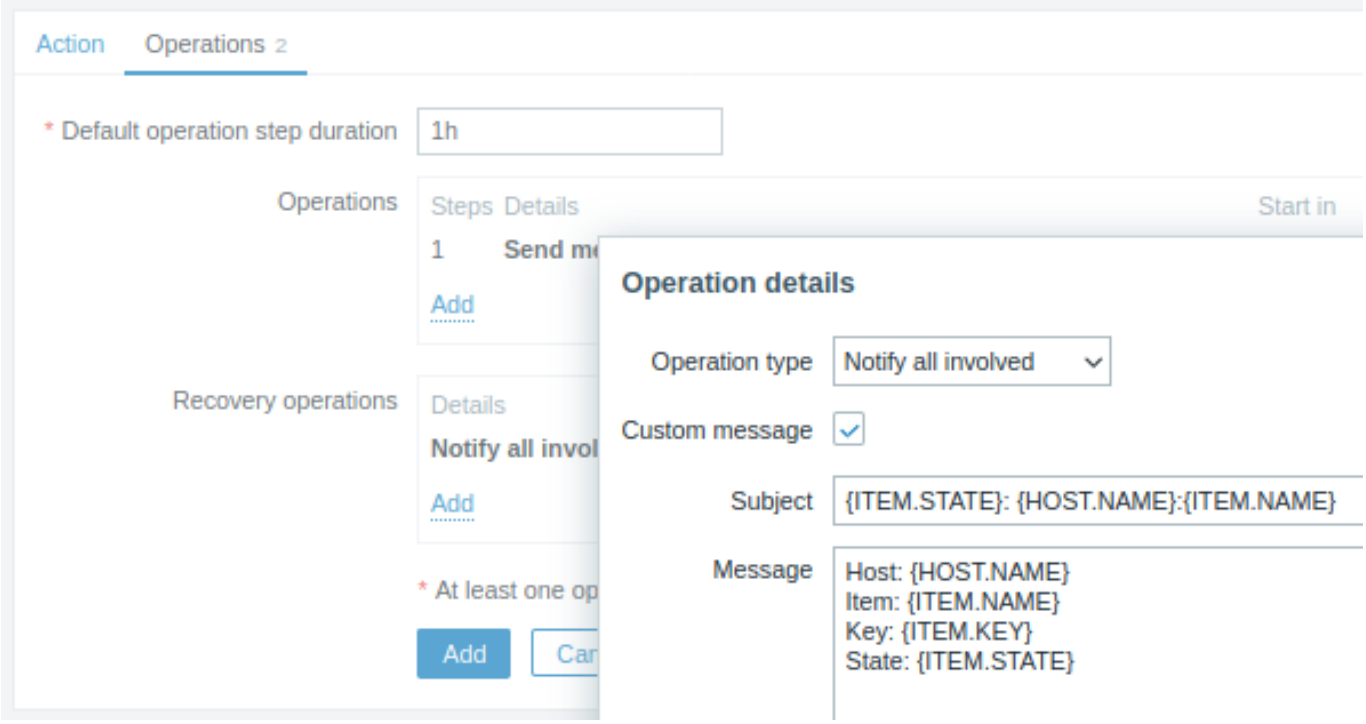
Click on *Add* to actually list the operation in the *Operations* block.

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.

10 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.

Besides built-in macros Zabbix also supports user-defined macros, user-defined macros with context and macros for low-level discovery.

See also:

- full list of built-in macros

- macro functions
- user macros
- user macros with context
- low-level discovery macros

1 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:

```
{{ITEM.VALUE}.regsub(pattern, output)}
{{#LLDMACRO}.regsub(pattern, output)}
```

Supported macro functions

FUNCTION	Description	Parameters	Supported for
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.	{ITEM.VALUE} {ITEM.LASTVALUE} Low-level discovery macros (except in low-level discovery rule filter)
iregsub (<pattern>,<output>)			

FUNCTION			
	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.	{ITEM.VALUE} {ITEM.LASTVALUE} Low-level discovery macros (except in low-level discovery rule filter)

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)

Examples

The ways in which macro functions can be used to customize macro values is illustrated in the following examples containing log lines as received value:

Received value	Macro	Output
123Log line	{{ITEM.VALUE}.regsub("^[0-9]+, Problem)}	Problem
123 Log line	{{ITEM.VALUE}.regsub("^[0-9]+)", "Problem")}	Problem
123 Log line	{{ITEM.VALUE}.regsub("^([0-9]+)", Problem ID: \1)}	123 Problem ID: 123
Log line	{{ITEM.VALUE}.regsub("Problem", "Problem ID: \1")}	Problem ID:
MySQL crashed errno 123	{{ITEM.VALUE}.regsub("(Problem) ([0-9]+)", " Problem ID: \1_\2 ")}	Problem ID: MySQL_123
123 Log line	{{ITEM.VALUE}.regsub("^[0-9]+", "Problem ID: \1")}	UNKNOWN (invalid regular expression)
customername_1	{{#IFALIAS}.regsub("(.*)_([0-9]+)", \1)}	customername
customername_1	{{#IFALIAS}.regsub("(.*)_([0-9]+)", \2)}	1
customername_1	{{#IFALIAS}.regsub("(.*)_([0-9]+)", \1)}	{{#IFALIAS}.regsub("(.*)_([0-9]+) (invalid regular expression)
customername_1	{$MACRO:"{{#IFALIAS}.regsub("(.*)_([0-9]+)", \1)}"}	{$MACRO:"customername"}
customername_1	{$MACRO:"{{#IFALIAS}.regsub("(.*)_([0-9]+)", \2)}"}	{$MACRO:"1"}

Received value	Macro	Output
customername_1	{$MACRO:"{{#IFALIAS}{{#M}.regsub("(.*)_([0-9]+)", \1)}"}	{$MACRO:"{{#M}.regsub("(.*)_([0-9]+)", \1)}"} (invalid regular expression)
customername_1	"{$MACRO:\"{{#IFALIAS}{{#M}.regsub("(.*)_([0-9]+)", \1)}\"}"	"{$MACRO:\"{{#M}.regsub("(.*)_([0-9]+)", \1)}\"}"
customername_1	"{$MACRO:\"{{#IFALIAS}{{#M}.regsub("(.*)_([0-9]+)", \2)}\"}"	"{$MACRO:\"{{#M}.regsub("(.*)_([0-9]+)", \2)}\"}"
customername_1	"{$MACRO:\"{{#IFALIAS}{{#M}.regsub("(.*)_([0-9]+)", \1)}\"}"	"{$MACRO:\"{{#IFALIAS}{{#M}.regsub("(.*)_([0-9]+)", \1)}\"}") (invalid regular expression)

Seeing full item values

Long values of resolved {ITEM.VALUE} and {ITEM.LASTVALUE} macros for text/log items are truncated to 20 characters in some frontend locations. To see the full values of these macros you may use macro functions, e.g.:

```
{{ITEM.VALUE}.regsub("(.*)", \1)}<br> {{ITEM.LASTVALUE}.regsub("(.*)", \1)}
```

See also: {ITEM.VALUE} and {ITEM.LASTVALUE} macro details.

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.

Warning:
If a macro with the **same name** exists on multiple 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.

Attention:
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 name
- 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.

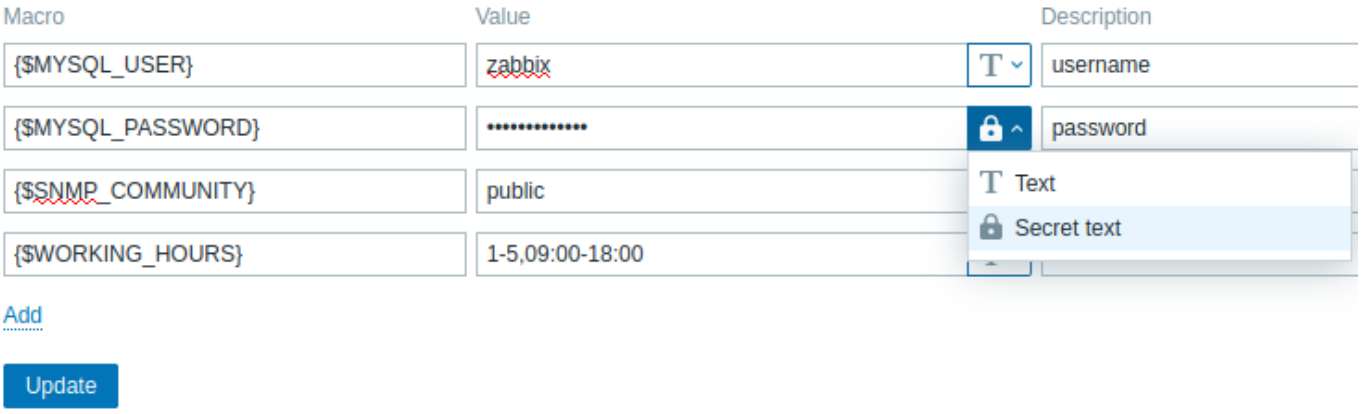
Configuration

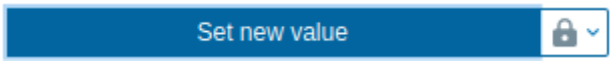
To define user macros, go to the corresponding location in the frontend:

- for global macros, visit *Administration* → *General* → *Macros*
- for host and template level macros, open host or template properties and look for the *Macros* tab

Note:
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.

User macro has the following attributes:



Parameter	Description
Macro	Name of a macro. Must be wrapped in curly brackets. Text inside the brackets must start with a dollar sign. Example: {$FRONTEND_URL}. The following characters are allowed in the macro names: **A-Z** (uppercase only) , **0-9** , _ , **.**
Value	Content of a macro. Starting from Zabbix 5.0 two types of values are supported in user macros: *Text* (default) and *Secret text*. *Secret text* mode masks the content of a macro with asterisks, which could be useful to protect sensitive information such as passwords or shared keys. *Note* that while the value of a secret macro is hidden from sight, 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. To select macro value type click on a button at the right end of the `Value` input field:  icon indicates a text macro  icon indicates a secret text macro. Upon hovering, the `Value` field transforms into  button, which allows to enter a new value of the macro (to exit without saving a new value, click backwards arrow ()
Description	Text field used to provide more information about this macro.

Note:
URLs that contain a secret macro will not work as the macro in them will be resolved as "******".

Attention:
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:

```
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:

```
{ca_001:system.cpu.load[,avg1].last()}>{$MAX_CPULOAD}
```

Such a trigger would be created on the template, not edited in individual hosts.

Note:
If you want to use amount of values as the function parameter (for example, **max(#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:

```
{ca_001:system.cpu.load[,avg1].min({$CPULOAD_PERIOD})}>{$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:

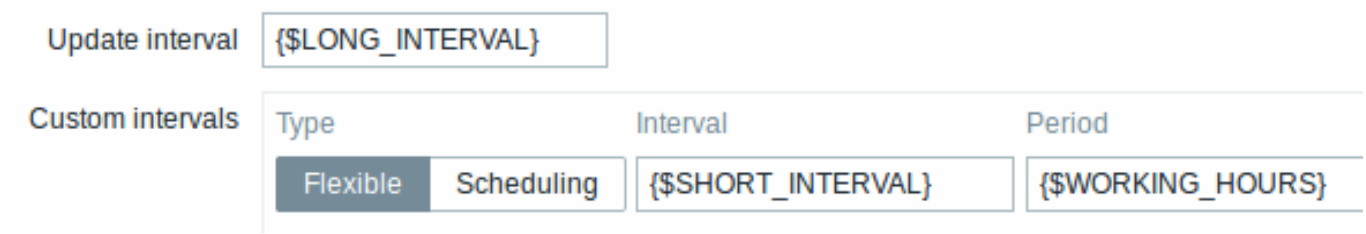
- define {$INTERVAL} macro and use it in the item update interval;
- use {$INTERVAL} as parameter of the agent unavailability trigger:

```
{ca_001:agent.ping.nodata({$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 *Administration → General → Working time*;
- use it in *User → Media → When active*;
- use it to set up more frequent item polling during working hours:

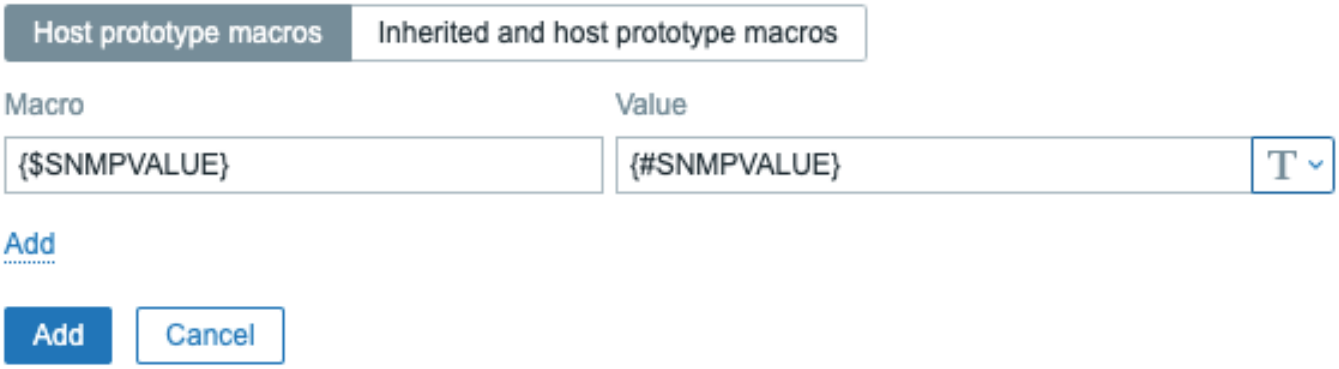


- use it in the *Time period* action condition;
- adjust the working time in *Administration → General → Macros*, if needed.

Example 6

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:



- assign *Template Module Generic SNMPv2* to the host prototype;
- use {$SNMPVALUE} in the *SNMP OID* field of *Template Module Generic SNMPv2* 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 (supported since Zabbix **5.0.2**):

```
{$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:

Example	Description
{$LOW_SPACE_LIMIT}	User macro without context.
{$LOW_SPACE_LIMIT:/tmp}	User macro with context (static string).
{$LOW_SPACE_LIMIT:regex:"^/tmp$"}	User macro with context (regular expression). Same as {$LOW_SPACE_LIMIT:/tmp}.
{$LOW_SPACE_LIMIT:regex:"^/var/log/.*$"}	User macro with context (regular expression). Matches all strings prefixed with /var/log/.

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:

```
{host:vfs.fs.size[{#FSNAME},pfree].last()}<{$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 that 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.

Warning:
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[1]
  - history storage period[1]
  - trend storage period[1]
  - application prototypes
  - item value preprocessing steps
  - SNMP OID
  - IPMI sensor field
  - calculated/aggregate item expression, in:
    - expression constants and function parameters
    - item key parameters
  - SSH script and Telnet script
  - database monitoring SQL query
  - JMX item endpoint field
  - description

    - HTTP agent URL field
    - HTTP agent HTTP query fields field
    - HTTP agent request body field
    - HTTP agent required status codes field
    - HTTP agent headers field key and value
    - HTTP agent HTTP authentication username field
    - HTTP agent HTTP authentication password field
    - HTTP agent HTTP proxy field
    - HTTP agent HTTP SSL certificate file field
    - HTTP agent HTTP SSL key file field
    - HTTP agent HTTP SSL key password field
    - HTTP agent HTTP timeout[1] field
- for trigger prototypes in
    - name
    - operational data
    - expression (only in constants and function parameters)
    - URL
    - description
    - event tag name and value
- for graph prototypes in
    - name
- for host prototypes in
    - name
    - visible name
    - host group prototype name
    - host macro value
    - (see the full list)

In all those places LLD macros can be used inside static user macro context.

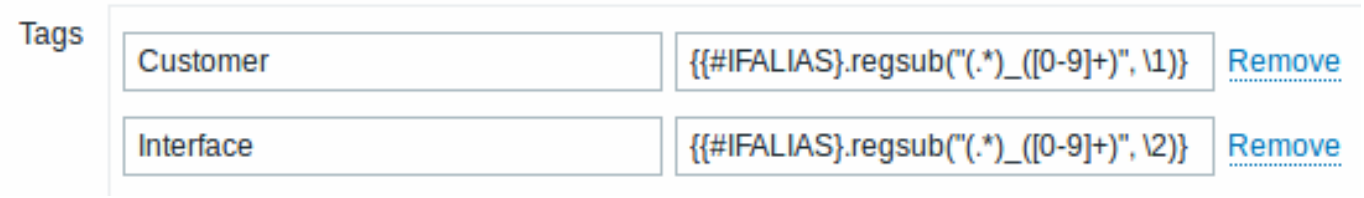
Using macro functions

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:



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]+)\", \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.

11 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 access to:

- administrative Zabbix frontend functions
- monitored hosts in hostgroups

1 Configuring a user

Overview

The initial Zabbix installation has two predefined users:

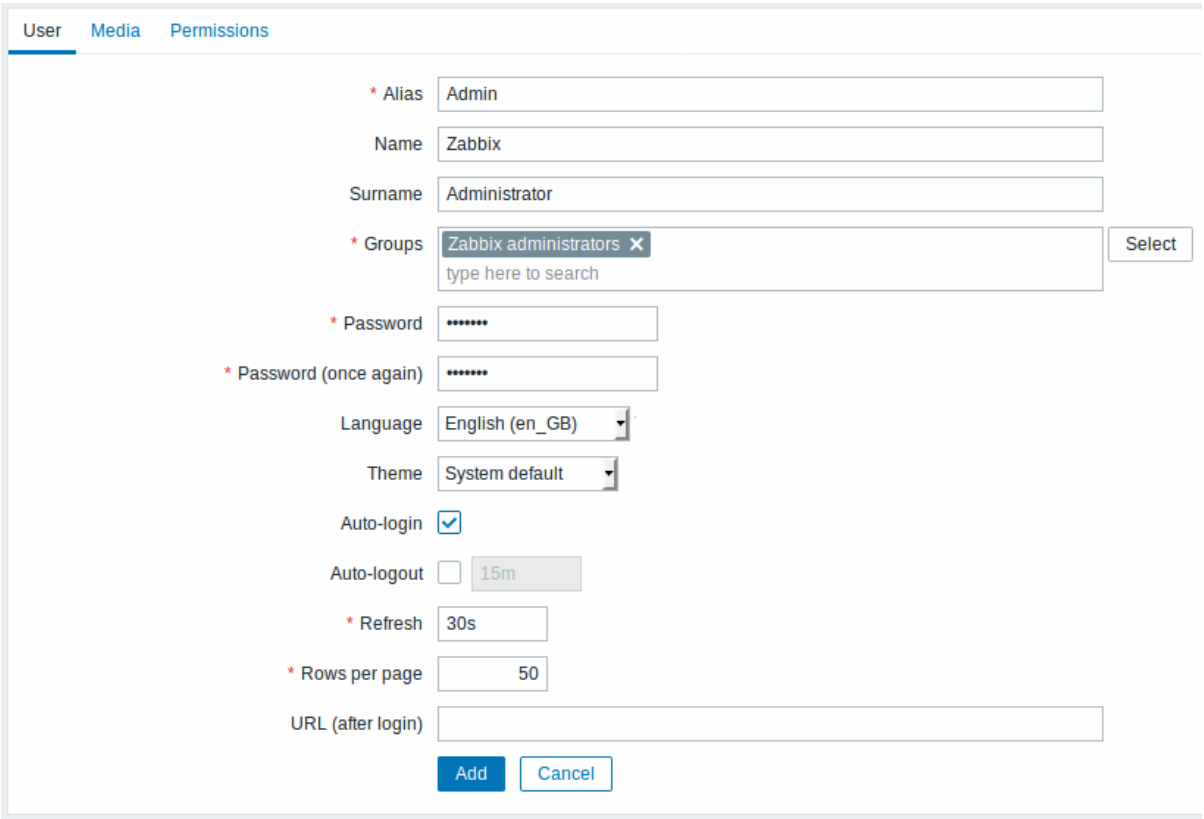
- *Admin* - a Zabbix superuser with full permissions;
- *guest* - a special Zabbix user. The 'guest' user is disabled by default. If you add it to the Guests user group, you may access monitoring pages in Zabbix without being logged in. Note that by default, 'guest' has no permissions on Zabbix objects.

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.

Parameter	Description
Alias	Unique username, used as the login name.
Name	User first name (optional). If not empty, visible in acknowledgment information and notification recipient information.

Parameter	Description
Last name	User last name (optional). If not empty, visible in acknowledgment information and notification recipient information.
Groups	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.
Password	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.
Language	Language of the Zabbix frontend. The php gettext extension is required for the translations to work.
Theme	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
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, screens, 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 media types.

Permissions

The *Permissions* tab contains information on:

- the user type (Zabbix User, Zabbix Admin, Zabbix Super Admin). Users cannot change their own type.
- host groups the user has access to. 'Zabbix User' and 'Zabbix Admin' users do not have access to any host groups and hosts by default. To get access they need to be included in user groups that have access to respective host groups and hosts.

See the User permissions page for details.

2 Permissions

Overview

You can differentiate user permissions in Zabbix by defining the respective user type and then by including the unprivileged users in user groups that have access to host group data.

User type

The user type defines the level of access to administrative menus and the default access to host group data.

User type	Description
Zabbix User	The user has access to the Monitoring menu. The user has no access to any resources by default. Any permissions to host groups must be explicitly assigned.
Zabbix Admin	The user has access to the Monitoring and Configuration menus. The user has no access to any host groups by default. Any permissions to host groups must be explicitly given.
Zabbix Super Admin	The user has access to everything: Monitoring, Configuration and Administration menus. The user has a read-write access to all host groups. Permissions cannot be revoked by denying access to specific host groups.

Permissions to host groups

Access to any host data in Zabbix are granted to user groups on host 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.

3 User groups

Overview

User groups allow to group users both for organizational purposes and for assigning permissions to data. Permissions to monitoring data of host 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 groups.

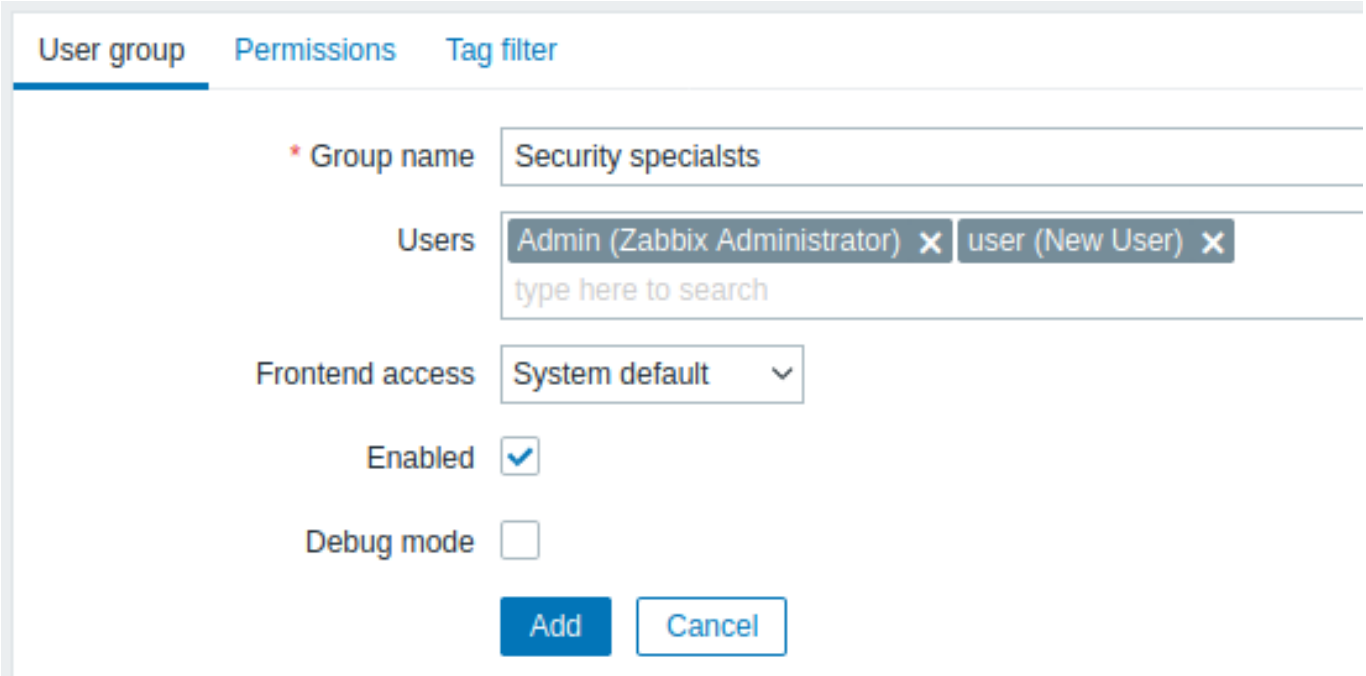
A user can belong to any number of groups.

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

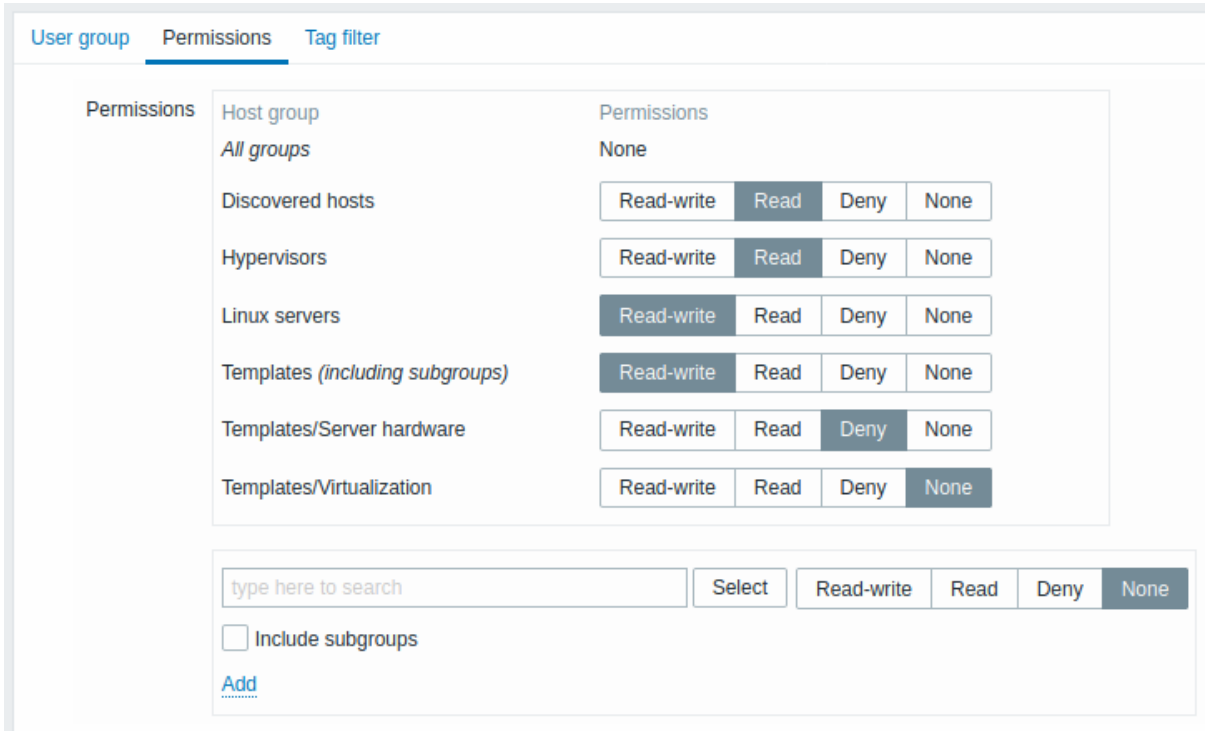
The **User group** tab contains general group attributes:



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Group name	Unique group name.
Users	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.
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
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.

The **Permissions** tab allows you to specify user group access to host group (and thereby host) data:



Current permissions to host groups are displayed in the *Permissions* block.

If current permissions of the host group are inherited by all nested host groups, this is indicated after the host group name (”*including subgroups*”). Note that a Zabbix *Super admin* user can enforce nested host groups to have the same level of permissions as the parent host group; this can be done in the host group configuration form.

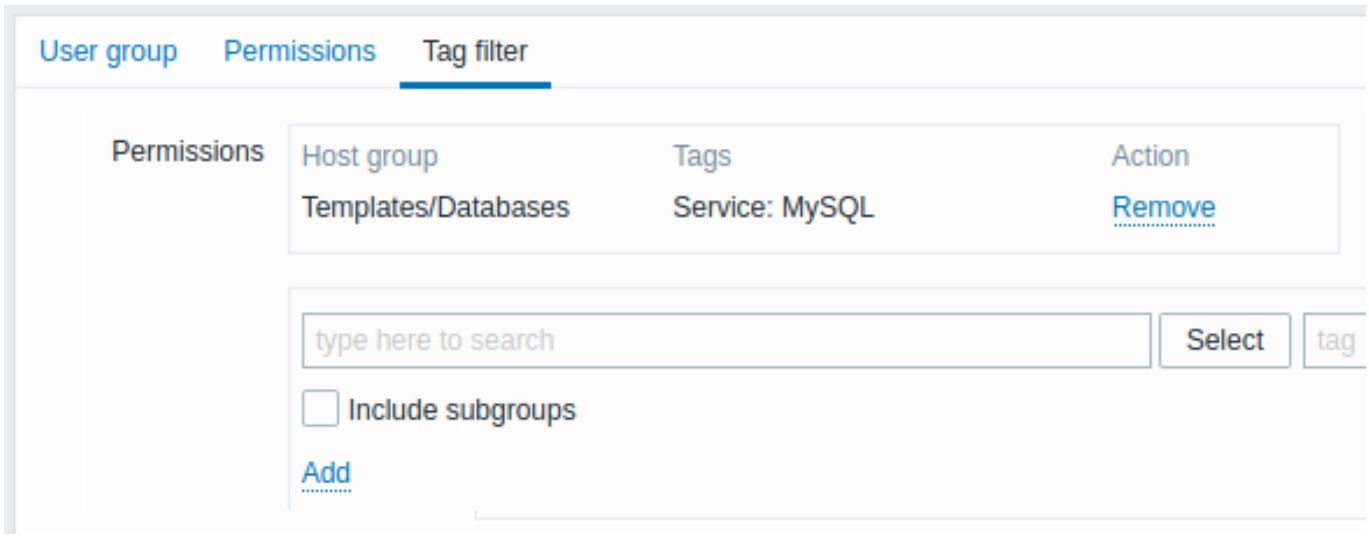
You may change the level of access to a host group:

- **Read-write** - read-write access to a host group;
- **Read** - read-only access to a host group;
- **Deny** - access to a host group denied;
- **None** - no permissions are set.

Use the selection field below to select host groups and the level of access to them. This field is auto-complete so starting to type the name of a host group will offer a dropdown of matching host groups. If you wish to see all host groups, click on *Select*. If you wish to include nested host groups, mark the *Include subgroups* checkbox. Click on Add to add the selected host groups to the list of host group permissions.

Attention:
Adding a parent host group with the *Include subgroups* checkbox marked will override (and remove from the list) previously configured permissions of all related nested host groups. Adding a host group with *None* as the level of access selected will remove the host group from the list if the host group is already in the list.

The **Tag filter** tab allows you to set tag based permissions for user groups to see problems filtered by tag name and its 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. If you want 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, corresponding user group will see all problems for selected host group with tag name "Service". If both tag name and value fields are left blank but host group selected, corresponding user group will see all problems for selected host group. Make sure a tag name and tag value are correctly specified otherwise a corresponding 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.

User group A			**User group B**			**Visible result for a user (member) of both groups**
Tag filter						
Host group	*Tag name*	*Tag value*	*Host group*	*Tag name*	*Tag value*	
Templates/Databases	Service	MySQL	Templates/Databases	Service	Oracle	Service: MySQL or Oracle problems visible
Templates/Databases	*blank*	*blank*	Templates/Databases	Service	Oracle	All problems visible
not selected	*blank*	*blank*	Templates/Databases	Service	Oracle	Service:Oracle problems visible

Attention:
Adding a filter (for example, all tags in a certain host group "Templates/Databases") results in not being able to see the problems of other host groups.

Host access from several user groups

A user may belong to any number of user groups. These groups may have different access permissions to hosts.

Therefore, it is important to know what hosts 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'.

Attention:
"Read-write" permissions have precedence over "Read" permissions starting with Zabbix 2.2.

- 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 Hostgroup 1, the user will get **Read-write** access to 'X'.
- If Group A has *Deny* access to Hostgroup 1 and Group B has a *Read-write* access to Hostgroup 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 *Templates* group. With *Read* access to *Templates* 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-Zabbix 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. The same applies to screens and slideshows as well. The users, regardless of permissions, will see any objects that are not directly or indirectly linked to hosts.
- Zabbix server will not send notifications to users defined as action operation recipients if access to the concerned host is explicitly "denied".

8. Service monitoring

Overview Service monitoring functionality is intended for those who want to get a high-level (business) view of monitored infrastructure. In many cases, we are not interested in low-level details, like the lack of disk space, high processor load, etc. What we are interested in is the availability of service provided by our IT department. We can also be interested in identifying weak places of IT infrastructure, SLA of various IT services, the structure of existing IT infrastructure, and other information of a higher level.

Zabbix service monitoring provides answers to all mentioned questions.

Services is a hierarchy representation of monitored data.

A very simple service structure may look like:

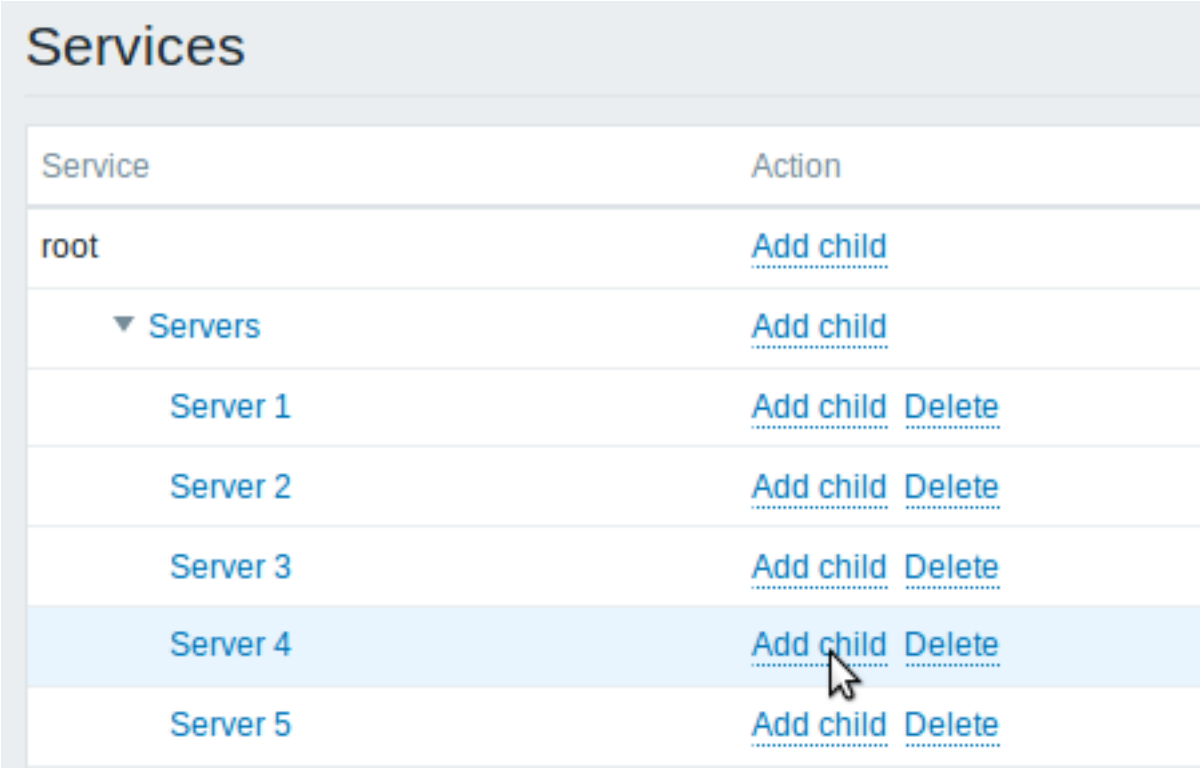
```
Service
|
|-Workstations
| |
| |-Workstation1
| |
| |-Workstation2
|
|-Servers
```

Each node of the structure has attribute status. The status is calculated and propagated to upper levels according to the selected algorithm. At the lowest level of services are triggers. The status of individual nodes is affected by the status of their triggers.

> **Note:**
> Note that triggers with a *Not classified* or *Information* severity do not impact SLA calculation.

Configuration To configure services, go to: *Configuration* → *Services*.

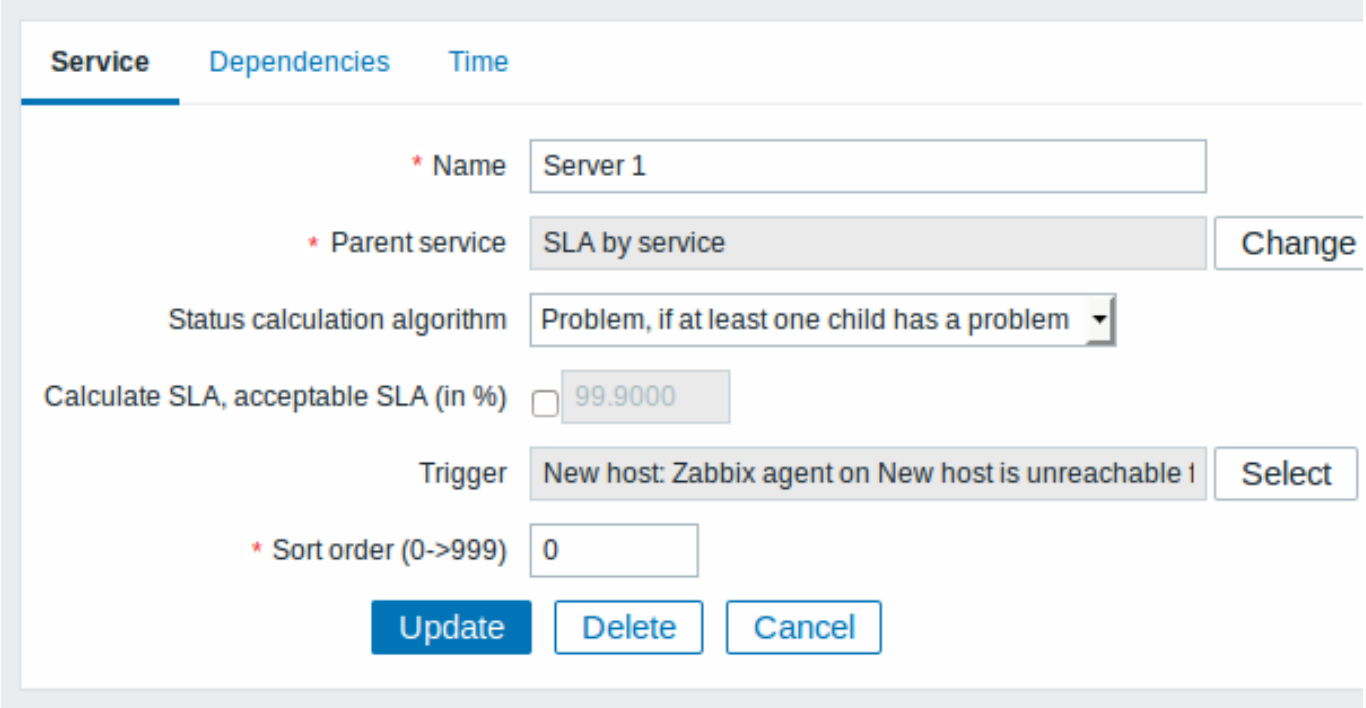
On this screen you can build a hierarchy of your monitored infrastructure. The highest-level parent service is 'root'. You can build your hierarchy downward by adding lower-level parent services and then individual nodes to them.



Click on *Add child* to add services. To edit an existing service, click on its name. A form is displayed where you can edit the service attributes.

Configuring a service

The **Service** tab contains general service attributes:

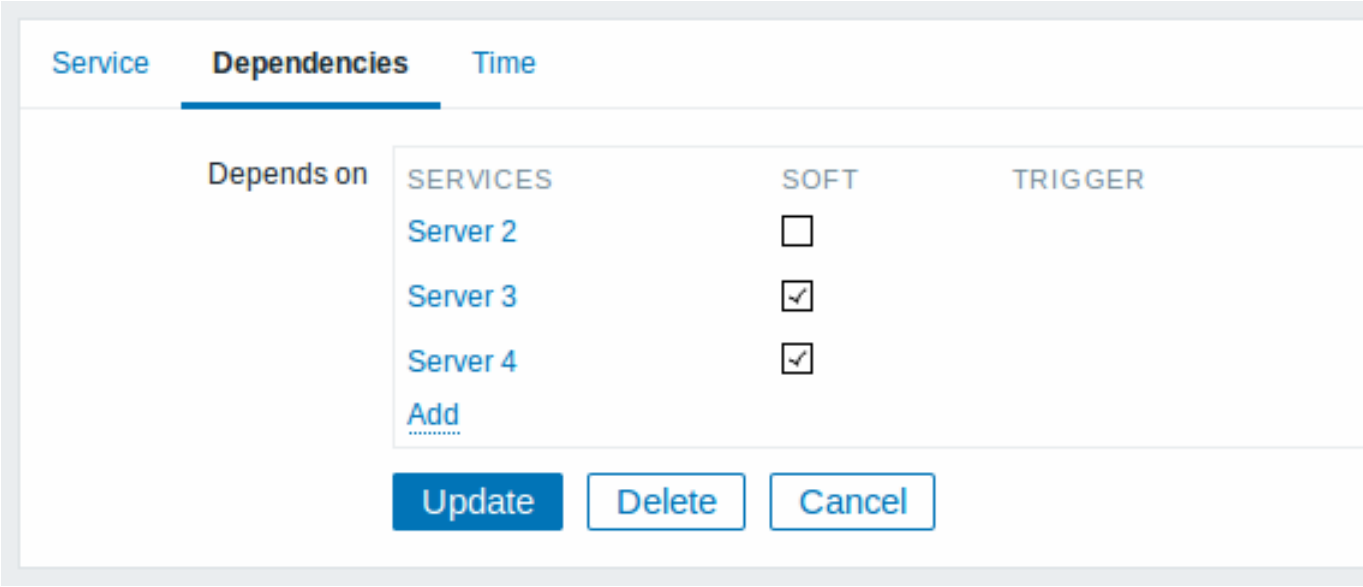


All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Service name.
Parent service	Parent service the service belongs to.

Parameter	Description
Status calculation algorithm	Method of calculating service status: **Do not calculate** - do not calculate service status **Problem, if at least one child has a problem** - problem status, if at least one child service has a problem **Problem, if all children have problems** - problem status, if all child services are having problems
Calculate SLA	Enable SLA calculation and display.
Acceptable SLA (in %)	SLA percentage that is acceptable for this service. Used for reporting.
Trigger	Linkage to trigger: **None** - no linkage **trigger name** - linked to the trigger, thus depends on the trigger status Services of the lowest level must be linked to triggers. (Otherwise their state will not be represented accurately.) When triggers are linked, their state prior to linking is not counted.
Sort order	Sort order for display, lowest comes first.

The **Dependencies** tab contains services the service depends on. Click on *Add* to add a service from those that are configured.

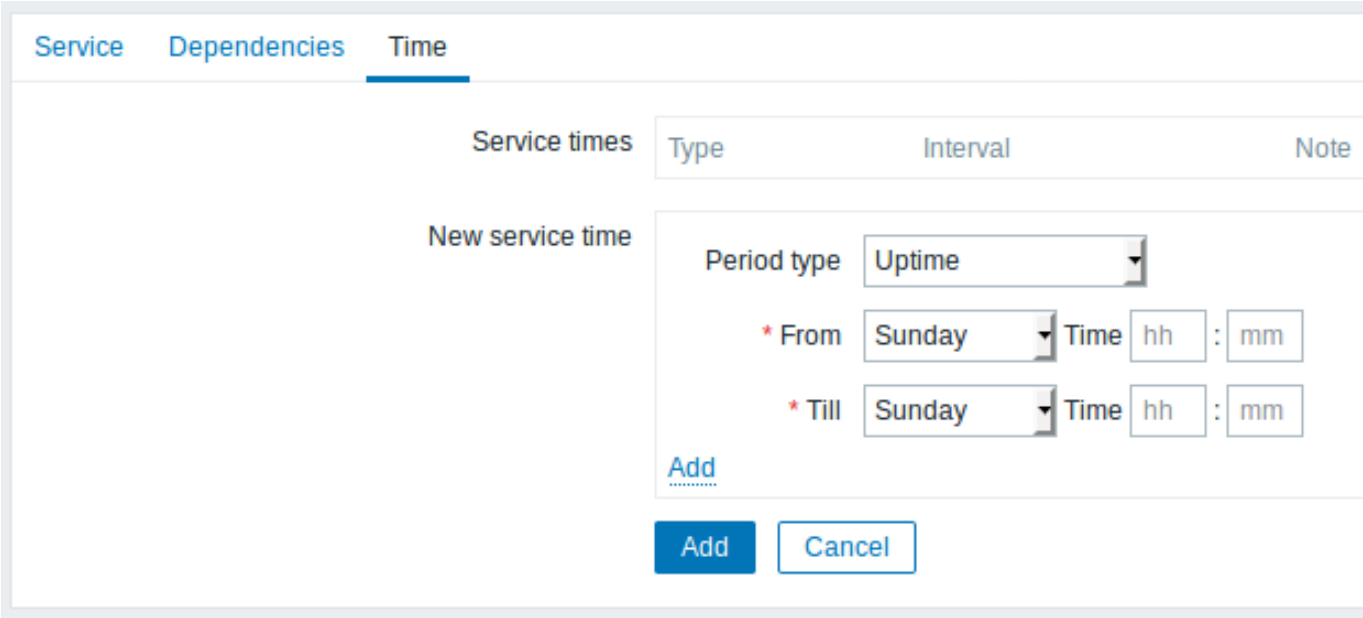


Hard and soft dependency

Availability of a service may depend on several other services, not just one. The first option is to add all those directly as child services.

However, if some service is already added somewhere else in the services tree, it cannot be simply moved out of there to a child service here. How to create a dependency on it? The answer is ”soft” linking. Add the service and mark the *Soft* check box. That way the service can remain in its original location in the tree, yet be depended upon from several other services. Services that are ”soft-linked” are displayed in gray in the tree. Additionally, if a service has only ”soft” dependencies, it can be deleted directly, without deleting child services first.

The **Time** tab contains the service time specification.



Parameter	Description
Service times	By default, all services are expected to operate 24x7x365. If exceptions needed, add new service times.
New service time	Service times: **Uptime** - service uptime **Downtime** - service state within this period does not affect SLA. **One-time downtime** - a single downtime. Service state within this period does not affect SLA. Add the respective hours. *Note*: Service times affect only the service they are configured for. Thus, a parent service will not take into account the service time configured on a child service (unless a corresponding service time is configured on the parent service as well). Service times are taken into account when calculating service status and SLA by the frontend. However, information on service availability is being inserted into database continuously, regardless of service times.

Display To monitor services, go to *Monitoring → Services*.

9. Web monitoring

Overview With Zabbix you can check several availability aspects of web sites.

Attention:
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.

Since Zabbix 2.2 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.

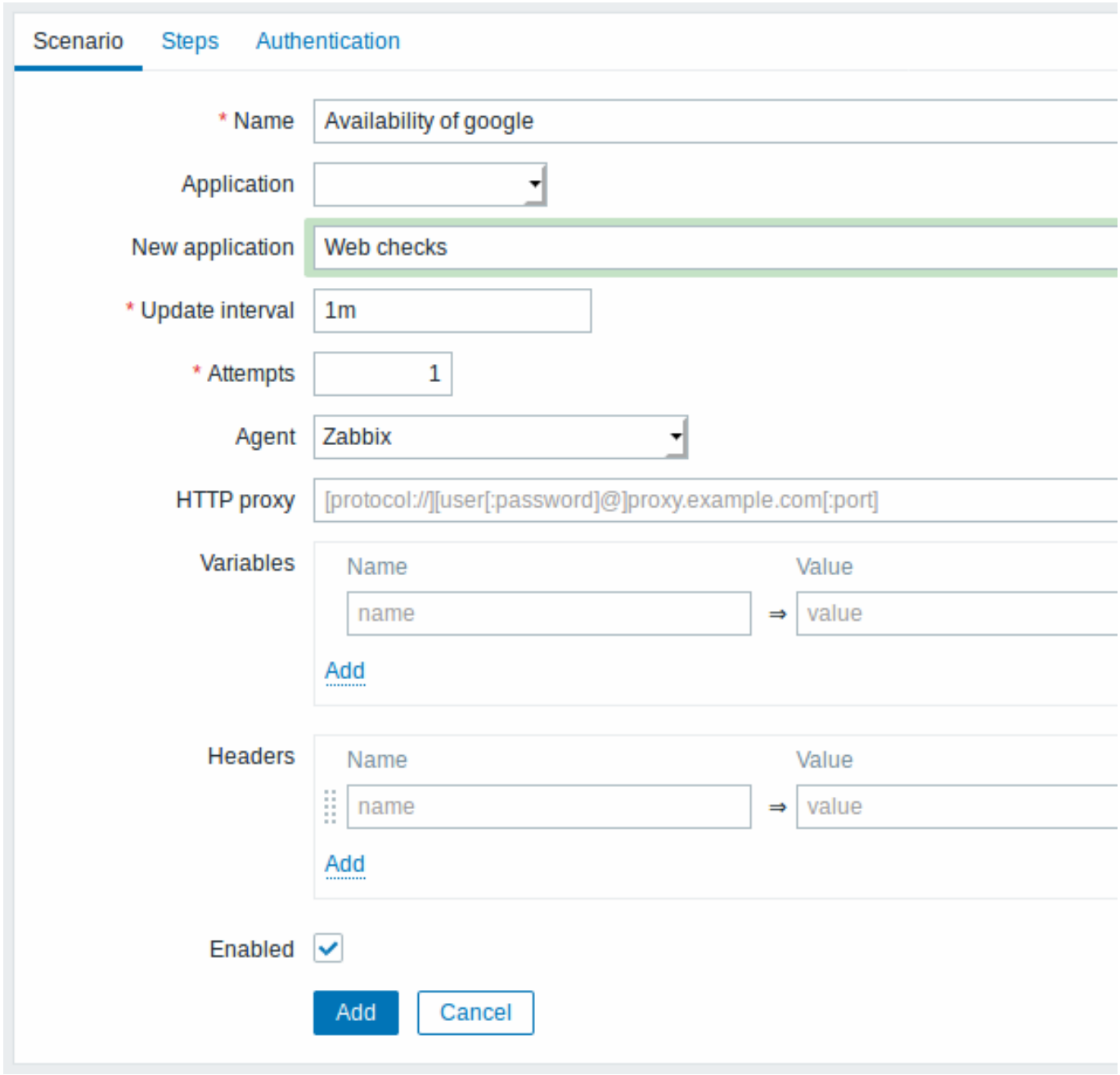
Zabbix can also check if a retrieved HTML page contains a pre-defined string. It can execute a simulated login and follow a path of simulated mouse clicks on the page.

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.

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 web 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.



All mandatory input fields are marked with a red asterisk.

Note:
User macros are supported in many web scenario parameters, but secret macros should not be used in URLs as they will resolve as "******" .

Scenario parameters:

Parameter	Description
Name	Unique scenario name.
Application	Select an application the scenario will belong to. Web scenario items will be grouped under the selected application in *Monitoring* → *Latest data*.
New application	Enter the name of a new application for the scenario.
Update interval	How often the scenario will be executed. Time suffixes are supported, e.g. 30s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. *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).

Parameter	Description
Attempts	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. This parameter is supported starting with *Zabbix 2.2*.
Agent	Select a client agent. Zabbix will pretend to be the selected browser. This is useful when a website returns different content for different browsers. User macros can be used in this field, *starting with Zabbix 2.2*.
HTTP proxy	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 "as is", 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. This parameter is supported starting with *Zabbix 2.2*.
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:hostid is ([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. Regular expression match in variables is supported *since Zabbix 2.2*. User macros and {HOST.*} macros are supported, since Zabbix 2.2. 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.

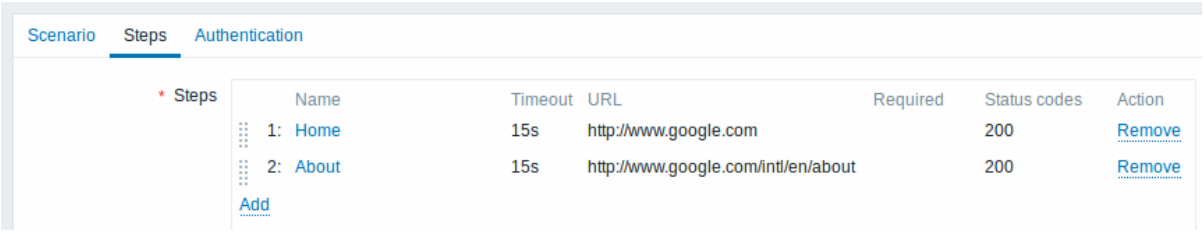
Parameter	Description
Headers	HTTP Headers are used when performing a request. Default and custom headers can be used. Headers will be assigned using default settings depending on the Agent type selected from a drop-down list on a scenario level, and will be applied to all the steps, unless they are custom defined on a step level. **It should be noted that defining the header on a step level automatically discards all the previously defined headers, except for a default header that is assigned by selecting the 'User-Agent' from a drop-down list on a scenario level.** However, even the 'User-Agent' default header can be overridden by specifying it on a step level. To unset the header on a scenario level, the header should be named and attributed with no value on a step level. 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. Specifying custom headers is supported *starting with Zabbix 2.4*.
Enabled	The scenario is active if this box is checked, otherwise - disabled.

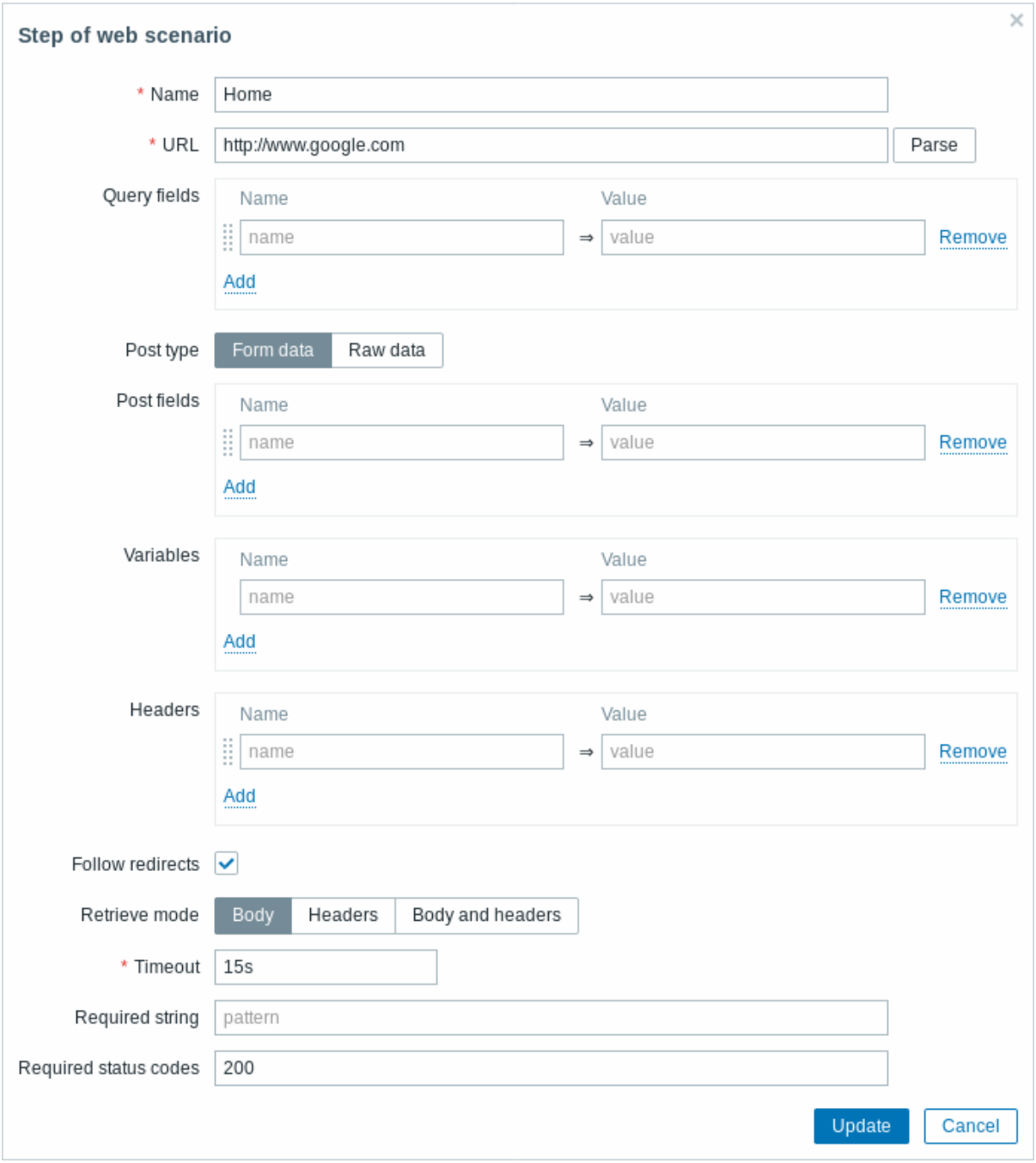
Note that when editing an existing scenario, two extra buttons are available in the form:

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.

Note:
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.





Configuring steps

Step parameters:

Parameter	Description
Name	Unique step name.
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 since Zabbix 3.4. 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, since Zabbix 2.2. Limited to 2048 characters *starting with Zabbix 2.4*.

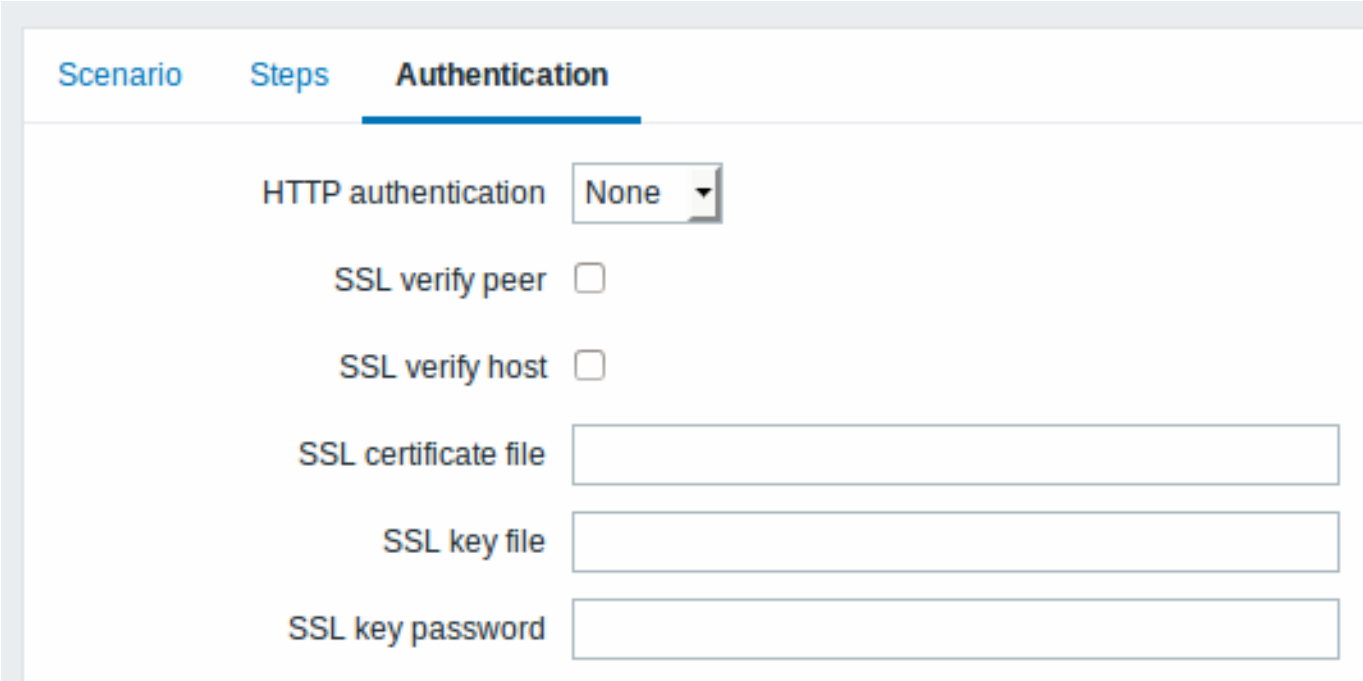
Parameter	Description
Query fields	HTTP GET variables for the URL. Specified as attribute and value pairs. Values are URL-encoded automatically. Values from scenario variables, user macros or {HOST.*} macros are resolved and then URL-encoded automatically. Using a {{macro}.urlencode()} syntax will double URL-encode them. User macros and {HOST.*} macros are supported since Zabbix 2.2.
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, since Zabbix 2.2.
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. Having step-level variables is supported since Zabbix 2.2. 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. A header defined on a step level will be used for that particular step. **It should be noted that defining the header on a step level automatically discards all the previously defined headers, except for a default header that is assigned by selecting the 'User-Agent' from a drop-down list on a scenario level.** However, even the 'User-Agent' default header can be overridden by specifying it on a step level. For example, assigning the name to a header, but setting no value will unset the default header on a scenario level. User macros and {HOST.*} macros are supported. This sets the CURLOPT_HTTPHEADER cURL option. Specifying custom headers is supported *starting with Zabbix 2.4*.
Follow redirects	Mark the checkbox to follow HTTP redirects. This sets the CURLOPT_FOLLOWLOCATION cURL option. This option is supported *starting with Zabbix 2.4*.
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 This option is supported *since Zabbix 4.2*.

Parameter	Description
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.
Required string	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.*admin *Note*: Referencing regular expressions created in the Zabbix frontend is not supported in this field. User macros and {HOST.*} macros are supported, since Zabbix 2.2.
Required status codes	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 since Zabbix 2.2.

Note:
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 authentication The **Authentication** tab allows you to configure scenario authentication options.



Authentication parameters:

Parameter	Description
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. 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, *starting with Zabbix 2.2*.
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_VERIFYPEER cURL option. This option is supported *starting with Zabbix 2.4*.
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. This option is supported *starting with Zabbix 2.4*.
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. `HOST.*` macros and user macros can be used in this field. This sets the CURLOPT_SSLCERT cURL option. This option is supported *starting with Zabbix 2.4*.
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. `HOST.*` macros and user macros can be used in this field. This sets the CURLOPT_SSLKEY cURL option. This option is supported *starting with Zabbix 2.4*.
SSL key password	SSL private key file password. User macros can be used in this field. This sets the CURLOPT_KEYPASSWD cURL option. This option is supported *starting with Zabbix 2.4*.

Attention:
[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
```

Note:
Zabbix server picks up changes in certificates without a restart.

Note:
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.

Scenario items

As soon as a scenario is created, Zabbix automatically adds the following items for monitoring, linking them to the selected application.

Item	Description
Download speed for scenario <Scenario>	This item will collect information about the download speed (bytes per second) of the whole scenario, i.e. average for all steps. Item key: web.test.in[Scenario„bps] Type: *Numeric(float)*
Failed step of scenario <Scenario>	This item will display the number of the step that failed on the scenario. If all steps are executed successfully, 0 is returned. Item key: web.test.fail[Scenario] Type: *Numeric(unsigned)*
Last error message of scenario <Scenario>	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. Item key: web.test.error[Scenario] Type: *Character*

The actual scenario name will be used instead of ”Scenario”.

Note:
Web monitoring items are added with a 30 day history and a 90 day trend retention period.

Note:
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:

```
{host:web.test.fail[Scenario].last()}<>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:

```
{host:web.test.error[Scenario].strlen()}>0 and {host:web.test.fail[Scenario].last()}>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:

```
{host:web.test.in[Scenario,,bps].last()}<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, linking them to the selected application.

Item	Description
Download speed for step <Step> of scenario <Scenario>	This item will collect information about the download speed (bytes per second) of the step. Item key: web.test.in[Scenario,Step,bps] Type: *Numeric(float)*
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.

Note:
Web monitoring items are added with a 30 day history and a 90 day trend retention period.

Note:
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:

```
{zabbix:web.test.time[ZABBIX GUI,Login,resp].last()}>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.

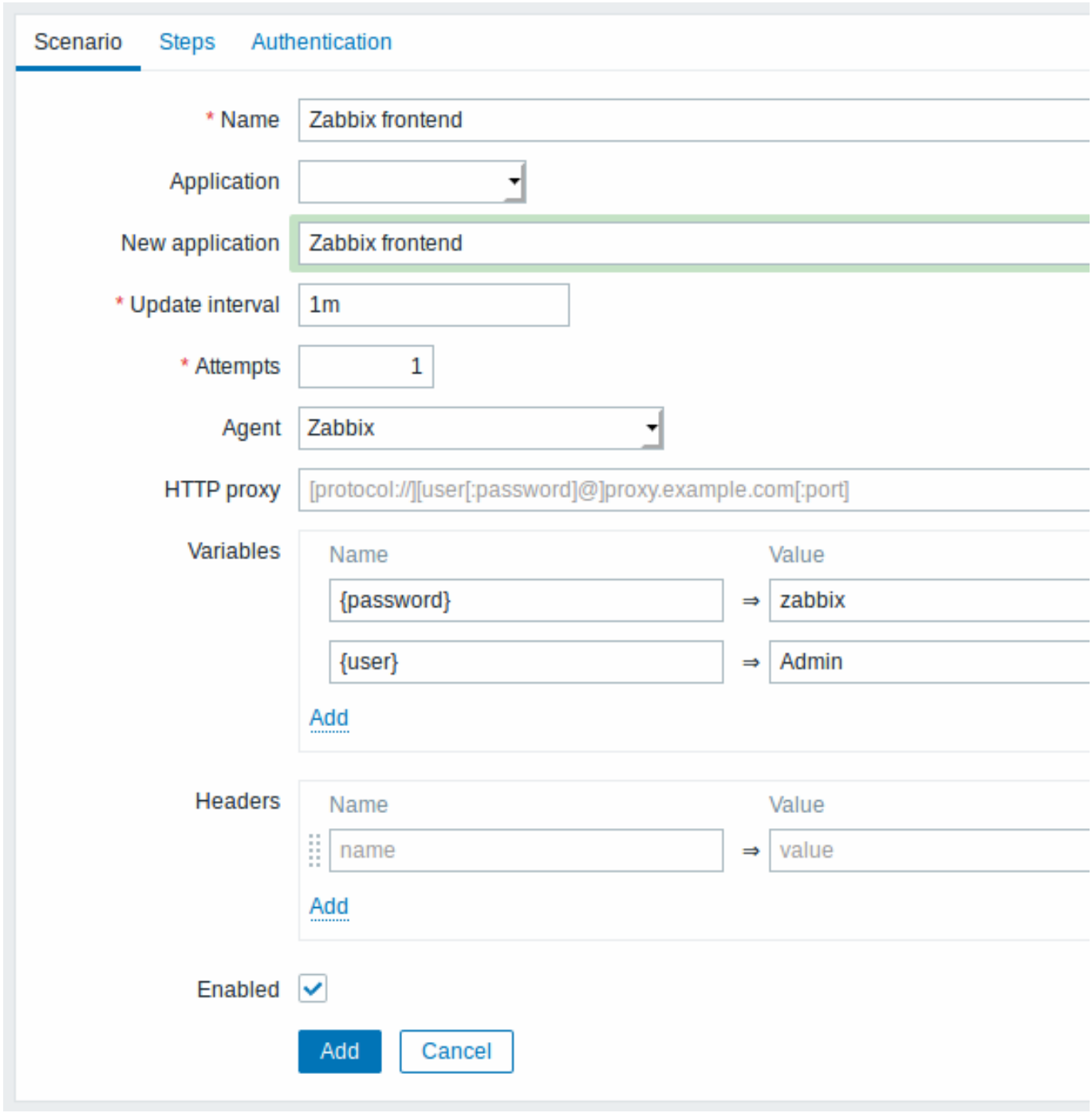
Scenario

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* and create a new *Zabbix frontend* application for it.

Note that we will also create two variables: {user} and {password}.

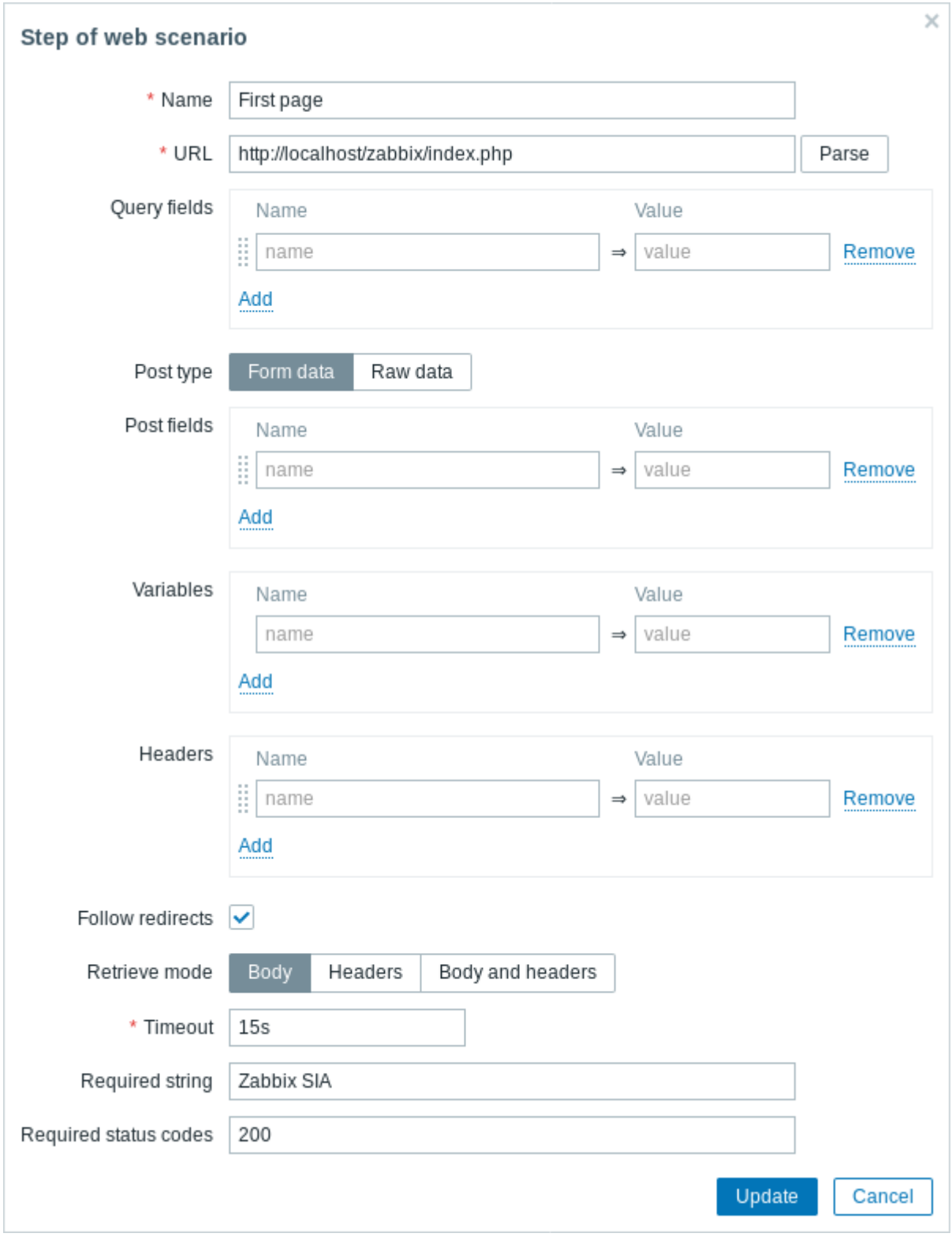
Step 2

Define steps for the scenario.

Click on *Add* button in the *Steps* tab to add individual steps.

Web scenario step 1

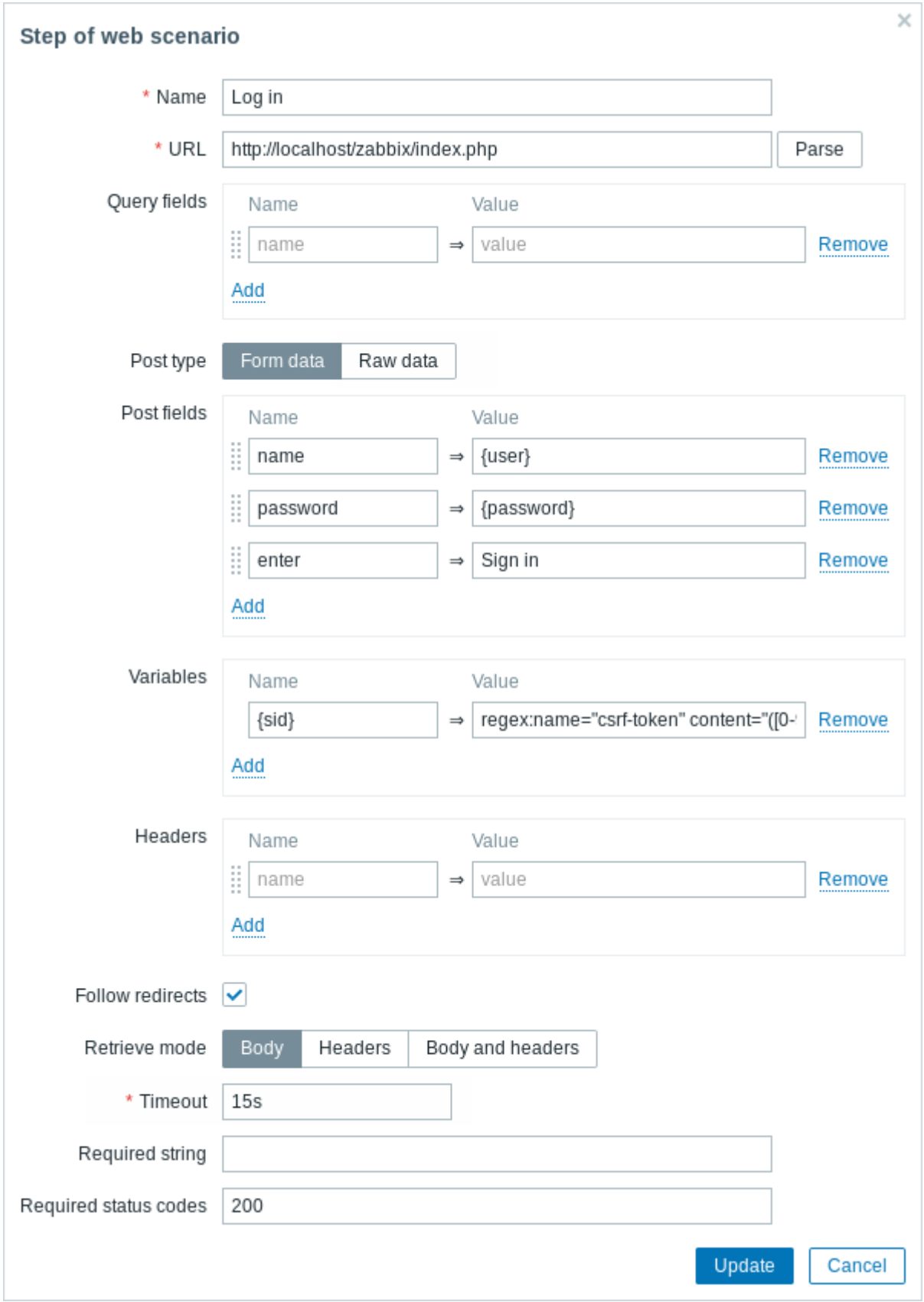
We start by checking that the first page responds correctly, returns with HTTP response code 200 and contains text ”Zabbix SIA”.



When done configuring the step, click on *Add*.

Web scenario step 2

We continue by logging in to the Zabbix frontend, and we do so by reusing the macros (variables) we defined on the scenario level - {user} and {password}.

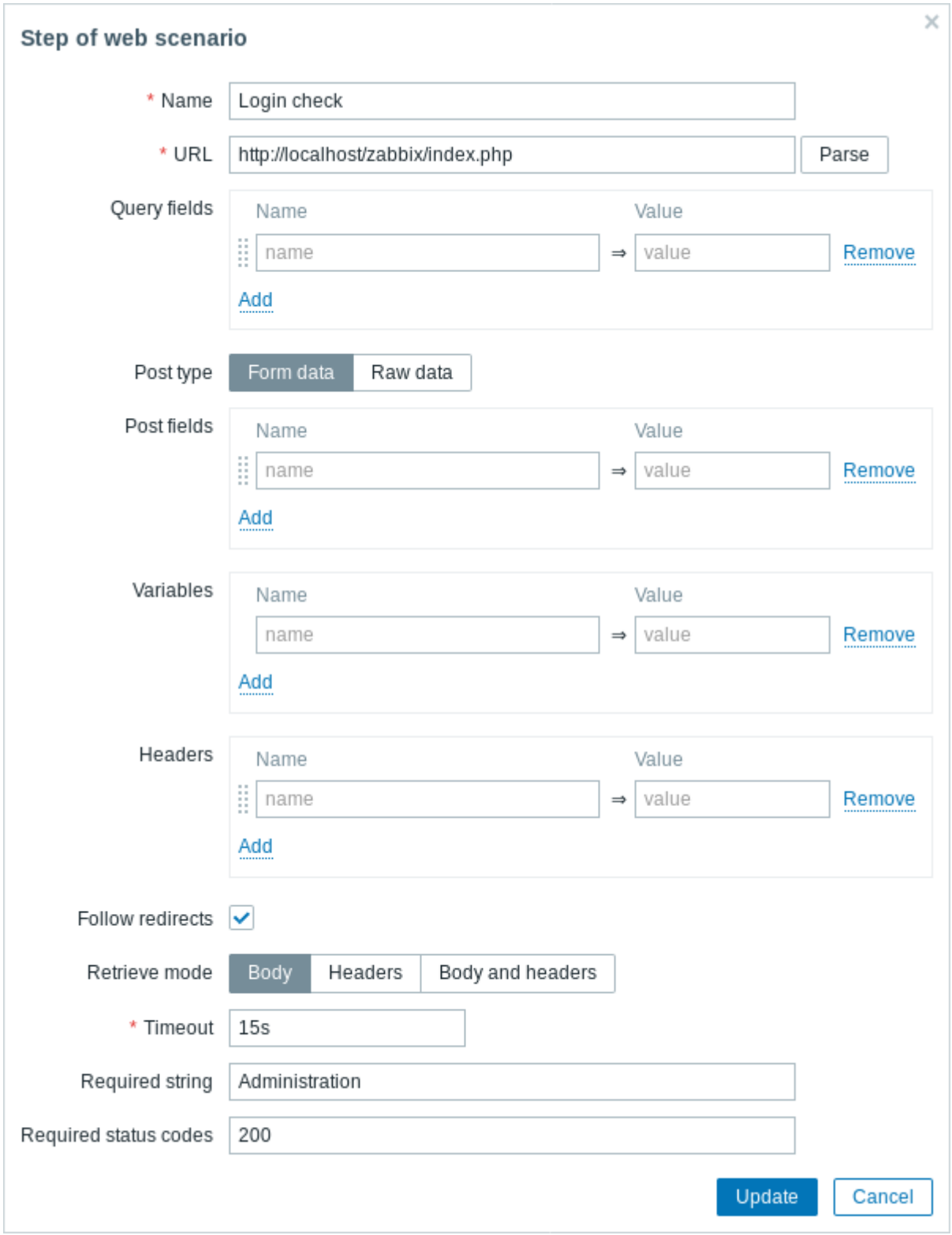


Attention:
Note that Zabbix frontend uses JavaScript redirect when logging in, thus first we must log in, and only in further steps we may check for logged-in features. Additionally, the login step must use full URL to **index.php** file.

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.

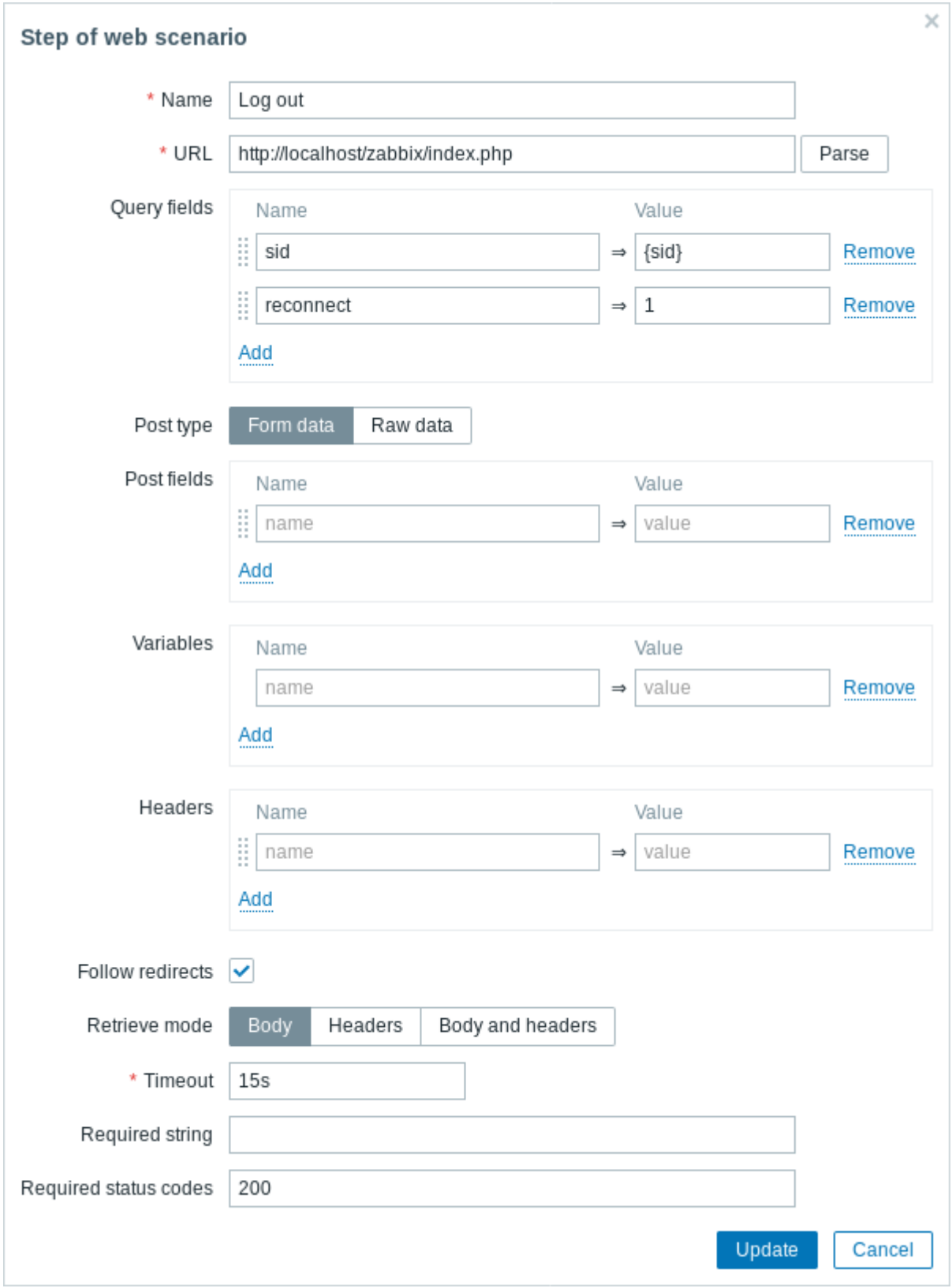
Web scenario step 3

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**.



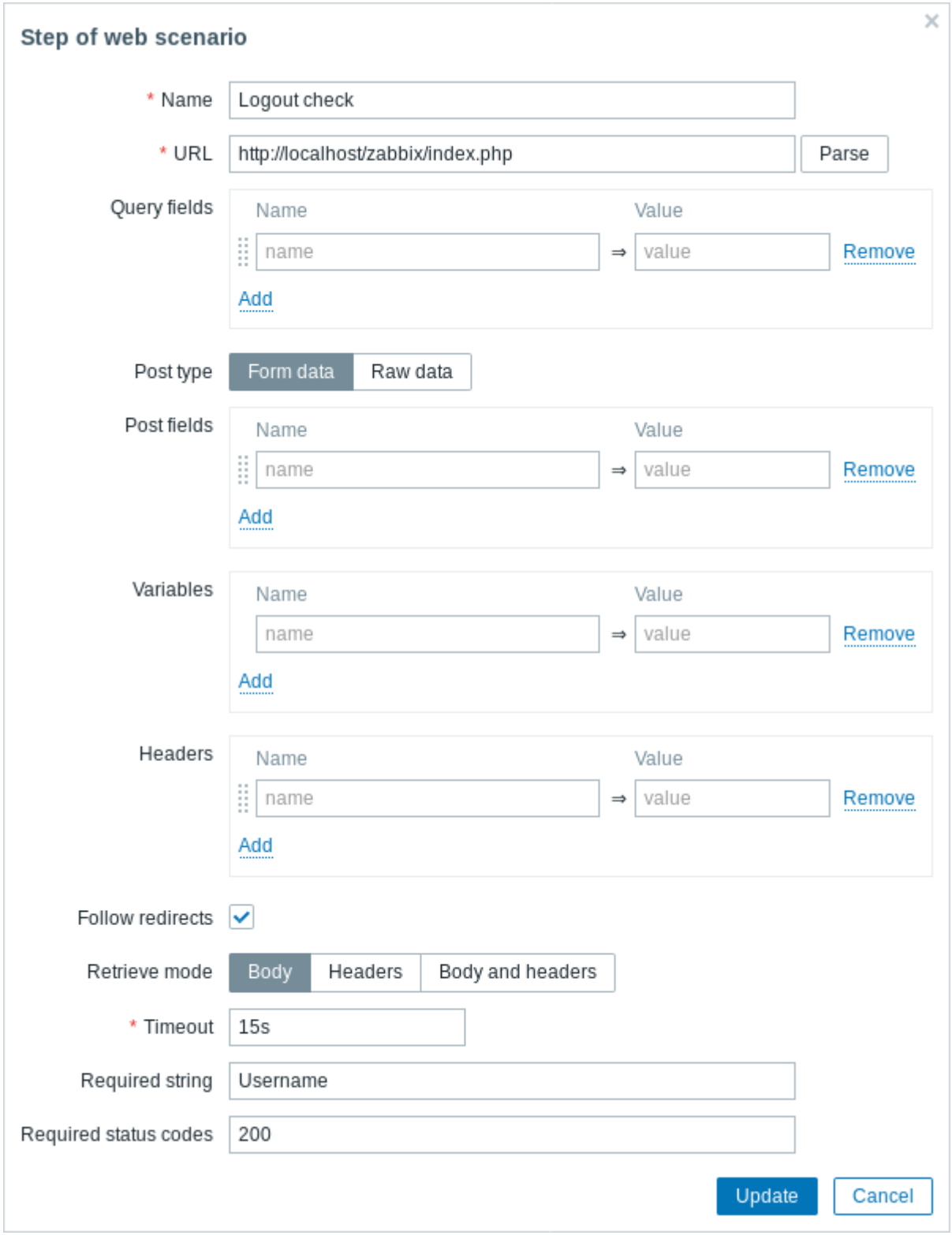
Web scenario step 4

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.



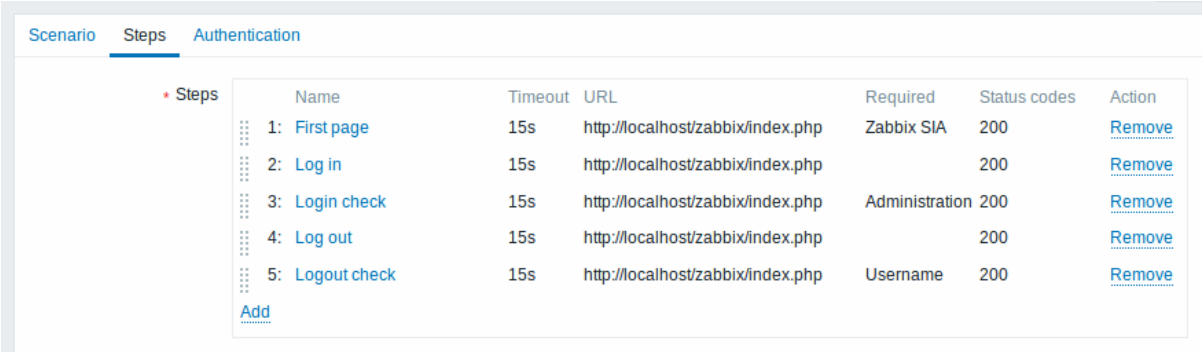
Web scenario step 5

We can also check that we have logged out by looking for the **Username** string.



Complete configuration of steps

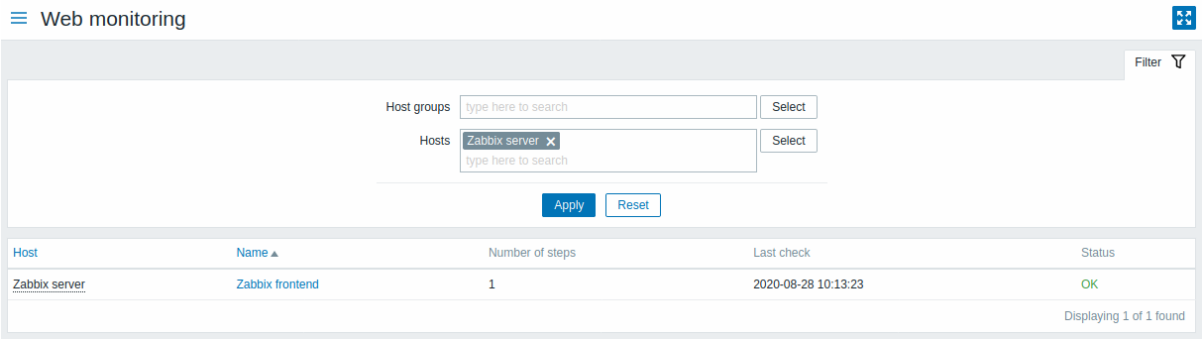
A complete configuration of web scenario steps should look like this:



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 pre-defined 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.

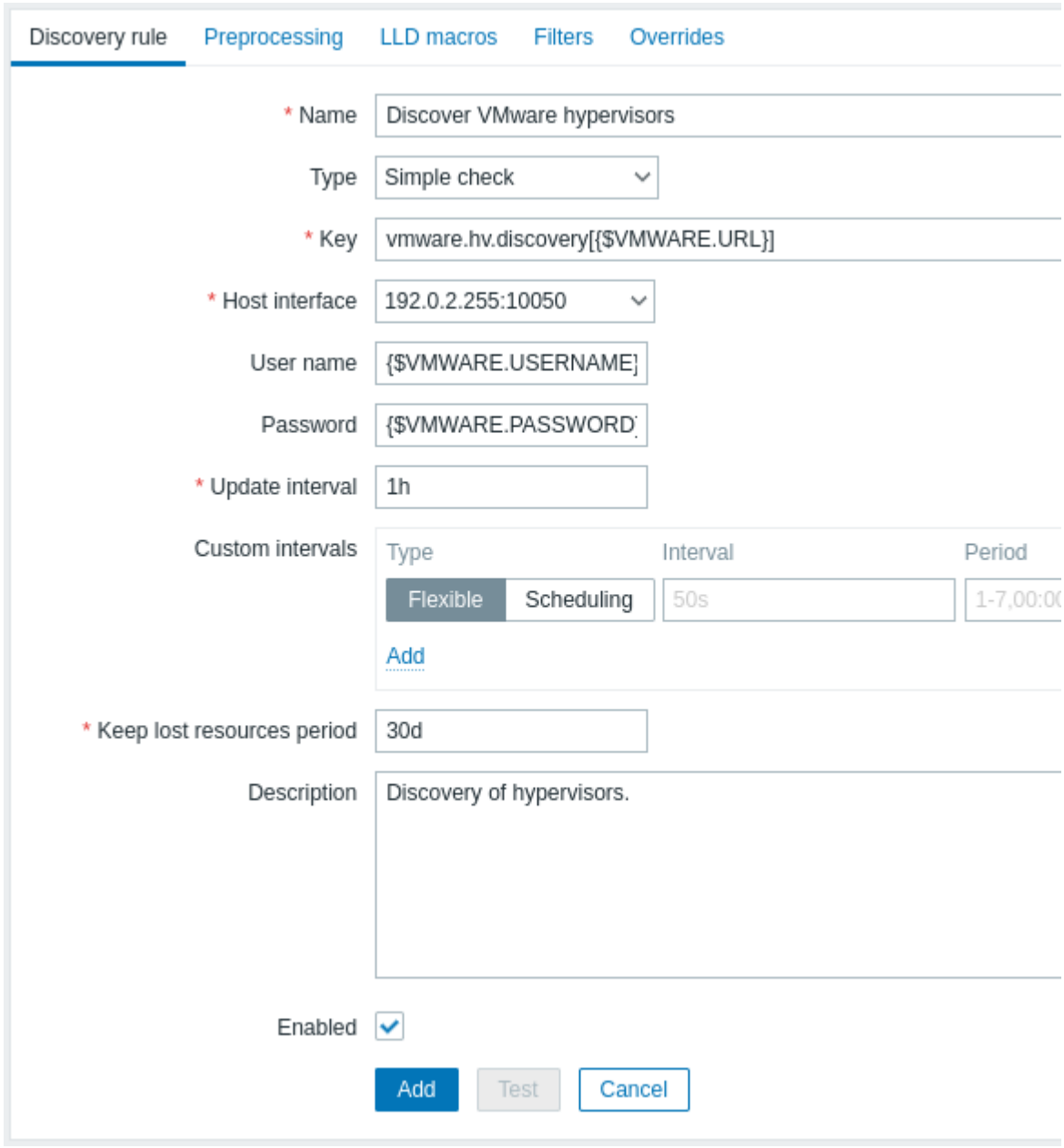
The following configuration file options can be used to tune the Virtual machine monitoring:

- **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:
  *servicenum < StartVMwareCollectors < (servicenum * 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.

> **Attention:**
> 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.

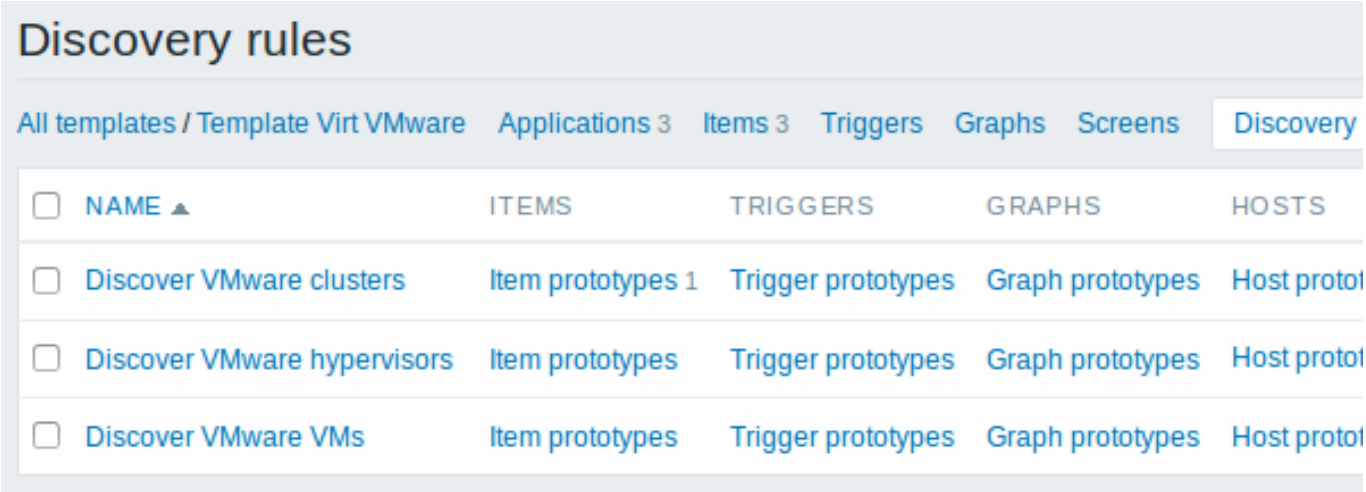
Discovery Zabbix can use a low-level discovery rule to automatically discover VMware hypervisors and virtual machines.



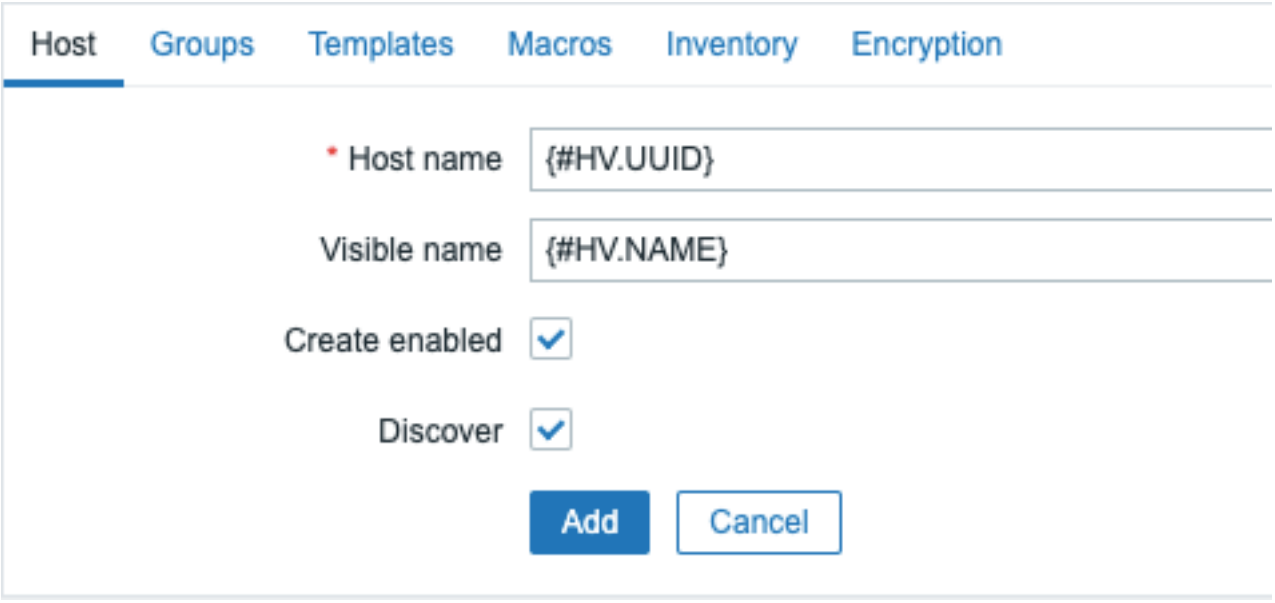
All mandatory input fields are marked with a red asterisk.

Discovery rule key in the above screenshot is *vmware.hv.discovery[{$VMWARE.URL}]*.

Host prototypes Host prototypes can be created with the low-level discovery rule. When virtual machines are discovered, these prototypes become real hosts. Prototypes, before becoming discovered, cannot have their own items and triggers, other than those from the linked templates. Discovered hosts will belong to an existing host and will take the IP of the existing host for the host configuration.



In a host prototype configuration, LLD macros are used for the host name, visible name and host group prototype fields. It is also possible to define user macros with LLD macros as values. Linkage to existing host groups, template linkage and encryption are other options that can be set.



If *Create enabled* is checked, the host will be added in an enabled state. If unchecked, the host will be added, but in disabled state.

If *Discover* is checked (default), the host will be created. If unchecked, the host will not be created, unless this setting is overridden in the discovery rule. This functionality provides additional flexibility when creating discovery rules.

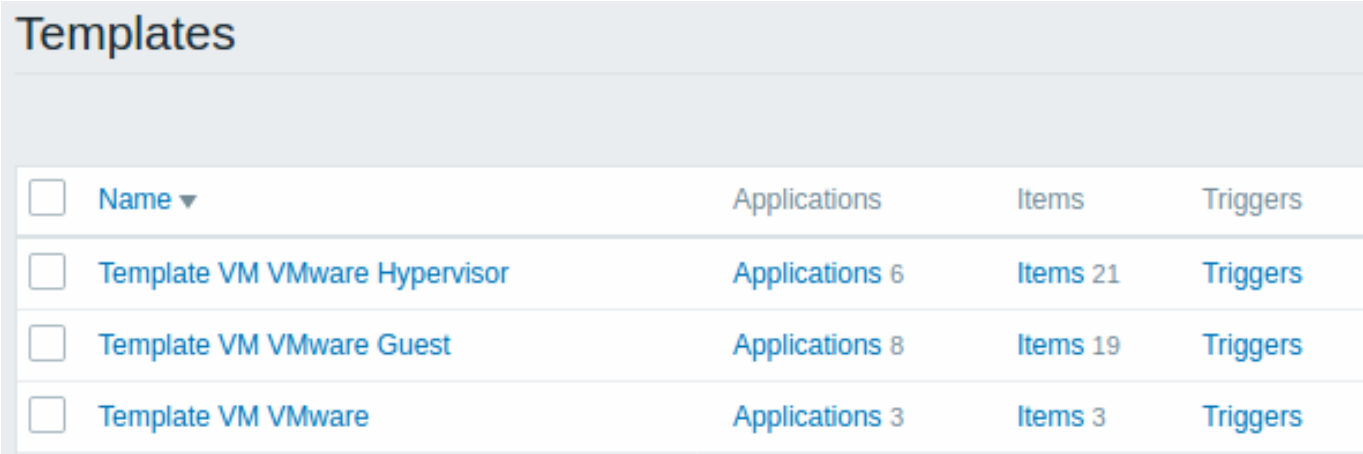
Discovered hosts are prefixed with the name of the discovery rule that created them, in the host list. Discovered hosts can be manually deleted. Discovered hosts will also be automatically deleted, based on the *Keep lost resources period (in days)* value of the discovery rule. Most of the configuration options are read-only, except for enabling/disabling the host and host inventory.

Note:
Zabbix does not support nested host prototypes, i.e. host prototypes are not supported on hosts that are discovered by low-level discovery rule.

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. Note that:

- The *"Template VM VMware"* template should be used for VMware vCenter and ESX hypervisor monitoring;
- The *"Template VM VMware Hypervisor"* and *"Template VM VMware Guest"* templates are used by discovery and normally should not be manually linked to a host.



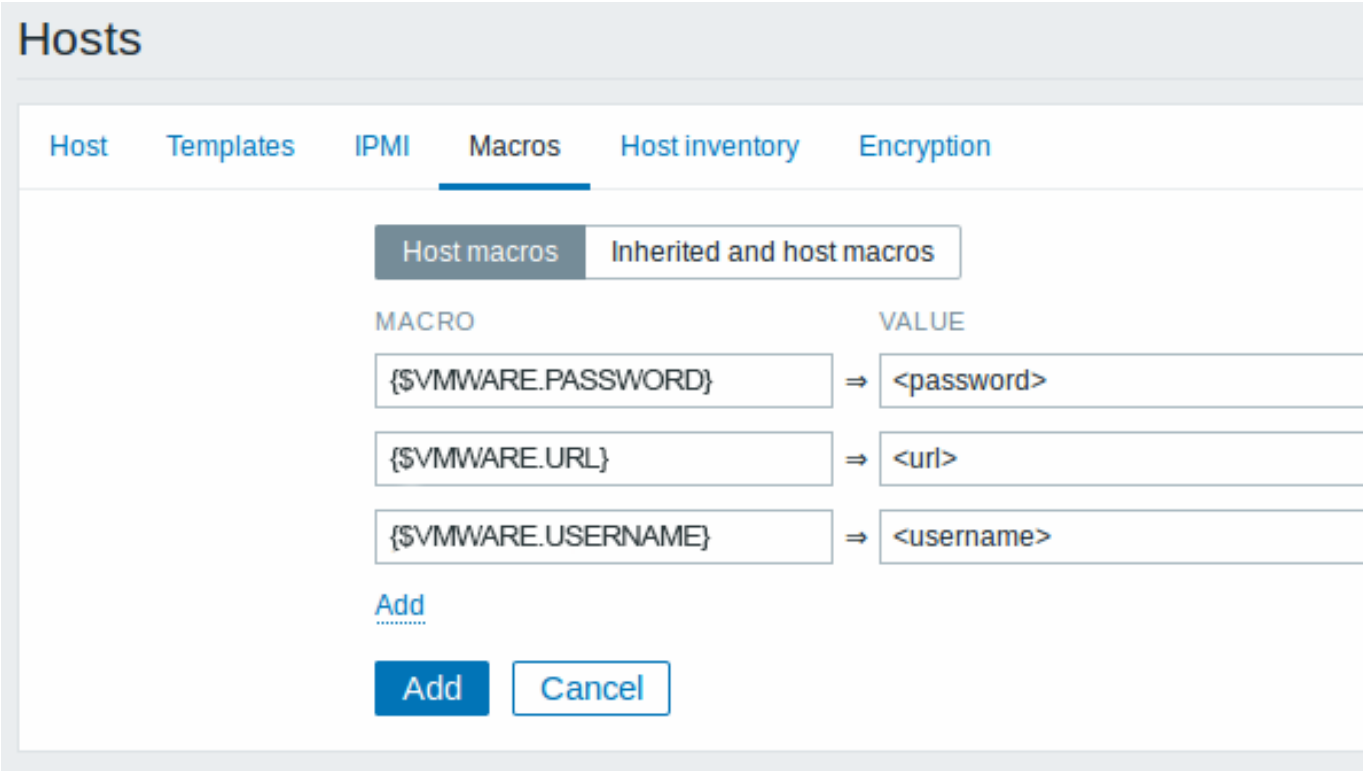
Note:
If your server has been upgraded from a pre-2.2 version and has no such templates, you can import them manually, downloading from the community page with official templates. However, these templates have dependencies from the *VMware VirtualMachinePowerState* and *VMware status* value maps, so it is necessary to create these value maps first (using an SQL script, manually or importing from an XML) before importing the templates.

Host configuration To use VMware simple checks the host must have the following user macros defined:

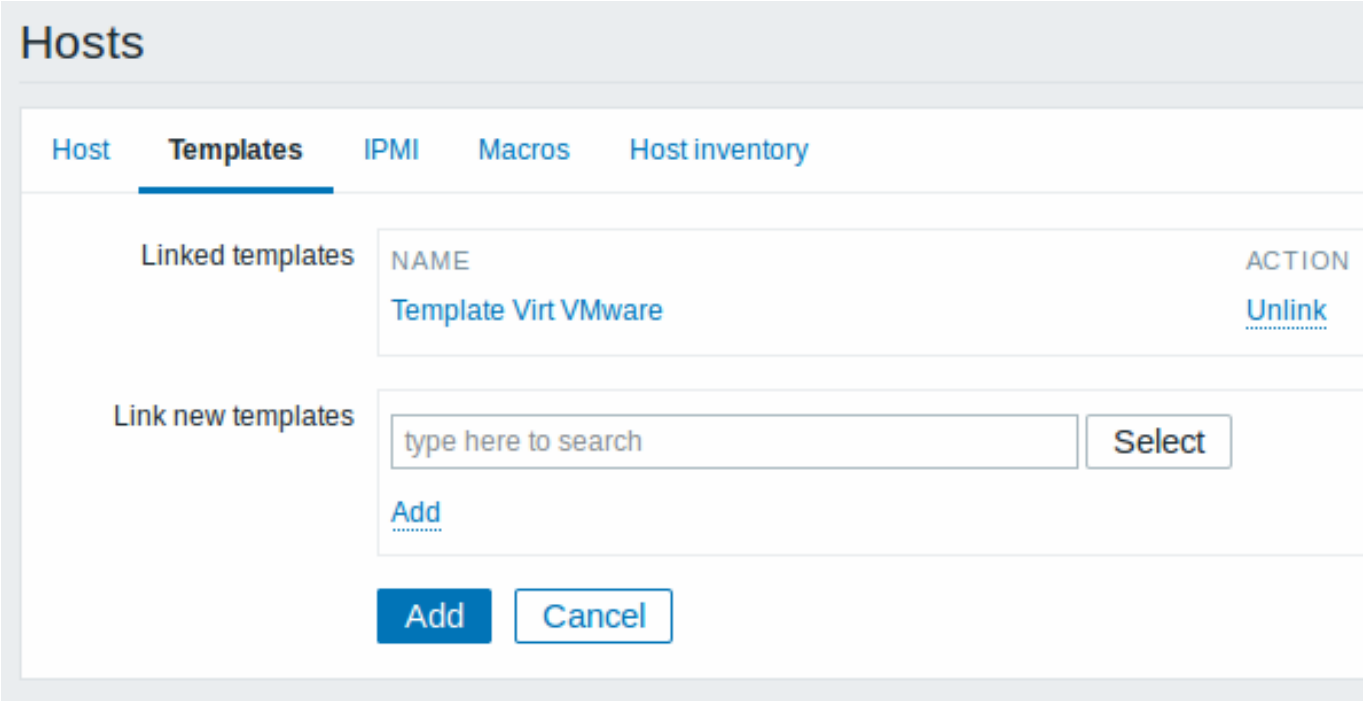
- **{$VMWARE.URL}** - VMware service (vCenter or ESX hypervisor) SDK URL (https://servername/sdk)
- **{$VMWARE.USERNAME}** - VMware service user name
- **{$VMWARE.PASSWORD}** - VMware service {$VMWARE.USERNAME} user password

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:



- link the host to the VMware service template:



- 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.

1 Virtual machine discovery key fields

The following table lists fields returned by virtual machine related discovery keys.

Item key		
Description	**Field**	**Retrieved content**
vmware.cluster.discovery Performs cluster discovery.	{#CLUSTER.ID} {#CLUSTER.NAME}	Cluster identifier. Cluster name.

Item key		
vmware.datastore.discovery Performs datastore discovery.	{#DATASTORE}	Datastore name.
vmware.dc.discovery Performs datacenter discovery (since Zabbix 5.0.3).	{#DATACENTER}	Datacenter name.
	{#DATACENTERID}	Datacenter ID.
vmware.hv.discovery Performs hypervisor discovery.	{#HV.UUID}	Unique hypervisor identifier.
	{#HV.ID}	Hypervisor identifier (HostSystem managed object name).
	{#HV.NAME}	Hypervisor name.
	{#HV.NETNAME}	Hypervisor network host name. Supported since Zabbix 5.0.19.
	{#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.

Item key		
	{#PARENT.TYPE}	Type of container in which the hypervisor is stored. The values could be `Datacenter`, `Folder`, `ClusterComputeRes` `VMware`, where 'VMware' stands for unknown container type. Supported since Zabbix 4.0.3.
vmware.hv.datastore.discovery Performs hypervisor datastore discovery. Note that multiple hypervisors can use the same datastore.	{#DATASTORE}	Datastore name.
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.
	{#CLUSTER.NAME}	Cluster name, might be empty.
	{#DATACENTER.NAME}	Datacenter name.
vmware.vm.net.if.discovery Performs virtual machine network interface discovery.	{#IFNAME}	Network interface name.
vmware.vm.vfs.dev.discovery		

Item key		
Performs virtual machine disk device discovery.	{#DISKNAME}	Disk device name.
vmware.vm.vfs.fs.discovery Performs virtual machine file system discovery.	{#FSNAME}	File system name.

11. Maintenance

Overview You can define maintenance periods for hosts and host groups in Zabbix.

Furthermore, it is possible to define maintenance only for a single trigger (or subset of triggers) by specifying trigger tags. In this case maintenance will be activated only for those triggers; all other triggers of the host or host group will not be in maintenance.

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.

Note:
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.

Note that the Zabbix server (or proxy) always collects 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.

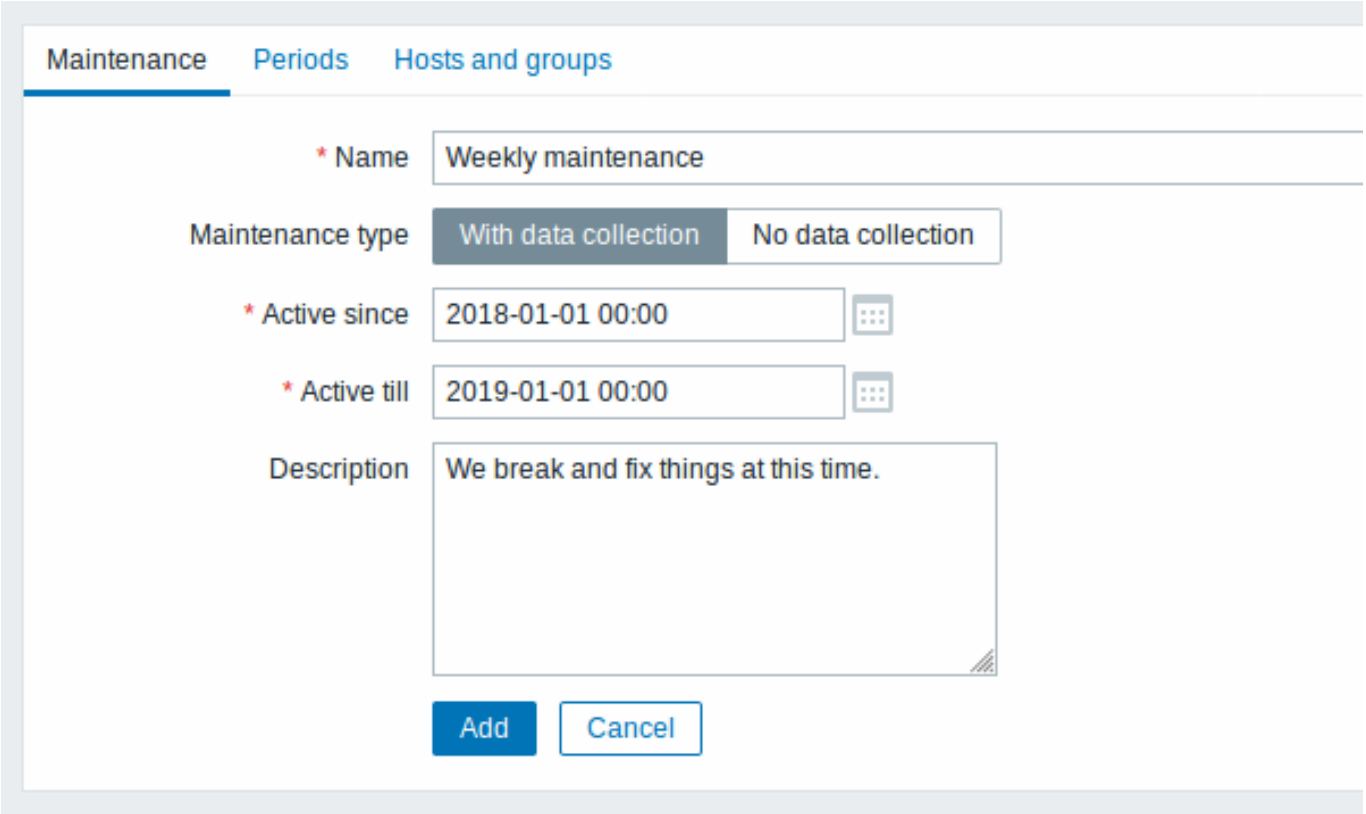
Attention:
To ensure predictable behavior of recurring maintenance periods (daily, weekly, monthly), it is required to use a common time zone for all parts of Zabbix.

If maintenance period, hosts, groups or tags are changed by 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)

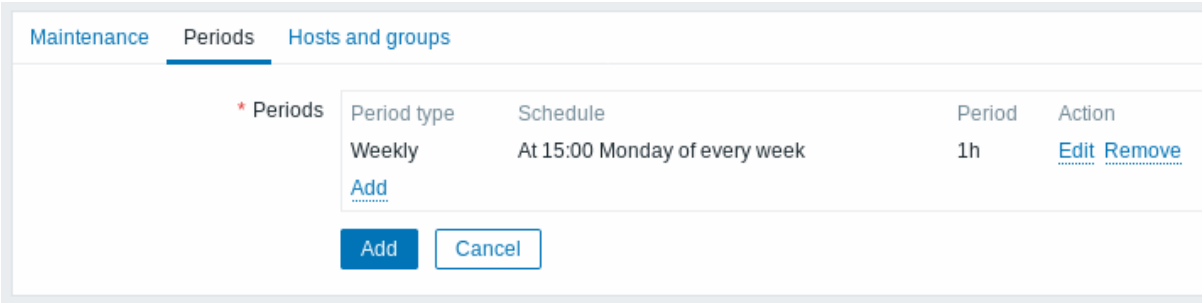
The **Maintenance** tab contains general maintenance period attributes:



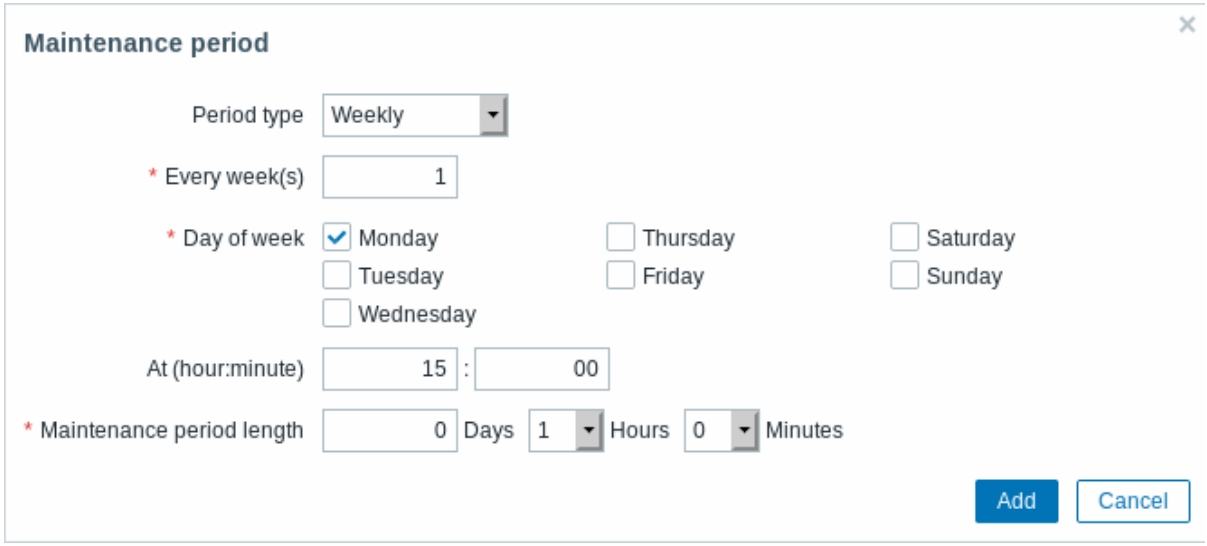
All mandatory input fields are marked with a red asterisk.

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.
Description	Description of maintenance period.

The **Periods** tab allows you to define the exact days and hours when the maintenance takes place.



Clicking on Add in the Periods block opens a popup window with a flexible *Maintenance period* form where you can define the times - for daily, weekly, monthly or one-time maintenance.

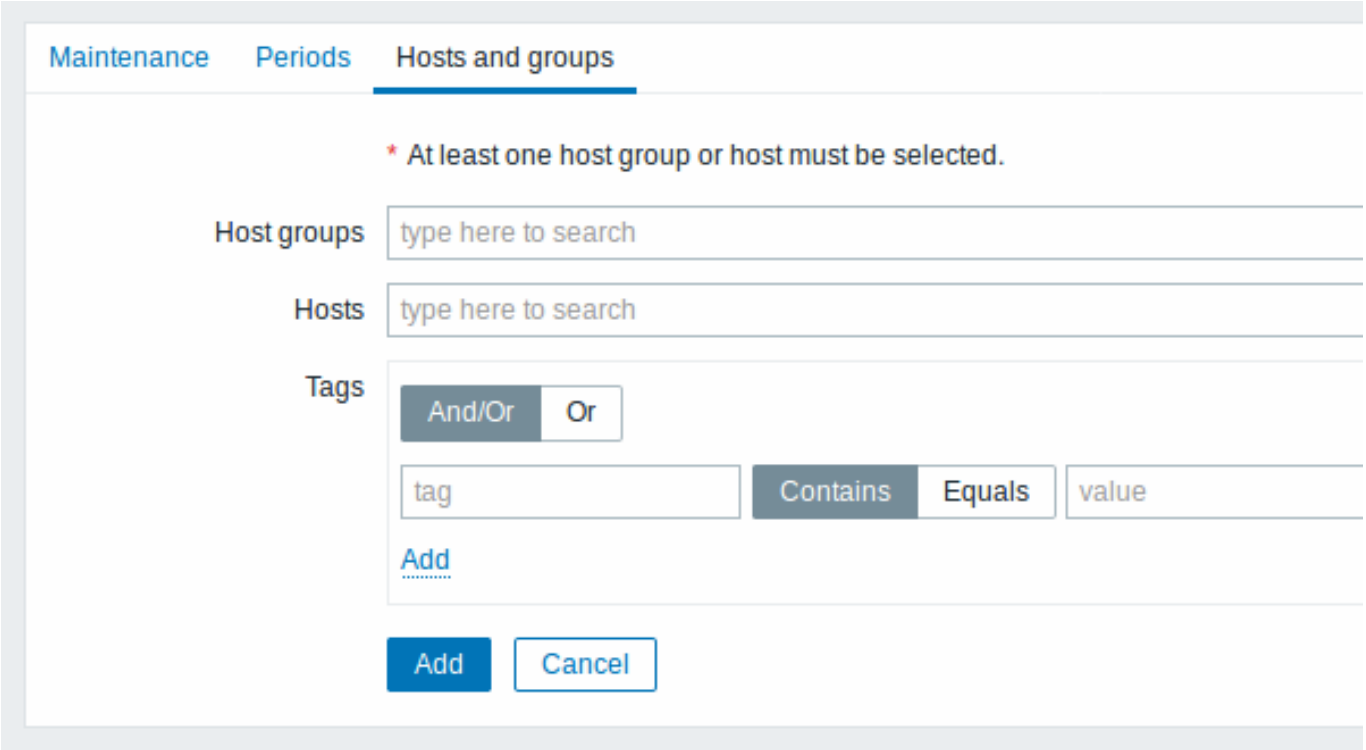


Notes:

- Daily and weekly periods have an *Every day/Every week* parameter, which defaults to '1'. Setting it to '2' would make the maintenance take place every two days or every two weeks and so on. In this case 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.

When done, click on *Add* to have the configured period listed in maintenance periods.

The **Hosts and groups** tab allows you to select the host groups, hosts and problem tags for maintenance.

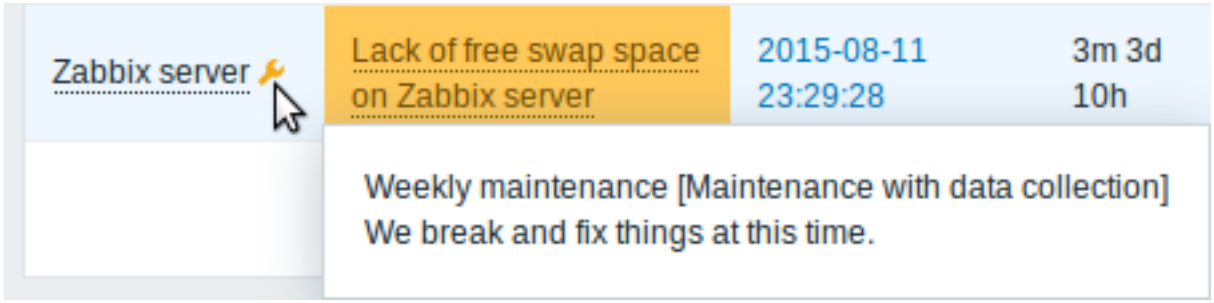


Parameter	Description
Host groups	Select host groups that the maintenance will be activated for. The maintenance will be activated for all hosts from the specified host group(s). This field is auto-complete, so starting to type in it will display a dropdown of all available host groups. Specifying a parent host group implicitly selects all nested host groups. Thus the maintenance will also be activated on hosts from nested groups.
Hosts	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.
Tags	If maintenance tags are specified, maintenance for the selected hosts will still be activated, but problems will only be suppressed (i.e. no actions will be taken) if their tags are a match. 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)

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.

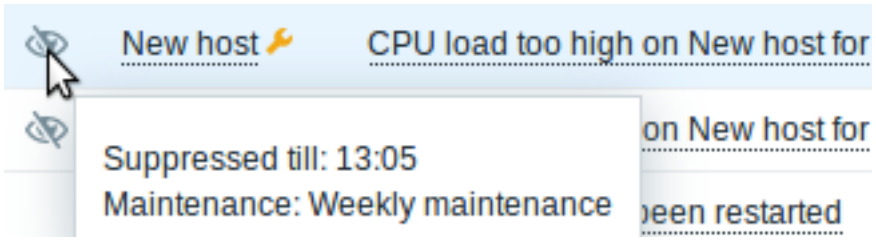
Additionally, hosts in maintenance get an orange background in *Monitoring* → *Maps*.

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* → *Overview* (in the filter; with 'Triggers' as *Type*)
- *Monitoring* → *Maps* (in map configuration)
- Global notifications (in user profile configuration)

When suppressed problems are displayed, the following icon is displayed: . Rolling a mouse over the icon displays more details:



12. Regular expressions

Overview Perl Compatible Regular Expressions (PCRE) 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.

Warning:
It's possible to run out of stack when using regular expressions. See the pcrestack man page for more information.

Note that in multiline matching, the ^ and $ anchors match at the beginning/end of each line respectively, instead of the beginning/end of the entire string.

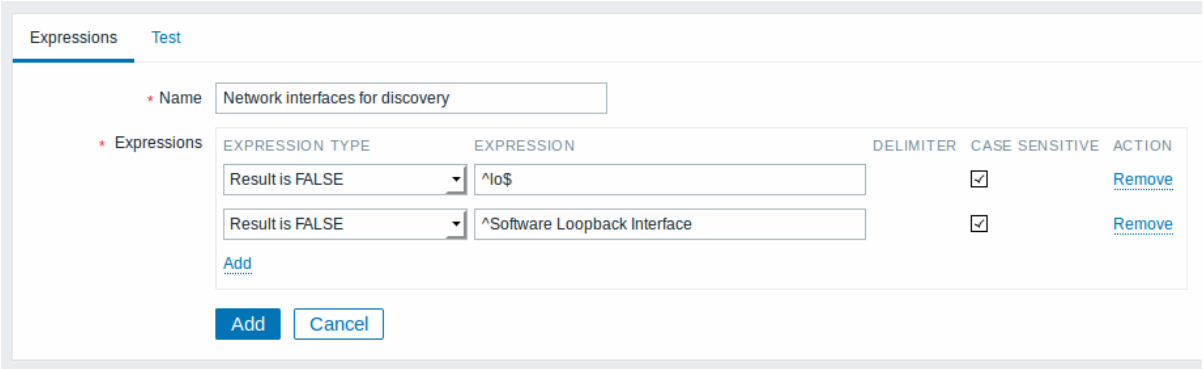
Global regular expressions There is an advanced editor for creating and testing complex regular expressions in Zabbix frontend.

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.

Parameter		Description
Name		Set the regular expression name. Any Unicode characters are allowed.
Expressions		Click on *Add* in the Expressions block to add a new subexpression.
	Expression type	Select expression type: **Character string included** - match the substring **Any character string included** - match any substring from a delimited list. The delimited list includes a comma (,), a dot (.) or a forward slash (/). **Character string not included** - match any string except the substring **Result is TRUE** - match the regular expression **Result is FALSE** - do not match the regular expression
	Expression	Enter substring/regular expression.
Delimiter		A comma (,), a dot (.) or a forward slash (/) to separate text strings in a regular expression. This parameter is active only when *"Any character string included"* expression type is selected.
Case sensitive		A checkbox to specify whether a regular expression is sensitive to capitalization of letters.

Since Zabbix 2.4.0, a forward slash (/) in the expression is treated literally, rather than a delimiter. This way it is possible to save expressions containing a slash, whereas previously it would produce an error.

Attention:
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).

Default global regular expressions

Zabbix comes with several global regular expression in its default dataset.

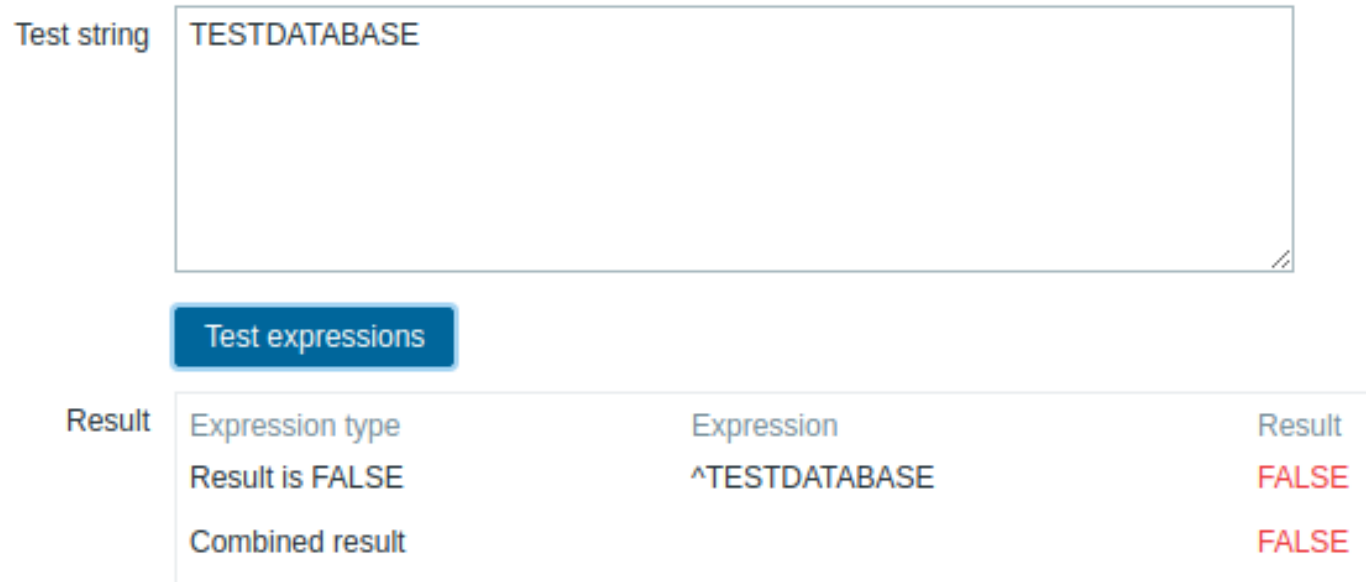
Name	Expression	Matches
File systems for discovery	`^(btrfs|ext2|ext3|ext4|jfs|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|refs|apfs|ntfs|fat32|zfs)$`	"btrfs" or "ext2" or "ext3" or "ext4" or "jfs" or "reiser" or "xfs" or "ffs" or "ufs" or "jfs" or "jfs2" or "vxfs" or "hfs" or "refs" or "apfs" or "ntfs" or "fat32" or "zfs"
Network interfaces for discovery	`^Software Loopback Interface`	Strings starting with "Software Loopback Interface".
	`^lo$`	"lo"

Name	Expression	Matches
	`^(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]o[0-9.]*$`	Strings starting with "Lo" or "lo" and optionally followed by any number of digits or dots.
	`^[Ss]ystem$`	"System" or "system"
	`^Nu[0-9.]*$`	Strings starting with "Nu" optionally followed by any number of digits or dots.
Storage devices for SNMP discovery	`^(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space)$`	"Physical memory" or "Virtual memory" or "Memory buffers" or "Cached memory" or "Swap space"
Windows service names for discovery	`^(MMCSS|gupdate|SysmonLog|clr_optimization_v2.0.50727_32|clr_optimization_v4.0.30`	"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.
Windows service startup states for discovery	`^(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$`

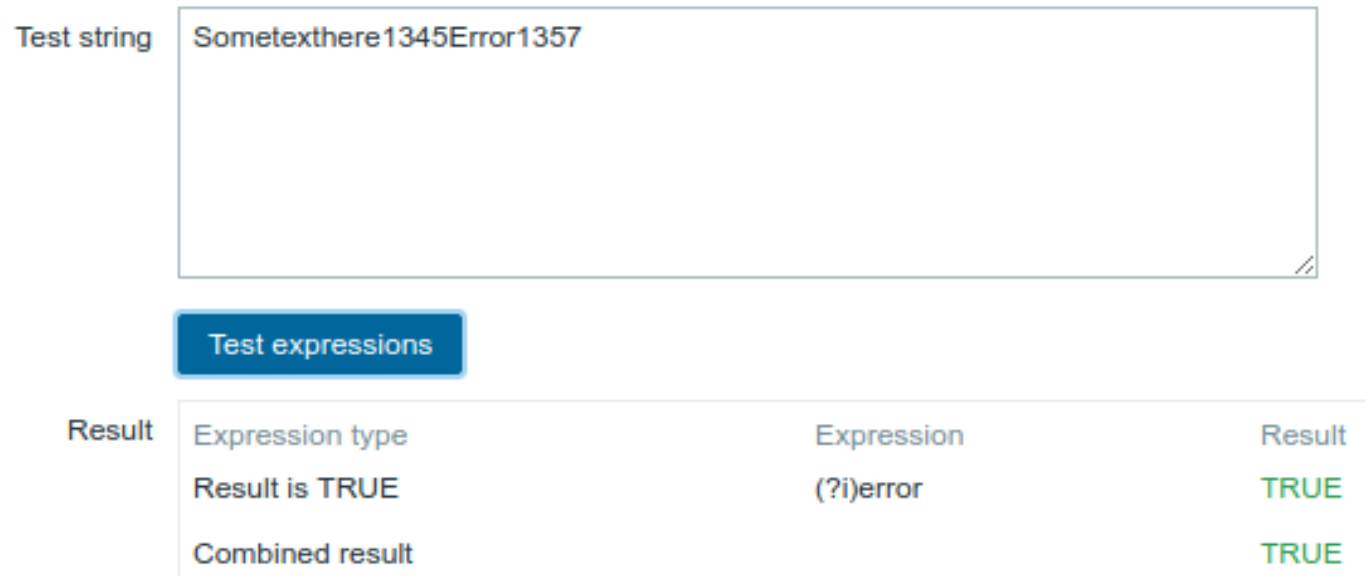


Chosen *Expression type*: "Result is FALSE". Doesn't match name, containing string "*TESTDATABASE*".

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:

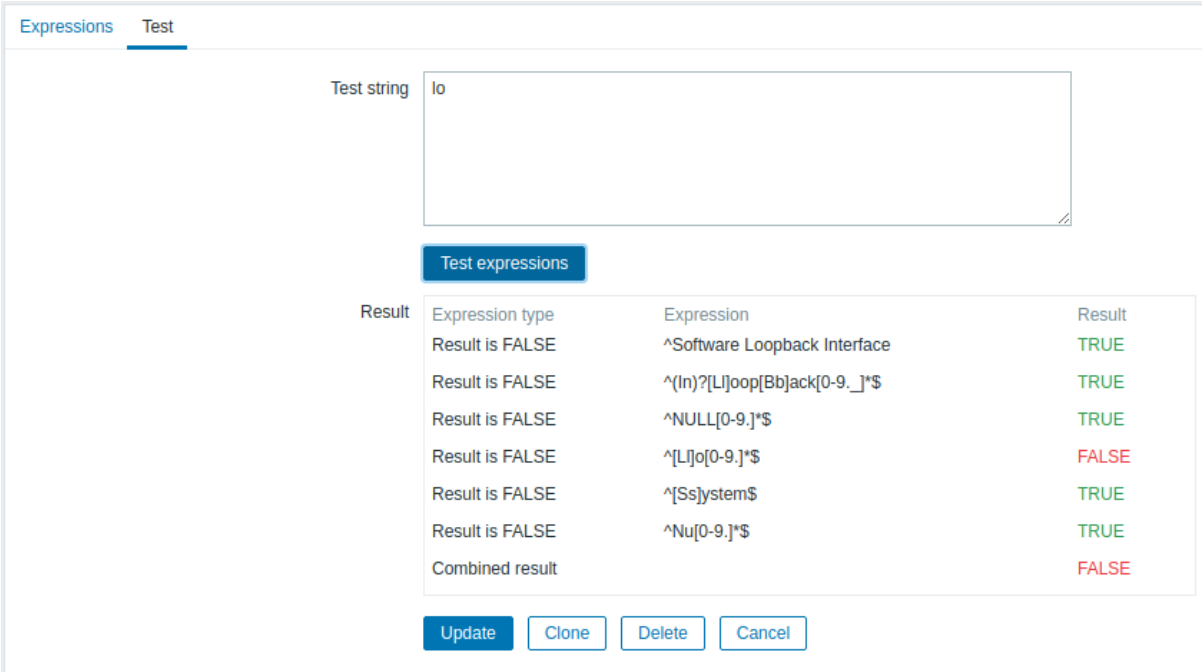
```
(?<=match (?i)everything(?-i) after this line\n)(?sx).*# we add s modifier to allow . match newline charac
```

Chosen Expression type: ”Result is TRUE”. Characters after a specific line are matched.

> **Attention:**
> **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.

More complex example

A custom regular expression may consist of multiple subexpressions, and it can be tested in the **Test** tab 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.

Regular expression support by location

Location	Regular expression	Global regular expression	Multiline matching	Comments
Agent items				
eventlog[]	Yes	Yes	Yes	`regexp`, `severity`, `source`, `eventid` parameters
log[]				`regexp` parameter
log.count[]				
logrt[]		Yes/No		`regexp` parameter supports both, `file_regexp` parameter supports non-global expressions only
logrt.count[]				
proc.cpu.util[]		No	No	`cmdline` parameter
proc.mem[]				
proc.num[]				
sensor[]				`device` and `sensor` parameters on Linux 2.4
system.hw.macaddr[]				`interface` parameter
system.sw.packages[]				`package` parameter
vfs.dir.count[]				`regex_incl`, `regex_excl`, `regex_excl_dir` parameters
vfs.dir.size[]				`regex_incl`, `regex_excl`, `regex_excl_dir` parameters
vfs.file.regexp[]			Yes	`regexp` parameter
vfs.file.regmatch[]				
web.page.regexp[]				
SNMP traps				
snmptrap[]	Yes	Yes	No	`regexp` parameter

Location	Regular expression	Global regular expression	Multiline matching	Comments
Item value preprocessing	Yes	No	No	`pattern` parameter
Trigger functions				
count()	Yes	Yes	Yes	`pattern` parameter if `operator` parameter is *regexp* or *iregexp*
logeventid()			No	`pattern` parameter
logsource()				
iregexp()			Yes	
regexp()				
Low-level discovery				
Filters	Yes	Yes	No	*Regular expression* field
Overrides	Yes	No	No	In *matches*, *does not match* options for *Operation* conditions
Action conditions	Yes	No	No	In *matches*, *does not match* options for *Host name* and *Host metadata* autoregistration conditions
Web monitoring	Yes	No	Yes	*Variables* with a **regex:** prefix *Required string* field
User macro context	Yes	No	No	In macro context with a **regex:** prefix Supported since Zabbix 5.0.2.
Macro functions				
regsub()	Yes	No	No	`pattern` parameter
iregsub()				
Icon mapping	Yes	Yes	No	*Expression* field

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 to acknowledge 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*
  - *Monitoring* → *Screens* (*Host group issues*, *Host issues*, *Problems by severity* elements)

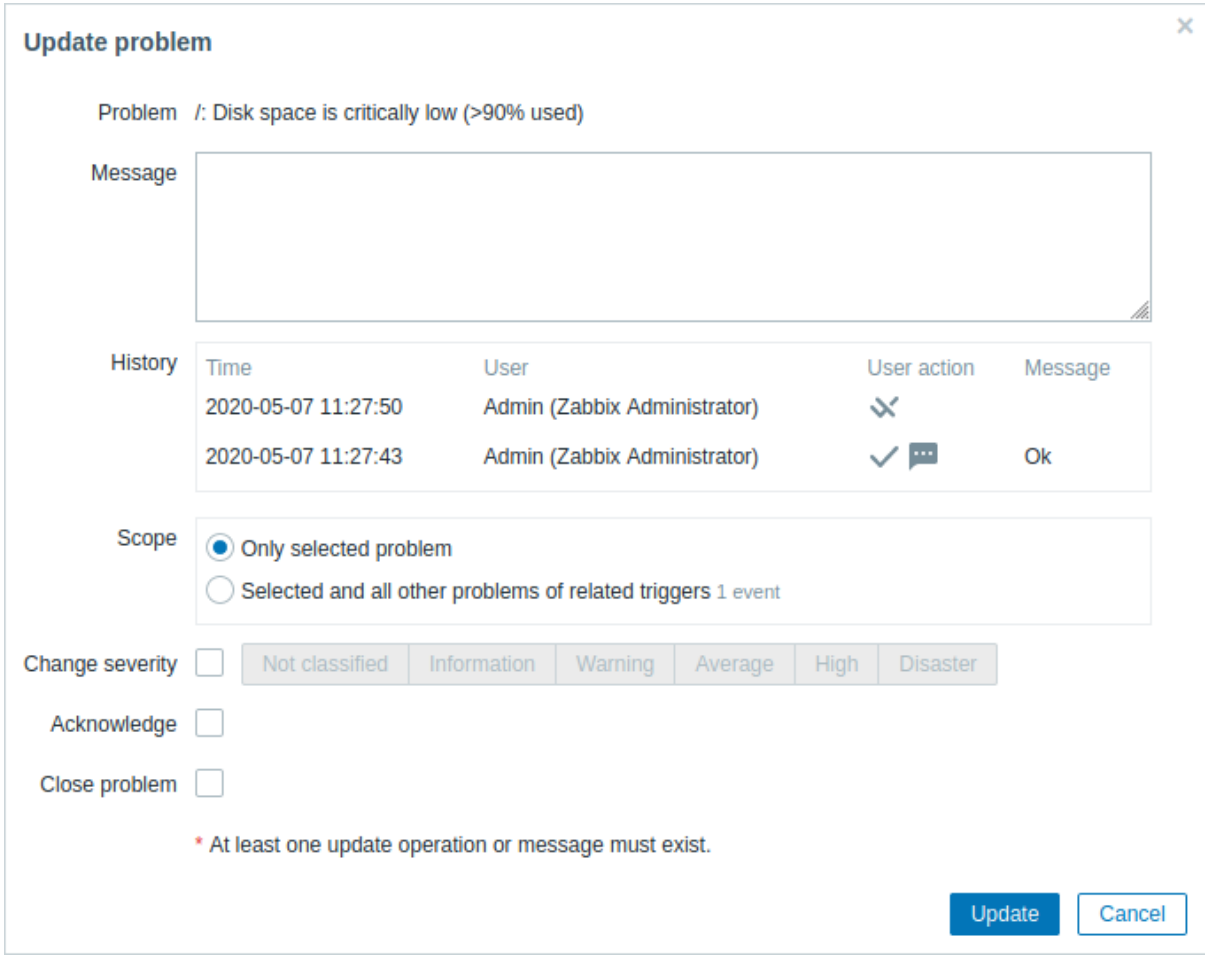
The *Ack* column contains either a 'Yes' or a 'No' link, indicating an acknowledged or an unacknowledged problem respectively. Clicking on the links will take you to the problem update popup window.

- You can click on an unresolved problem cell in:
  - *Monitoring* → *Dashboard* (*Trigger overview* widget)
  - *Monitoring* → *Overview*
  - *Monitoring* → *Screens* (*Trigger overview* element)

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
- acknowledge/unacknowledge problem
- manually close problem



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Problem	If only one problem is selected, the problem name is displayed. If several problems are selected, *N problems selected* is displayed.
Message	Enter text to comment on the problem (maximum 2048 characters).
History	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.
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.
Acknowledge	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).
Unacknowledge	Mark the checkbox to unacknowledge the problem. This checkbox is available if there is at least one acknowledged problem among the selected.
Close problem	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.

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 message or execute remote commands.

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 XML into Zabbix.
- integrate with third-party tools - the universal XML format makes 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*)
- templates
- hosts
- network maps
- screens
- media types
- images
- value maps

Export format

Data can be exported using the Zabbix web frontend or Zabbix API. Supported export formats are:

- XML - in the frontend
- XML or JSON - in Zabbix API

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 imported XML file will be deleted from the host/template after the import.
- 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 both XML and JSON, the import file must have a correct file extension: .xml for XML and .json for JSON.
- See compatibility information about supported XML versions.

XML base format The XML export format contains the following tags:

- Default header for XML documents
- Root tag for Zabbix XML export
- Export version
- Date when export was created in ISO 8601 long format

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>5.0</version>
    <date>2020-04-22T06:20:11Z</date>
</zabbix_export>
```

Other tags are dependent on exported objects.

1 Host groups

In the frontend host groups can be exported only with host or template export. When a host or template is exported all groups it belongs to are exported with it automatically.

API allows to export host groups independently from hosts or templates.

```
<groups>
    <group>
        <name>Zabbix servers</name>
    </group>
</groups>
```

groups/group

Parameter	Type	Description	Details
name	*string*	Group name.	

2 Templates

Overview

Templates are exported with many related objects and object relations.

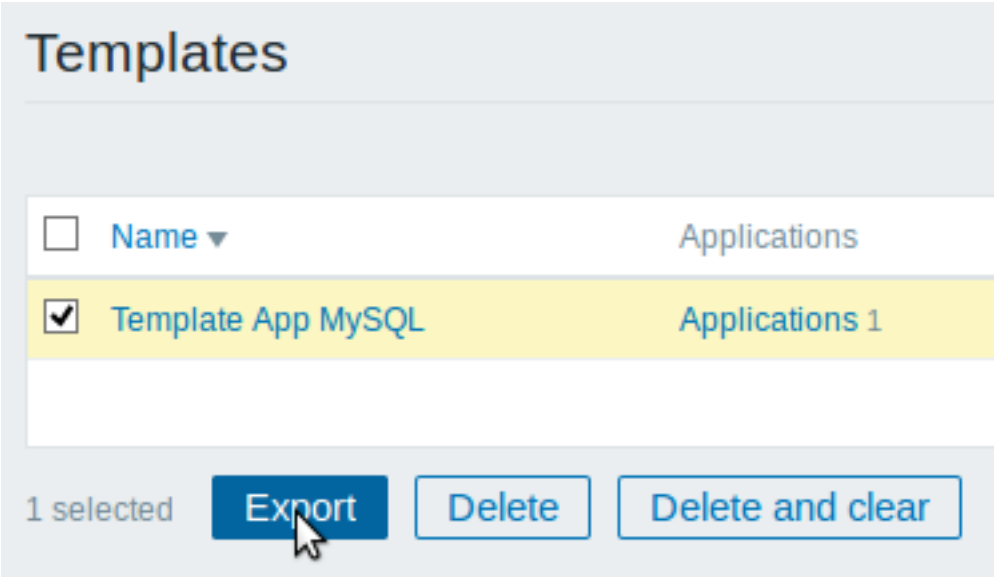
Template export contains:

- linked host groups
- template data
- linkage to other templates
- linkage to host groups
- directly linked applications
- directly linked items
- directly linked triggers
- directly linked graphs
- directly linked screens
- 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

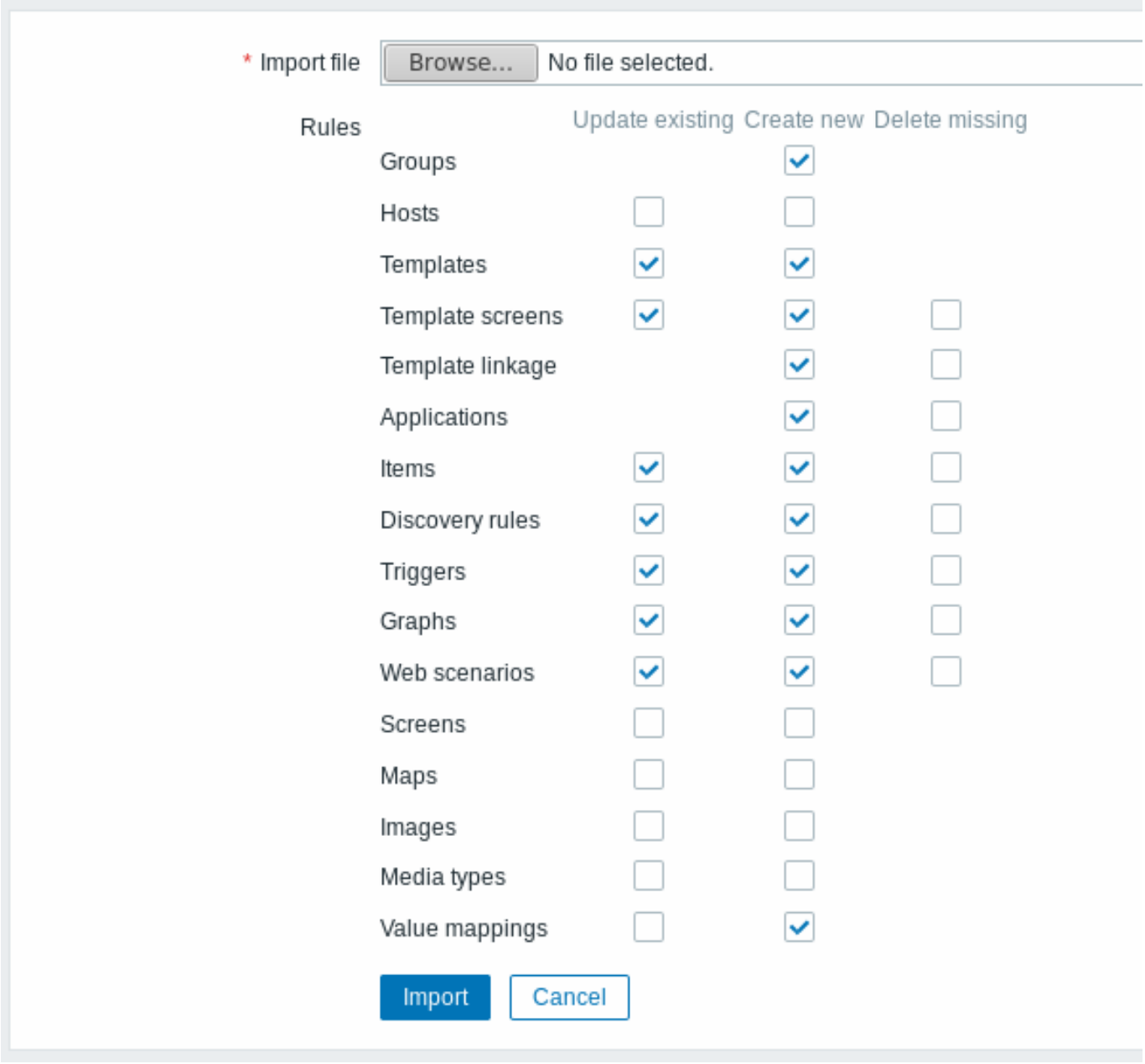


Selected templates are exported to a local XML file with default name *zabbix_export_templates.xml*.

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.

A success or failure message of the import will be displayed in the frontend.

Import rules:

Rule	Description
Update existing	Existing elements will be updated with data taken from the import file. Otherwise they will not be updated.
Create new	The import will add new elements using data from the import file. Otherwise it will not add them.

Rule	Description
Delete missing	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).

Export format

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>5.0</version>
    <date>2021-08-31T12:23:11Z</date>
    <groups>
        <group>
            <name>Templates/Modules</name>
        </group>
    </groups>
    <templates>
        <template>
            <template>Template Module Linux filesystems by Zabbix agent</template>
            <name>Template Module Linux filesystems by Zabbix agent</name>
            <description>Template tooling version used: 0.38</description>
            <groups>
                <group>
                    <name>Templates/Modules</name>
                </group>
            </groups>
            <applications>
                <application>
                    <name>Filesystems</name>
                </application>
            </applications>
            <discovery_rules>
                <discovery_rule>
                    <name>Mounted filesystem discovery</name>
                    <key>vfs.fs.discovery</key>
                    <delay>1h</delay>
                    <filter>
                        <evaltype>AND</evaltype>
                        <conditions>
                            <condition>
                                <macro>{#FSNAME}</macro>
                                <value>{$VFS.FS.FSNAME.MATCHES}</value>
                                <formulaid>A</formulaid>
                            </condition>
                            <condition>
                                <macro>{#FSNAME}</macro>
                                <value>{$VFS.FS.FSNAME.NOT_MATCHES}</value>
                                <operator>NOT_MATCHES_REGEX</operator>
                                <formulaid>B</formulaid>
                            </condition>
                            <condition>
                                <macro>{#FSTYPE}</macro>
                                <value>{$VFS.FS.FSTYPE.MATCHES}</value>
                                <formulaid>C</formulaid>
                            </condition>
                            <condition>
                                <macro>{#FSTYPE}</macro>
                                <value>{$VFS.FS.FSTYPE.NOT_MATCHES}</value>
                                <operator>NOT_MATCHES_REGEX</operator>
                                <formulaid>D</formulaid>
```

```
                            </condition>
                        </conditions>
                    </filter>
                    <description>Discovery of file systems of different types.</description>
                    <item_prototypes>
                        <item_prototype>
                            <name>{#FSNAME}: Free inodes in %</name>
                            <key>vfs.fs.inode[{#FSNAME},pfree]</key>
                            <history>7d</history>
                            <value_type>FLOAT</value_type>
                            <units>%</units>
                            <application_prototypes>
                                <application_prototype>
                                    <name>Filesystem {#FSNAME}</name>
                                </application_prototype>
                            </application_prototypes>
                            <trigger_prototypes>
                                <trigger_prototype>
                                    <expression>{min(5m)}<{$VFS.FS.INODE.PFREE.MIN.CRIT:"{#FSNAME}"}</expr
                                    <name>{#FSNAME}: Running out of free inodes (free < {$VFS.FS.INODE.PFR
                                    <opdata>Free inodes: {ITEM.LASTVALUE1}</opdata>
                                    <priority>AVERAGE</priority>
                                    <description>It may become impossible to write to disk if there are no
As symptoms, 'No space left on device' or 'Disk is full' errors may be seen even though free space is avai
                                </trigger_prototype>
                                <trigger_prototype>
                                    <expression>{min(5m)}<{$VFS.FS.INODE.PFREE.MIN.WARN:"{#FSNAME}"}</expr
                                    <name>{#FSNAME}: Running out of free inodes (free < {$VFS.FS.INODE.PFR
                                    <opdata>Free inodes: {ITEM.LASTVALUE1}</opdata>
                                    <priority>WARNING</priority>
                                    <description>It may become impossible to write to disk if there are no
As symptoms, 'No space left on device' or 'Disk is full' errors may be seen even though free space is avai
                                    <dependencies>
                                        <dependency>
                                            <name>{#FSNAME}: Running out of free inodes (free < {$VFS.FS.I
                                            <expression>{Template Module Linux filesystems by Zabbix agent
                                        </dependency>
                                    </dependencies>
                                </trigger_prototype>
                            </trigger_prototypes>
                        </item_prototype>
                        <item_prototype>
                            <name>{#FSNAME}: Space utilization</name>
                            <key>vfs.fs.size[{#FSNAME},pused]</key>
                            <history>7d</history>
                            <value_type>FLOAT</value_type>
                            <units>%</units>
                            <description>Space utilization in % for {#FSNAME}</description>
                            <application_prototypes>
                                <application_prototype>
                                    <name>Filesystem {#FSNAME}</name>
                                </application_prototype>
                            </application_prototypes>
                        </item_prototype>
                        <item_prototype>
                            <name>{#FSNAME}: Total space</name>
                            <key>vfs.fs.size[{#FSNAME},total]</key>
                            <history>7d</history>
                            <units>B</units>
                            <description>Total space in Bytes</description>
                            <application_prototypes>
                                <application_prototype>
```

```
                                <name>Filesystem {#FSNAME}</name>
                            </application_prototype>
                        </application_prototypes>
                    </item_prototype>
                    <item_prototype>
                        <name>{#FSNAME}: Used space</name>
                        <key>vfs.fs.size[{#FSNAME},used]</key>
                        <history>7d</history>
                        <units>B</units>
                        <description>Used storage in Bytes</description>
                        <application_prototypes>
                            <application_prototype>
                                <name>Filesystem {#FSNAME}</name>
                            </application_prototype>
                        </application_prototypes>
                    </item_prototype>
                </item_prototypes>
                <trigger_prototypes>
                    <trigger_prototype>
                        <expression>{Template Module Linux filesystems by Zabbix agent:vfs.fs.size[{#F
(({Template Module Linux filesystems by Zabbix agent:vfs.fs.size[{#FSNAME},total].last()}-{Template Module
                        <name>{#FSNAME}: Disk space is critically low (used > {$VFS.FS.PUSED.MAX.CRIT:
                        <opdata>Space used: {ITEM.LASTVALUE3} of {ITEM.LASTVALUE2} ({ITEM.LASTVALUE1})
                        <priority>AVERAGE</priority>
                        <description>Two conditions should match: First, space utilization should be a
 Second condition should be one of the following:
 - The disk free space is less than 5G.
 - The disk will be full in less than 24 hours.</description>
                        <manual_close>YES</manual_close>
                    </trigger_prototype>
                    <trigger_prototype>
                        <expression>{Template Module Linux filesystems by Zabbix agent:vfs.fs.size[{#F
(({Template Module Linux filesystems by Zabbix agent:vfs.fs.size[{#FSNAME},total].last()}-{Template Module
                        <name>{#FSNAME}: Disk space is low (used > {$VFS.FS.PUSED.MAX.WARN:"{#FSNAME}"
                        <opdata>Space used: {ITEM.LASTVALUE3} of {ITEM.LASTVALUE2} ({ITEM.LASTVALUE1})
                        <priority>WARNING</priority>
                        <description>Two conditions should match: First, space utilization should be a
 Second condition should be one of the following:
 - The disk free space is less than 10G.
 - The disk will be full in less than 24 hours.</description>
                        <manual_close>YES</manual_close>
                        <dependencies>
                            <dependency>
                                <name>{#FSNAME}: Disk space is critically low (used > {$VFS.FS.PUSED.M
                                <expression>{Template Module Linux filesystems by Zabbix agent:vfs.fs.
(({Template Module Linux filesystems by Zabbix agent:vfs.fs.size[{#FSNAME},total].last()}-{Template Module
                            </dependency>
                        </dependencies>
                    </trigger_prototype>
                </trigger_prototypes>
                <graph_prototypes>
                    <graph_prototype>
                        <name>{#FSNAME}: Disk space usage</name>
                        <width>600</width>
                        <height>340</height>
                        <type>PIE</type>
                        <show_3d>YES</show_3d>
                        <graph_items>
                            <graph_item>
                                <sortorder>1</sortorder>
                                <color>199C0D</color>
                                <calc_fnc>LAST</calc_fnc>
```

```
                                <type>GRAPH_SUM</type>
                                <item>
                                    <host>Template Module Linux filesystems by Zabbix agent</host>
                                    <key>vfs.fs.size[{#FSNAME},total]</key>
                                </item>
                            </graph_item>
                            <graph_item>
                                <sortorder>2</sortorder>
                                <color>F63100</color>
                                <calc_fnc>LAST</calc_fnc>
                                <item>
                                    <host>Template Module Linux filesystems by Zabbix agent</host>
                                    <key>vfs.fs.size[{#FSNAME},used]</key>
                                </item>
                            </graph_item>
                        </graph_items>
                    </graph_prototype>
                </graph_prototypes>
            </discovery_rule>
        </discovery_rules>
        <macros>
            <macro>
                <macro>{$VFS.FS.FSNAME.MATCHES}</macro>
                <value>.+</value>
                <description>This macro is used in filesystems discovery. Can be overridden on the hos
            </macro>
            <macro>
                <macro>{$VFS.FS.FSNAME.NOT_MATCHES}</macro>
                <value>^(/dev|/sys|/run|/proc|.+/shm$)</value>
                <description>This macro is used in filesystems discovery. Can be overridden on the hos
            </macro>
            <macro>
                <macro>{$VFS.FS.FSTYPE.MATCHES}</macro>
                <value>^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|ntfs|fat3
                <description>This macro is used in filesystems discovery. Can be overridden on the hos
            </macro>
            <macro>
                <macro>{$VFS.FS.FSTYPE.NOT_MATCHES}</macro>
                <value>^\s$</value>
                <description>This macro is used in filesystems discovery. Can be overridden on the hos
            </macro>
            <macro>
                <macro>{$VFS.FS.INODE.PFREE.MIN.CRIT}</macro>
                <value>10</value>
            </macro>
            <macro>
                <macro>{$VFS.FS.INODE.PFREE.MIN.WARN}</macro>
                <value>20</value>
            </macro>
            <macro>
                <macro>{$VFS.FS.PUSED.MAX.CRIT}</macro>
                <value>90</value>
            </macro>
            <macro>
                <macro>{$VFS.FS.PUSED.MAX.WARN}</macro>
                <value>80</value>
            </macro>
        </macros>
    </template>
  </templates>
</zabbix_export>
```

Element tags

Element tag values are explained in the table below.

Template tags

Element	Element property	Required	Type	Range	Description
templates		-			Root element for templates.
template		-			Individual template.
	template	x	string		Unique template name.
	name	-	string		Visible template name.
	description	-	text		Template description.
groups		x			Root element for template host groups.
group		x			Individual template host group.
	name	x	string		Host group name.
applications		-			Root element for template applications.
application		-			Individual template application.
	name	x	string		Application name.
macros		-			Root element for template user macros.
macro		-			Individual template user macro.
	macro	x	string		User macro name.
	type	-	string	0 - TEXT (default) 1 - SECRET_TEXT	Type of the macro.
	value	-	string		User macro value.
	description	-	string		User macro description.
tags		-			Root element for template tags.
tag		-			Individual template tag.
	tag	x	string		Tag name.
	value	-	string		Tag value.
templates		-			Root element for linked templates.
template		-			Individual linked template.
	name	x	string		Template name.

Template item tags

Element	Element property	Required	Type	Range[1]	Description
items		-			Root element for items.
item		-			Individual item.
	name	x	string		Item name.
	type	-	string	0 - ZABBIX_PASSIVE (default) 2 - TRAP 3 - SIMPLE 5 - INTERNAL 7 - ZABBIX_ACTIVE 8 - AGGREGATE 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	Item type.
	snmp_oid	-	string		SNMP object ID. Required by SNMP items.
	key	x	string		Item key.

Element	Element property	Required	Type	Range[1]	Description
	delay	-	`string`	Default: 1m	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. `{$FLEX_INTERVAL}/{$FLEX`
	history	-	`string`	Default: 90d	A time unit of how long the history data should be stored. Time unit with suffix, user macro or LLD macro.
	trends	-	`string`	Default: 365d	A time unit of how long the trends data should be stored. Time unit with suffix, user macro or LLD macro.
	status	-	`string`	0 - ENABLED (default) 1 - DISABLED	Item status.
	value_type	-	`string`	0 - FLOAT 1 - CHAR 2 - LOG 3 - UNSIGNED (default) 4 - TEXT	Received value type.

Element	Element property	Required	Type	Range[1]	Description
	allowed_hosts	-	`string`		List of IP addresses (comma delimited) of hosts allowed sending data for the item. Used by trapper and HTTP agent items.
	units	-	`string`		Units of returned values (bps, B, etc).
	params	-	`text`		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.
	ipmi_sensor	-	`string`		IPMI sensor. Used only by IPMI items.
	authtype	-	`string`	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	Authentication type. Used only by SSH and HTTP agent items.

Element	Element property	Required	Type	Range[1]	Description
	username	-	`string`		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.
	password	-	`string`		Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. When used by JMX agent, username should also be specified together with the password or both properties should be left blank.
	publickey	-	`string`		Name of the public key file. Required for SSH agent items.
	privatekey	-	`string`		Name of the private key file. Required for SSH agent items.
	port	-	`string`		Custom port monitored by the item. Can contain user macros. Used only by SNMP items.
	description	-	`text`		Item description.

Element	Element property	Required	Type	Range[1]	Description
	inventory_link	-	`string`	0 - NONE Capitalized host inventory field name. For example: 4 - ALIAS 6 - OS_FULL 14 - HARDWARE etc.	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.
	logtimefmt	-	`string`		Format of the time in log entries. Used only by log items.
	jmx_endpoint	-	`string`		JMX endpoint. Used only by JMX agent items.
	url	-	`string`		URL string. Required only for HTTP agent items.
	allow_traps	-	`string`	0 - NO (default) 1 - YES	Allow to populate value as in a trapper item. Used only by HTTP agent items.
	follow_redirects	-	`string`	0 - NO 1 - YES (default)	Follow HTTP response redirects while polling data. Used only by HTTP agent items.
headers		-			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.
header		-			Individual header.
	name	x	`string`		Header name.
	value	x	`string`		Header value.
	http_proxy	-	`string`		HTTP(S) proxy connection string. Used only by HTTP agent items.

Element	Element property	Required	Type	Range[1]	Description
	output_format	-	`string`	0 - RAW (default) 1 - JSON	How to process response. Used only by HTTP agent items.
	post_type	-	`string`	0 - RAW (default) 2 - JSON 3 - XML	Type of post data body. Used only by HTTP agent items.
	posts	-	`string`		HTTP(S) request body data. Used only by HTTP agent items.
query_fields		-			Root element for query parameters. Used only by HTTP agent items.
query_field		-			Individual query parameter.
	name	x	`string`		Parameter name.
	value	-	`string`		Parameter value.
	request_method	-	`string`	0 - GET (default) 1 - POST 2 - PUT 3 - HEAD	Request method. Used only by HTTP agent items.
	retrieve_mode	-	`string`	0 - BODY (default) 1 - HEADERS 2 - BOTH	What part of response should be stored. Used only by HTTP agent items.
	ssl_cert_file	-	`string`		Public SSL Key file path. Used only by HTTP agent items.
	ssl_key_file	-	`string`		Private SSL Key file path. Used only by HTTP agent items.
	ssl_key_password	-	`string`		Password for SSL Key file. Used only by HTTP agent items.

Element	Element property	Required	Type	Range[1]	Description
	status_codes	-	`string`		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.
	timeout	-	`string`		Item data polling request timeout. Supports user macros. Used only by HTTP agent items.
	verify_host	-	`string`	0 - NO (default) 1 - YES	Whether to validate that the host name for the connection matches the one in the host's certificate. Used only by HTTP agent items.
	verify_peer	-	`string`	0 - NO (default) 1 - YES	Whether to validate that the host's certificate is authentic. Used only by HTTP agent items.
value map		-			Value map.
	name	x	`string`		Name of the value map to use for the item.
applications		-			Root element for applications.
application		-			Individual application.
	name	x	`string`		Application name.
preprocessing		-			Root element for item value preprocessing.
step		-			Individual item value preprocessing step.

Element	Element property	Required	Type	Range[1]	Description
	type	x	string	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	Type of the item value preprocessing step.
	params	x	string		Parameters of the item value preprocessing step. Multiple parameters are separated by LF (\n) character.
	error_handler	-	string	0 - ORIGINAL_ERROR (default) 1 - DISCARD_VALUE 2 - CUSTOM_VALUE 3 - CUSTOM_ERROR	Action type used in case of preprocessing step failure.
	error_handler_params	-	string		Error handler parameters used with 'error_handler'.
master_item		-			Individual item master item. Required by dependent items.

Element	Element property	Required	Type	Range[1]	Description
	key	x	`string`		Dependent item master item key value. Recursion up to 3 dependent items and maximum count of dependent items equal to 29999 are allowed.
triggers		-			Root element for simple triggers.
trigger		-			Individual simple trigger.
	For trigger element tag values, see template trigger tags.				

Template low-level discovery rule tags

Element	Element property	Required	Type	Range	Description
discovery_rules		-			Root element for low-level discovery rules.
discovery_rule		-			Individual low-level discovery rule.
	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.				
	type	-	`string`	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	Item type.

Element	Element property	Required	Type	Range	Description
	lifetime	-	`string`	Default: 30d	Time period after which items that are no longer discovered will be deleted. Seconds, time unit with suffix or user macro.
filter					Individual filter.
	evaltype	-	`string`	0 - AND_OR (default) 1 - AND 2 - OR 3 - FORMULA	Logic to use for checking low-level discovery rule filter conditions.
	formula	-	`string`		Custom calculation formula for filter conditions.
conditions		-			Root element for filter conditions.
condition		-			Individual filter condition.
	macro	x	`string`		Low-level discovery macro name.
	value	-	`string`		Filter value: regular expression or global regular expression.
	operator	-	`string`	8 - MATCHES_REGEX (default) 9 - NOT_MATCHES_REGEX	Condition operator.
	formulaid	x	`character`		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.
lld_macro_paths		-			Root element for LLD macro paths.
lld_macro_path		-			Individual LLD macro path.
	lld_macro	x	`string`		Low-level discovery macro name.
	path	x	`string`		Selector for value which will be assigned to the corresponding macro.
preprocessing		-			LLD rule value preprocessing.
step		-			Individual LLD rule value preprocessing step.

Element	Element property	Required	Type	Range	Description
	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.				
	type	x	`string`	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	Type of the item value preprocessing step.
trigger_prototypes		-			Root element for trigger prototypes.
trigger_prototype		-			Individual trigger prototype.
	For trigger prototype element tag values, see regular template trigger tags.				
graph_prototypes		-			Root element for graph prototypes.
graph_prototype		-			Individual graph prototype.
	For graph prototype element tag values, see regular template graph tags.				
host_prototypes		-			Root element for host prototypes.
host_prototype		-			Individual host prototype.
	For host prototype element tag values, see regular host tags.				
item_prototypes		-			Root element for item prototypes.
item_prototype		-			Individual item prototype.
	For item prototype element tag values, see regular template item tags.				
application_prototypes		-			Root element for application prototypes.
application_prototype		-			Individual application prototype.
	name	x	`string`		Application prototype name.

Element	Element property	Required	Type	Range	Description
master_item		-			Individual item prototype master item/item prototype data.
	key	x	`string`		Dependent item prototype master item/item prototype key value. Required for a dependent item.

Template trigger tags

Element	Element property	Required	Type	Range[1]	Description
triggers		-			Root element for triggers.
trigger		-			Individual trigger.
	expression	x	`string`		Trigger expression.
	recovery_mode	-	`string`	0 - EXPRESSION (default) 1 - RECOVERY_EXPRESSION 2 - NONE	Basis for generating OK events.
	recovery_expression	-	`string`		Trigger recovery expression.
	correlation_mode	-	`string`	0 - DISABLED (default) 1 - TAG_VALUE	Correlation mode (no event correlation or event correlation by tag).
	correlation_tag	-	`string`		The tag name to be used for event correlation.
	name	x	`string`		Trigger name.
	opdata	-	`string`		Operational data.
	url	-	`string`		URL associated with the trigger.
	status	-	`string`	0 - ENABLED (default) 1 - DISABLED	Trigger status.
	priority	-	`string`	0 - NOT_CLASSIFIED (default) 1 - INFO 2 - WARNING 3 - AVERAGE 4 - HIGH 5 - DISASTER	Trigger severity.
	description	-	`text`		Trigger description.
	type	-	`string`	0 - SINGLE (default) 1 - MULTIPLE	Event generation type (single problem event or multiple problem events).

Element	Element property	Required	Type	Range[1]	Description
	manual_close	-	`string`	0 - NO (default) 1 - YES	Manual closing of problem events.
dependencies		-			Root element for dependencies.
dependency		-			Individual trigger dependency.
	name	x	`string`		Dependency trigger name.
	expression	x	`string`		Dependency trigger expression.
	recovery_expression	-	`string`		Dependency trigger recovery expression.
tags		-			Root element for event tags.
tag		-			Individual event tag.
	tag	x	`string`		Tag name.
	value	-	`string`		Tag value.

Template graph tags

Element	Element property	Required	Type	Range[1]	Description
graphs		-			Root element for graphs.
graph		-			Individual graph.
	name	x	`string`		Graph name.
	width	-	`integer`	20-65535 (default: 900)	Graph width, in pixels. Used for preview and for pie/exploded graphs.
	height	-	`integer`	20-65535 (default: 200)	Graph height, in pixels. Used for preview and for pie/exploded graphs.
	yaxismin	-	`double`	Default: 0	Value of Y axis minimum. Used if 'ymin_type_1' is FIXED.
	yaxismax	-	`double`	Default: 0	Value of Y axis maximum. Used if 'ymax_type_1' is FIXED.
	show_work_period	-	`string`	0 - NO 1 - YES (default)	Highlight non-working hours. Used by normal and stacked graphs.

Element	Element property	Required	Type	Range[1]	Description
	show_triggers	-	string	0 - NO 1 - YES (default)	Display simple trigger values as a line. Used by normal and stacked graphs.
	type	-	string	0 - NORMAL (default) 1 - STACKED 2 - PIE 3 - EXPLODED	Graph type.
	show_legend	-	string	0 - NO 1 - YES (default)	Display graph legend.
	show_3d	-	string	0 - NO (default) 1 - YES	Enable 3D style. Used by pie and exploded pie graphs.
	percent_left	-	double	Default:0	Show the percentile line for left axis. Used only for normal graphs.
	percent_right	-	double	Default:0	Show the percentile line for right axis. Used only for normal graphs.
	ymin_type_1	-	string	0 - CALCULATED (default) 1 - FIXED 2 - ITEM	Minimum value of Y axis. Used by normal and stacked graphs.
	ymax_type_1	-	string	0 - CALCULATED (default) 1 - FIXED 2 - ITEM	Maximum value of Y axis. Used by normal and stacked graphs.
ymin_item_1		-			Individual item details. Required if 'ymin_type_1' is ITEM.
	host	x	string		Item host.
	key	x	string		Item key.
ymax_item_1		-			Individual item details. Required if 'ymax_type_1' is ITEM.
	host	x	string		Item host.
	key	x	string		Item key.
graph_items		x			Root element for graph items.
graph_item		x			Individual graph item.

Element	Element property	Required	Type	Range[1]	Description
	sortorder	-	integer		Draw order. The smaller value is drawn first. Can be used to draw lines or regions behind (or in front of) another.
	drawtype	-	string	0 - SINGLE_LINE (default) 1 - FILLED_REGION 2 - BOLD_LINE 3 - DOTTED_LINE 4 - DASHED_LINE 5 - GRADIENT_LINE	Draw style of the graph item. Used only by normal graphs.
	color	-	string		Element color (6 symbols, hex).
	yaxisside	-	string	0 - LEFT (default) 1 - RIGHT	Side of the graph where the graph item's Y scale will be drawn. Used by normal and stacked graphs.
	calc_fnc	-	string	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)	Data to draw if more than one value exists for an item.
	type	-	string	0 - SIMPLE (default) 2 - GRAPH_SUM (value of the item represents the whole pie; used only by pie and exploded pie graphs)	Graph item type.
item		x			Individual item.
	host	x	string		Item host.
	key	x	string		Item key.

Template web scenario tags

Element	Element property	Required	Type	Range[1]	Description
httptests		-			Root element for web scenarios.
httptest		-			Individual web scenario.
	name	x	string		Web scenario name.
	delay	-	string	Default: 1m	Frequency of executing the web scenario. Seconds, time unit with suffix or user macro.
	attempts	-	integer	1-10 (default: 1)	The number of attempts for executing web scenario steps.

Element	Element property	Required	Type	Range[1]	Description
	agent	-	string	Default: Zabbix	Client agent. Zabbix will pretend to be the selected browser. This is useful when a website returns different content for different browsers.
	http_proxy	-	string		Specify an HTTP proxy to use, using the format: http://[username[:passw
variables		-			Root element for scenario-level variables (macros) that may be used in scenario steps.
variable		-			Individual variable.
	name	x	text		Variable name.
	value	x	text		Variable value.
headers		-			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.
header		-			Individual header.
	name	x	text		Header name.
	value	x	text		Header value.
	status	-	string	0 - ENABLED (default) 1 - DISABLED	Web scenario status.
	authentication	-	string	0 - NONE (default) 1 - BASIC 2 - NTLM	Authentication method.
	http_user	-	string		User name used for basic, HTTP or NTLM authentication.
	http_password	-	string		Password used for basic, HTTP or NTLM authentication.
	verify_peer	-	string	0 - NO (default) 1 - YES	Whether to validate that the host's certificate is authentic.

Element	Element property	Required	Type	Range[1]	Description
	verify_host	-	`string`	0 - NO (default) 1 - YES	Whether to validate that the host name for the connection matches the one in the host's certificate.
	ssl_cert_file	-	`string`		Name of the SSL certificate file used for client authentication (must be in PEM format).
	ssl_key_file	-	`string`		Name of the SSL private key file used for client authentication (must be in PEM format).
	ssl_key_password	-	`string`		SSL private key file password.
steps		x			Root element for web scenario steps.
step		x			Individual web scenario step.
	name	x	`string`		Web scenario step name.
	url	x	`string`		URL for monitoring.
query_fields		-			Root element for query fields - an array of HTTP fields that will be added to the URL when performing a request.
query_field		-			Individual query field.
	name	x	`string`		Query field name.
	value	-	`string`		Query field value.
posts		-			HTTP POST variables as a string (raw post data) or as an array of HTTP fields (form field data).
post_field		-			Individual post field.
	name	x	`string`		Post field name.
	value	x	`string`		Post field value.

Element	Element property	Required	Type	Range[1]	Description
variables		-			Root element of step-level variables (macros) that should be applied after this step. If the variable value has a 'regex:' prefix, then its value is extracted from the data returned by this step according to the regular expression pattern following the 'regex:' prefix
variable		-			Individual variable.
	name	x	`string`		Variable name.
	value	x	`string`		Variable value.
headers		-			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.
header		-			Individual header.
	name	x	`string`		Header name.
	value	x	`string`		Header value.
	follow_redirects	-	`string`	0 - NO 1 - YES (default)	Follow HTTP redirects.
	retrieve_mode	-	`string`	0 - BODY (default) 1 - HEADERS 2 - BOTH	HTTP response retrieve mode.
	timeout	-	`string`	Default: 15s	Timeout of step execution. Seconds, time unit with suffix or user macro.
	required	-	`string`		Text that must be present in the response. Ignored if empty.

Element	Element property	Required	Type	Range[1]	Description
	status_codes	-	`string`		A comma delimited list of accepted HTTP status codes. Ignored if empty. For example: 200-201,210-299

Template screen tags

Element	Element property	Required	Type	Range[1]	Description
screens		-			Root element for template screens.
screen		-			Individual template screen.
screen_items		-			Root element for template screen items.
screen_item		-			Individual template screen item.

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.

3 Hosts

Overview

Hosts are exported with many related objects and object relations.

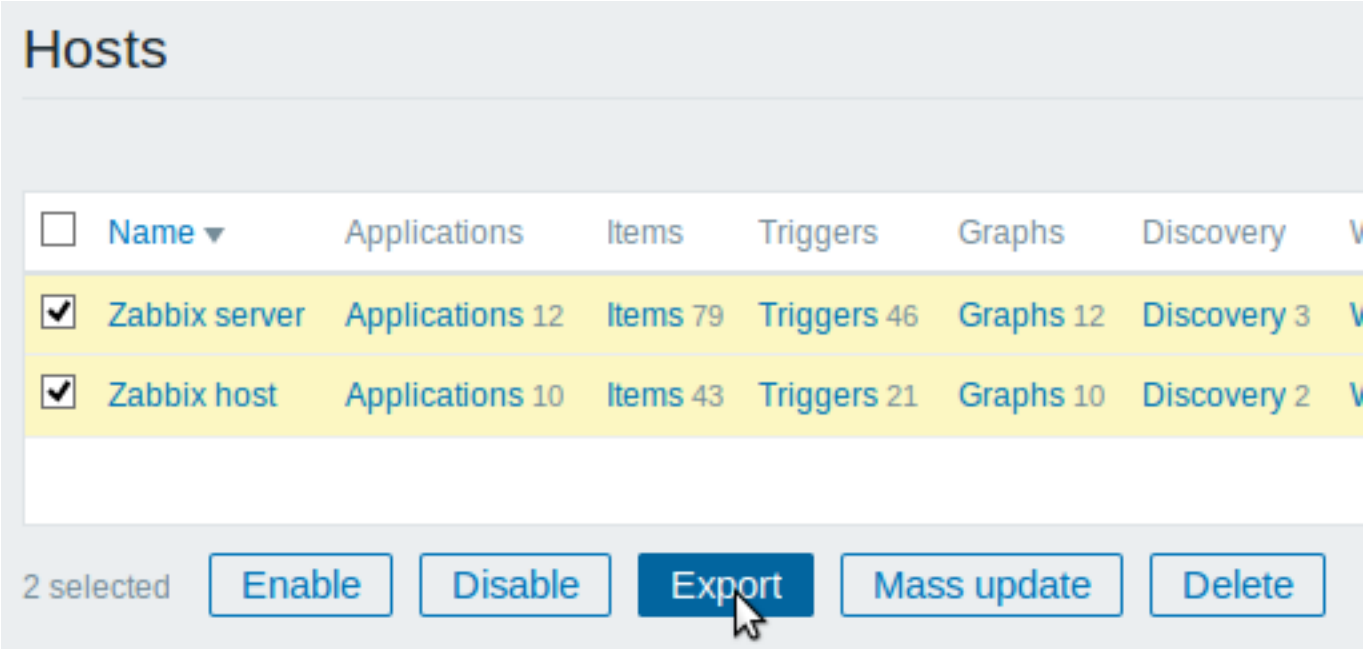
Host export contains:

- linked host groups
- host data
- template linkage
- host group linkage
- host interfaces
- directly linked applications
- 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

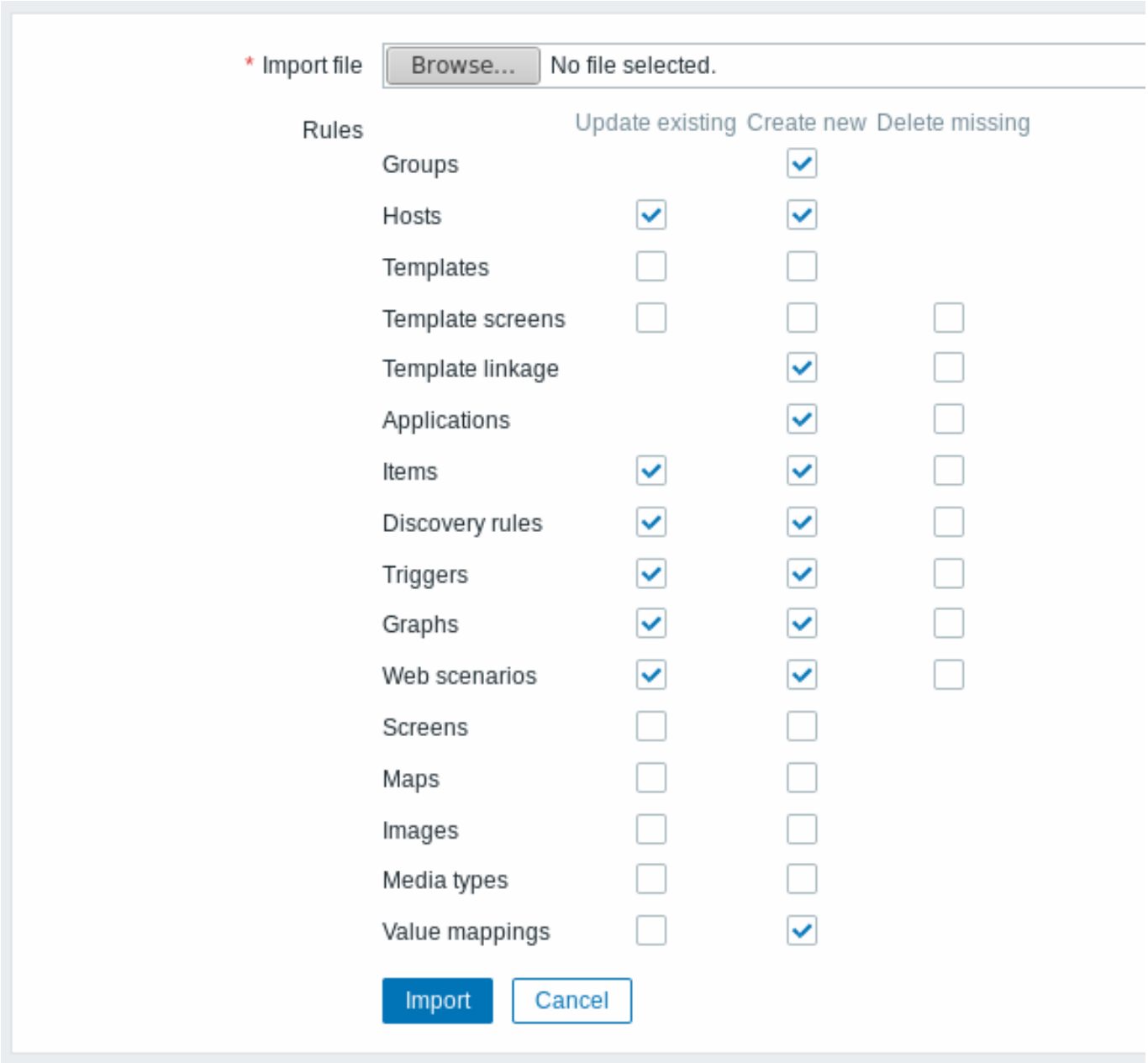


Selected hosts are exported to a local XML file with default name *zbx_export_hosts.xml*.

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:

Rule	Description
Update existing	Existing elements will be updated with data taken from the import file. Otherwise they will not be updated.
Create new	The import will add new elements using data from the import file. Otherwise it will not add them.
Delete missing	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 host along with all entities inherited from the potentially unlinked templates (items, triggers, etc).

Export format

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>5.0</version>
    <date>2020-04-22T07:47:33Z</date>
    <groups>
```

```
        <group>
            <name>Discovered hosts</name>
        </group>
        <group>
            <name>Zabbix servers</name>
        </group>
    </groups>
    <hosts>
        <host>
            <host>Zabbix server 1</host>
            <name>Main Zabbix server</name>
            <tls_connect>TLS_PSK</tls_connect>
            <tls_accept>
                <option>NO_ENCRYPTION</option>
                <option>TLS_PSK</option>
            </tls_accept>
            <tls_psk_identity>z112</tls_psk_identity>
            <tls_psk>1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952</tls_psk>
            <templates>
                <template>
                    <name>Template App Zabbix Server</name>
                </template>
                <template>
                    <name>Template OS Linux</name>
                </template>
            </templates>
            <groups>
                <group>
                    <name>Discovered hosts</name>
                </group>
                <group>
                    <name>Zabbix servers</name>
                </group>
            </groups>
            <interfaces>
                <interface>
                    <ip>192.168.1.1</ip>
                    <interface_ref>if1</interface_ref>
                </interface>
            </interfaces>
            <items>
                <item>
                    <name>Zabbix trap</name>
                    <type>TRAP</type>
                    <key>trap</key>
                    <delay>0</delay>
                    <history>1w</history>
                    <applications>
                        <application>
                            <name>Zabbix server</name>
                        </application>
                    </applications>
                    <preprocessing>
                        <step>
                            <type>MULTIPLIER</type>
                            <params>8</params>
                        </step>
                    </preprocessing>
                    <triggers>
                        <trigger>
                            <expression>{last()}=0</expression>
                            <name>Last value is zero</name>
                            <priority>WARNING</priority>
```

```
                        <tags>
                            <tag>
                                <tag>Process</tag>
                                <value>Internal test</value>
                            </tag>
                        </tags>
                    </trigger>
                </triggers>
            </item>
        </items>
        <tags>
            <tag>
                <tag>Process</tag>
                <value>Zabbix</value>
            </tag>
        </tags>
        <macros>
            <macro>
                <macro>{$HOST.MACRO}</macro>
                <value>123</value>
            </macro>
            <macro>
                <macro>{$PASSWORD1}</macro>
                <type>SECRET_TEXT</type>
            </macro>
        </macros>
        <inventory>
            <type>Zabbix server</type>
            <name>yyyyyy-HP-Pro-3010-Small-Form-Factor-PC</name>
            <os>Linux yyyyyy-HP-Pro-3010-Small-Form-Factor-PC 4.4.0-165-generic #193-Ubuntu SMP Tue Se
        </inventory>
        <inventory_mode>AUTOMATIC</inventory_mode>
    </host>
</hosts>
<graphs>
    <graph>
        <name>CPU utilization server</name>
        <show_work_period>NO</show_work_period>
        <show_triggers>NO</show_triggers>
        <graph_items>
            <graph_item>
                <drawtype>FILLED_REGION</drawtype>
                <color>FF5555</color>
                <item>
                    <host>Zabbix server 1</host>
                    <key>system.cpu.util[,steal]</key>
                </item>
            </graph_item>
            <graph_item>
                <sortorder>1</sortorder>
                <drawtype>FILLED_REGION</drawtype>
                <color>55FF55</color>
                <item>
                    <host>Zabbix server 1</host>
                    <key>system.cpu.util[,softirq]</key>
                </item>
            </graph_item>
            <graph_item>
                <sortorder>2</sortorder>
                <drawtype>FILLED_REGION</drawtype>
                <color>009999</color>
                <item>
                    <host>Zabbix server 1</host>
```

```
                        <key>system.cpu.util[,interrupt]</key>
                    </item>
                </graph_item>
                <graph_item>
                    <sortorder>3</sortorder>
                    <drawtype>FILLED_REGION</drawtype>
                    <color>990099</color>
                    <item>
                        <host>Zabbix server 1</host>
                        <key>system.cpu.util[,nice]</key>
                    </item>
                </graph_item>
                <graph_item>
                    <sortorder>4</sortorder>
                    <drawtype>FILLED_REGION</drawtype>
                    <color>999900</color>
                    <item>
                        <host>Zabbix server 1</host>
                        <key>system.cpu.util[,iowait]</key>
                    </item>
                </graph_item>
                <graph_item>
                    <sortorder>5</sortorder>
                    <drawtype>FILLED_REGION</drawtype>
                    <color>990000</color>
                    <item>
                        <host>Zabbix server 1</host>
                        <key>system.cpu.util[,system]</key>
                    </item>
                </graph_item>
                <graph_item>
                    <sortorder>6</sortorder>
                    <drawtype>FILLED_REGION</drawtype>
                    <color>000099</color>
                    <calc_fnc>MIN</calc_fnc>
                    <item>
                        <host>Zabbix server 1</host>
                        <key>system.cpu.util[,user]</key>
                    </item>
                </graph_item>
                <graph_item>
                    <sortorder>7</sortorder>
                    <drawtype>FILLED_REGION</drawtype>
                    <color>009900</color>
                    <item>
                        <host>Zabbix server 1</host>
                        <key>system.cpu.util[,idle]</key>
                    </item>
                </graph_item>
            </graph_items>
        </graph>
    </graphs>
</zabbix_export>
```

Element tags

Element tag values are explained in the table below.

Host tags

Element	Element property	Required	Type	Range[1]	Description
groups		x			Root element for host groups.

Element	Element property	Required	Type	Range[1]	Description
group		x			Individual host group.
	name	x	`string`		Host group name.
hosts		-			Root element for hosts.
host		-			Individual host.
	host	x	`string`		Unique host name.
	name	-	`string`		Visible host name.
	description	-	`text`		Host description.
	status	-	`string`	0 - ENABLED (default) 1 - DISABLED	Host status.
	ipmi_authtype	-	`string`	-1 - DEFAULT (default) 0 - NONE 1 - MD2 2 - MD5 4 - STRAIGHT 5 - OEM 6 - RMCP_PLUS	IPMI session authentication type.
	ipmi_privilege	-	`string`	1 - CALLBACK 2 - USER (default) 3 - OPERATOR 4 - ADMIN 5 - OEM	IPMI session privilege level.
	ipmi_username	-	`string`		Username for IPMI checks.
	ipmi_password	-	`string`		Password for IPMI checks.
	tls_connect	-	`string`	1 - NO_ENCRYPTION (default) 2 - TLS_PSK 4 - TLS_CERTIFICATE	Type of outgoing connection.
	inventory_mode	-	`string`	-1 - DISABLED 0 - MANUAL (default) 1 - AUTOMATIC	Inventory mode.
tls_accept		-			Root element for incoming connection options.
	option	-	`string`	1 - NO_ENCRYPTION (default) 2 - TLS_PSK 4 - TLS_CERTIFICATE	Type of incoming connection. If both unencrypted and encrypted connection is allowed, the <option> property is used twice, one time with NO_ENCRYPTION and another with the encryption option (see example above).
	tls_issuer	-	`string`		Allowed agent/proxy certificate issuer.

Element	Element property	Required	Type	Range[1]	Description
	tls_subject	-	string		Allowed agent/proxy certificate subject.
	tls_psk_identity	-	string		PSK identity string. Required if either tls_connect or tls_accept has PSK enabled.
	tls_psk	-	string		The preshared key string, at least 32 hex digits. Required if either tls_connect or tls_accept has PSK enabled.
proxy		-			Proxy.
	name	x	string		Name of the proxy (if any) that monitors the host.
templates		-			Root element for linked templates.
template		-			Individual template.
	name	x	string		Template name.
interfaces		-			Root element for host interfaces.
interface		-			Individual interface.
	default	-	string	0 - NO 1 - YES (default)	Whether this is the primary host interface. There can be only one primary interface of one type on a host.
	type	-	string	1 - ZABBIX (default) 2 - SNMP 3 - IPMI 4 - JMX	Interface type.
	useip	-	string	0 - NO 1 - YES (default)	Whether to use IP as the interface for connecting to the host (if not, DNS will be used).

Element	Element property	Required	Type	Range[1]	Description
	ip	-	`string`		IP address, can be either IPv4 or IPv6. Required if the connection is made via IP.
	dns	-	`string`		DNS name. Required if the connection is made via DNS.
	port	-	`string`		Port number. Supports user macros.
	interface_ref	x	`string`	Format: `if<N>`	Interface reference name to be used in items.
details		-			Root element for interface details[2].
	version	-	`string`	1 - SNMPV1 2 - SNMP_V2C (default) 3 - SNMP_V3	Use this SNMP version.
	community	-	`string`		SNMP community. Required by SNMPv1 and SNMPv2 items.
	contextname	-	`string`		SNMPv3 context name. Used only by SNMPv3 items.
	securityname	-	`string`		SNMPv3 security name. Used only by SNMPv3 items.
	securitylevel	-	`string`	0 - NOAUTHNOPRIV (default) 1 - AUTHNOPRIV 2 - AUTHPRIV	SNMPv3 security level. Used only by SNMPv3 items.
	authprotocol	-	`string`	0 - MD5 (default) 1 - SHA	SNMPv3 authentication protocol. Used only by SNMPv3 items.
	authpassphrase	-	`string`		SNMPv3 authentication passphrase. Used only by SNMPv3 items.

Element	Element property	Required	Type	Range[1]	Description
	privprotocol	-	`string`	0 - DES (default) 1 - AES	SNMPv3 privacy protocol. Used only by SNMPv3 items.
	privpassphrase	-	`string`		SNMPv3 privacy passphrase. Used only by SNMPv3 items.
	bulk	-	`string`	0 - NO 1 - YES (default)	Use bulk requests for SNMP.
items		-			Root element for items.
item		-			Individual item.
	For item element tag values, see host item tags.				
tags		-			Root element for host tags.
tag		-			Individual host tag.
	tag	x	`string`		Tag name.
	value	-	`string`		Tag value.
macros		-			Root element for macros.
macro		-			Individual macro.
	macro	x			User macro name.
	type	-	`string`	0 - TEXT (default) 1 - SECRET_TEXT	Type of the macro.
	value	-	`string`		User macro value.
	description	-	`string`		User macro description.
inventory		-			Root element for host inventory.
	<inventory_property>	-			Individual inventory property. All available inventory properties are listed under the respective tags, e.g. <type>, <name>, <os> (see example above).

Host item tags

Element	Element property	Required	Type	Range[1]	Description
items		-			Root element for items.
item		-			Individual item.
	name	x	`string`		Item name.

Element	Element property	Required	Type	Range[1]	Description
	type	-	`string`	0 - ZABBIX_PASSIVE (default) 2 - TRAP 3 - SIMPLE 5 - INTERNAL 7 - ZABBIX_ACTIVE 8 - AGGREGATE 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	Item type.
	snmp_oid	-	`string`		SNMP object ID. Required by SNMP items.
	key	x	`string`		Item key.
	delay	-	`string`	Default: 1m	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`

Element	Element property	Required	Type	Range[1]	Description
	history	-	`string`	Default: 90d	A time unit of how long the history data should be stored. Time unit with suffix, user macro or LLD macro.
	trends	-	`string`	Default: 365d	A time unit of how long the trends data should be stored. Time unit with suffix, user macro or LLD macro.
	status	-	`string`	0 - ENABLED (default) 1 - DISABLED	Item status.
	value_type	-	`string`	0 - FLOAT 1 - CHAR 2 - LOG 3 - UNSIGNED (default) 4 - TEXT	Received value type.
	allowed_hosts	-	`string`		List of IP addresses (comma delimited) of hosts allowed sending data for the item. Used by trapper and HTTP agent items.
	units	-	`string`		Units of returned values (bps, B, etc).
	params	-	`text`		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.
	ipmi_sensor	-	`string`		IPMI sensor. Used only by IPMI items.

Element	Element property	Required	Type	Range[1]	Description
	authtype	-	`string`	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	Authentication type. Used only by SSH and HTTP agent items.
	username	-	`string`		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.
	password	-	`string`		Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. When used by JMX agent, username should also be specified together with the password or both properties should be left blank.
	publickey	-	`string`		Name of the public key file. Required for SSH agent items.
	privatekey	-	`string`		Name of the private key file. Required for SSH agent items.
	description	-	`text`		Item description.

Element	Element property	Required	Type	Range[1]	Description
	inventory_link	-	`string`	0 - NONE Capitalized host inventory field name. For example: 4 - ALIAS 6 - OS_FULL 14 - HARDWARE etc.	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.
	logtimefmt	-	`string`		Format of the time in log entries. Used only by log items.
	interface_ref	-	`string`	Format: `if<N>`	Reference to the host interface.
	jmx_endpoint	-	`string`		JMX endpoint. Used only by JMX agent items.
	url	-	`string`		URL string. Required only for HTTP agent items.
	allow_traps	-	`string`	0 - NO (default) 1 - YES	Allow to populate value as in a trapper item. Used only by HTTP agent items.
	follow_redirects	-	`string`	0 - NO 1 - YES (default)	Follow HTTP response redirects while polling data. Used only by HTTP agent items.
headers		-			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.
header		-			Individual header.
	name	x	`string`		Header name.
	value	x	`string`		Header value.

Element	Element property	Required	Type	Range[1]	Description
	http_proxy	-	`string`		HTTP(S) proxy connection string. Used only by HTTP agent items.
	output_format	-	`string`	0 - RAW (default) 1 - JSON	How to process response. Used only by HTTP agent items.
	post_type	-	`string`	0 - RAW (default) 2 - JSON 3 - XML	Type of post data body. Used only by HTTP agent items.
	posts	-	`string`		HTTP(S) request body data. Used only by HTTP agent items.
query_fields		-			Root element for query parameters. Used only by HTTP agent items.
query_field		-			Individual query parameter.
	name	x	`string`		Parameter name.
	value	-	`string`		Parameter value.
	request_method	-	`string`	0 - GET (default) 1 - POST 2 - PUT 3 - HEAD	Request method. Used only by HTTP agent items.
	retrieve_mode	-	`string`	0 - BODY (default) 1 - HEADERS 2 - BOTH	What part of response should be stored. Used only by HTTP agent items.
	ssl_cert_file	-	`string`		Public SSL Key file path. Used only by HTTP agent items.

Element	Element property	Required	Type	Range[1]	Description
	ssl_key_file	-	`string`		Private SSL Key file path. Used only by HTTP agent items.
	ssl_key_password	-	`string`		Password for SSL Key file. Used only by HTTP agent items.
	status_codes	-	`string`		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.
	timeout	-	`string`		Item data polling request timeout. Supports user macros. Used only by HTTP agent items.
	verify_host	-	`string`	0 - NO (default) 1 - YES	Whether to validate that the host name for the connection matches the one in the host's certificate. Used only by HTTP agent items.
	verify_peer	-	`string`	0 - NO (default) 1 - YES	Whether to validate that the host's certificate is authentic. Used only by HTTP agent items.
value map		-			Value map.
	name	x	`string`		Name of the value map to use for the item.
applications		-			Root element for applications.

Element	Element property	Required	Type	Range[1]	Description
application		-			Individual application.
	name	x	`string`		Application name.
preprocessing		-			Root element for item value preprocessing.
step		-			Individual item value preprocessing step.
	type	x	`string`	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	Type of the item value preprocessing step.
	params	x	`string`		Parameters of the item value preprocessing step. Multiple parameters are separated by LF (\n) character.
	error_handler	-	`string`	0 - ORIGINAL_ERROR (default) 1 - DISCARD_VALUE 2 - CUSTOM_VALUE 3 - CUSTOM_ERROR	Action type used in case of preprocessing step failure.
	error_handler_params	-	`string`		Error handler parameters used with 'error_handler'.

Element	Element property	Required	Type	Range[1]	Description
master_item		-			Individual item master item. Required by dependent items.
	key	x	`string`		Dependent item master item key value. Recursion up to 3 dependent items and maximum count of dependent items equal to 29999 are allowed.
triggers		-			Root element for simple triggers.
trigger		-			Individual simple trigger.
	For trigger element tag values, see host trigger tags.				

Host low-level discovery rule tags

Element	Element property	Required	Type	Range[1]	Description
discovery_rules		-			Root element for low-level discovery rules.
discovery_rule		-			Individual low-level discovery rule.
	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.				
	type	-	`string`	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	Item type.

Element	Element property	Required	Type	Range[1]	Description
	lifetime	-	string	Default: 30d	Time period after which items that are no longer discovered will be deleted. Seconds, time unit with suffix or user macro.
filter					Individual filter.
	evaltype	-	string	0 - AND_OR (default) 1 - AND 2 - OR 3 - FORMULA	Logic to use for checking low-level discovery rule filter conditions.
	formula	-	string		Custom calculation formula for filter conditions.
conditions		-			Root element for filter conditions.
condition		-			Individual filter condition.
	macro	x	string		Low-level discovery macro name.
	value	-	string		Filter value: regular expression or global regular expression.
	operator	-	string	8 - MATCHES_REGEX (default) 9 - NOT_MATCHES_REGEX	Condition operator.
	formulaid	x	character		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.
lld_macro_paths		-			Root element for LLD macro paths.
lld_macro_path		-			Individual LLD macro path.
	lld_macro	x	string		Low-level discovery macro name.

Element	Element property	Required	Type	Range[1]	Description
	path	x	`string`		Selector for value which will be assigned to the corresponding macro.
preprocessing		-			LLD rule value preprocessing.
step		-			Individual LLD rule value preprocessing step.
	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	x	`string`	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	Type of the item value preprocessing step.
trigger_prototypes		-			Root element for trigger prototypes.
trigger_prototype		-			Individual trigger prototype.
	For trigger prototype element tag values, see regular host trigger tags.				
graph_prototypes		-			Root element for graph prototypes.
graph_prototype		-			Individual graph prototype.
	For graph prototype element tag values, see regular host graph tags.				
host_prototypes		-			Root element for host prototypes.
host_prototype		-			Individual host prototype.
	For host prototype element tag values, see regular host tags.				
item_prototypes		-			Root element for item prototypes.
item_prototype		-			Individual item prototype.

Element	Element property	Required	Type	Range[1]	Description
	For item prototype element tag values, see regular host item tags.				
application_prototypes		-			Root element for application prototypes.
application_prototype		-			Individual application prototype.
	name	x	`string`		Application prototype name.
master_item		-			Individual item prototype master item/item prototype data.
	key	x	`string`		Dependent item prototype master item/item prototype key value. Required for a dependent item.

Host trigger tags

Element	Element property	Required	Type	Range[1]	Description
triggers		-			Root element for triggers.
trigger		-			Individual trigger.
	expression	x	`string`		Trigger expression.
	recovery_mode	-	`string`	0 - EXPRESSION (default) 1 - RECOVERY_EXPRESSION 2 - NONE	Basis for generating OK events.
	recovery_expression	-	`string`		Trigger recovery expression.
	correlation_mode	-	`string`	0 - DISABLED (default) 1 - TAG_VALUE	Correlation mode (no event correlation or event correlation by tag).
	correlation_tag	-	`string`		The tag name to be used for event correlation.
	name	x	`string`		Trigger name.
	opdata	-	`string`		Operational data.
	url	-	`string`		URL associated with the trigger.
	status	-	`string`	0 - ENABLED (default) 1 - DISABLED	Trigger status.

Element	Element property	Required	Type	Range[1]	Description
	priority	-	`string`	0 - NOT_CLASSIFIED (default) 1 - INFO 2 - WARNING 3 - AVERAGE 4 - HIGH 5 - DISASTER	Trigger severity.
	description	-	`text`		Trigger description.
	type	-	`string`	0 - SINGLE (default) 1 - MULTIPLE	Event generation type (single problem event or multiple problem events).
	manual_close	-	`string`	0 - NO (default) 1 - YES	Manual closing of problem events.
dependencies		-			Root element for dependencies.
dependency		-			Individual trigger dependency.
	name	x	`string`		Dependency trigger name.
	expression	x	`string`		Dependency trigger expression.
	recovery_expression	-	`string`		Dependency trigger recovery expression.
tags		-			Root element for event tags.
tag		-			Individual event tag.
	tag	x	`string`		Tag name.
	value	-	`string`		Tag value.

Host graph tags

Element	Element property	Required	Type	Range[1]	Description
graphs		-			Root element for graphs.
graph		-			Individual graph.
	name	x	`string`		Graph name.
	width	-	`integer`	20-65535 (default: 900)	Graph width, in pixels. Used for preview and for pie/exploded graphs.
	height	-	`integer`	20-65535 (default: 200)	Graph height, in pixels. Used for preview and for pie/exploded graphs.

Element	Element property	Required	Type	Range[1]	Description
	yaxismin	-	`double`	Default: 0	Value of Y axis minimum. Used if 'ymin_type_1' is FIXED.
	yaxismax	-	`double`	Default: 0	Value of Y axis maximum. Used if 'ymax_type_1' is FIXED.
	show_work_period	-	`string`	0 - NO 1 - YES (default)	Highlight non-working hours. Used by normal and stacked graphs.
	show_triggers	-	`string`	0 - NO 1 - YES (default)	Display simple trigger values as a line. Used by normal and stacked graphs.
	type	-	`string`	0 - NORMAL (default) 1 - STACKED 2 - PIE 3 - EXPLODED	Graph type.
	show_legend	-	`string`	0 - NO 1 - YES (default)	Display graph legend.
	show_3d	-	`string`	0 - NO (default) 1 - YES	Enable 3D style. Used by pie and exploded pie graphs.
	percent_left	-	`double`	Default:0	Show the percentile line for left axis. Used only for normal graphs.
	percent_right	-	`double`	Default:0	Show the percentile line for right axis. Used only for normal graphs.
	ymin_type_1	-	`string`	0 - CALCULATED (default) 1 - FIXED 2 - ITEM	Minimum value of Y axis. Used by normal and stacked graphs.
	ymax_type_1	-	`string`	0 - CALCULATED (default) 1 - FIXED 2 - ITEM	Maximum value of Y axis. Used by normal and stacked graphs.

Element	Element property	Required	Type	Range[1]	Description
ymin_item_1		-			Individual item details. Required if 'ymin_type_1' is ITEM.
	host	x	string		Item host.
	key	x	string		Item key.
ymax_item_1		-			Individual item details. Required if 'ymax_type_1' is ITEM.
	host	x	string		Item host.
	key	x	string		Item key.
graph_items		x			Root element for graph items.
graph_item		x			Individual graph item.
	sortorder	-	integer		Draw order. The smaller value is drawn first. Can be used to draw lines or regions behind (or in front of) another.
	drawtype	-	string	0 - SINGLE_LINE (default) 1 - FILLED_REGION 2 - BOLD_LINE 3 - DOTTED_LINE 4 - DASHED_LINE 5 - GRADIENT_LINE	Draw style of the graph item. Used only by normal graphs.
	color	-	string		Element color (6 symbols, hex).
	yaxisside	-	string	0 - LEFT (default) 1 - RIGHT	Side of the graph where the graph item's Y scale will be drawn. Used by normal and stacked graphs.
	calc_fnc	-	string	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)	Data to draw if more than one value exists for an item.
	type	-	string	0 - SIMPLE (default) 2 - GRAPH_SUM (value of the item represents the whole pie; used only by pie and exploded pie graphs)	Graph item type.
item		x			Individual item.
	host	x	string		Item host.
	key	x	string		Item key.

Host web scenario tags

Element	Element property	Required	Type	Range[1]	Description
httptests		-			Root element for web scenarios.
httptest		-			Individual web scenario.
	name	x	`string`		Web scenario name.
	delay	-	`string`	Default: 1m	Frequency of executing the web scenario. Seconds, time unit with suffix or user macro.
	attempts	-	`integer`	1-10 (default: 1)	The number of attempts for executing web scenario steps.
	agent	-	`string`	Default: Zabbix	Client agent. Zabbix will pretend to be the selected browser. This is useful when a website returns different content for different browsers.
	http_proxy	-	`string`		Specify an HTTP proxy to use, using the format: `http://[username[:passw`
variables		-			Root element for scenario-level variables (macros) that may be used in scenario steps.
variable		-			Individual variable.
	name	x	`text`		Variable name.
	value	x	`text`		Variable value.
headers		-			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.
header		-			Individual header.
	name	x	`text`		Header name.
	value	x	`text`		Header value.
	status	-	`string`	0 - ENABLED (default) 1 - DISABLED	Web scenario status.
	authentication	-	`string`	0 - NONE (default) 1 - BASIC 2 - NTLM	Authentication method.

Element	Element property	Required	Type	Range[1]	Description
	http_user	-	`string`		User name used for basic, HTTP or NTLM authentication.
	http_password	-	`string`		Password used for basic, HTTP or NTLM authentication.
	verify_peer	-	`string`	0 - NO (default) 1 - YES	Whether to validate that the host's certificate is authentic.
	verify_host	-	`string`	0 - NO (default) 1 - YES	Whether to validate that the host name for the connection matches the one in the host's certificate.
	ssl_cert_file	-	`string`		Name of the SSL certificate file used for client authentication (must be in PEM format).
	ssl_key_file	-	`string`		Name of the SSL private key file used for client authentication (must be in PEM format).
	ssl_key_password	-	`string`		SSL private key file password.
steps		x			Root element for web scenario steps.
step		x			Individual web scenario step.
	name	x	`string`		Web scenario step name.
	url	x	`string`		URL for monitoring.
query_fields		-			Root element for query fields - an array of HTTP fields that will be added to the URL when performing a request.
query_field		-			Individual query field.
	name	x	`string`		Query field name.
	value	-	`string`		Query field value.

Element	Element property	Required	Type	Range[1]	Description
posts		-			HTTP POST variables as a string (raw post data) or as an array of HTTP fields (form field data).
post_field		-			Individual post field.
	name	x	`string`		Post field name.
	value	x	`string`		Post field value.
variables		-			Root element of step-level variables (macros) that should be applied after this step. If the variable value has a 'regex:' prefix, then its value is extracted from the data returned by this step according to the regular expression pattern following the 'regex:' prefix
variable		-			Individual variable.
	name	x	`string`		Variable name.
	value	x	`string`		Variable value.
headers		-			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.
header		-			Individual header.
	name	x	`string`		Header name.
	value	x	`string`		Header value.
	follow_redirects	-	`string`	0 - NO 1 - YES (default)	Follow HTTP redirects.
	retrieve_mode	-	`string`	0 - BODY (default) 1 - HEADERS 2 - BOTH	HTTP response retrieve mode.
	timeout	-	`string`	Default: 15s	Timeout of step execution. Seconds, time unit with suffix or user macro.

Element	Element property	Required	Type	Range[1]	Description
	required	-	`string`		Text that must be present in the response. Ignored if empty.
	status_codes	-	`string`		A comma delimited list of accepted HTTP status codes. Ignored if empty. For example: 200-201,210-299

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.
[2] The prefix "snmpv3_", used for interface details in v4.4 and prior, was replaced with introduction of the `version` element.

4 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

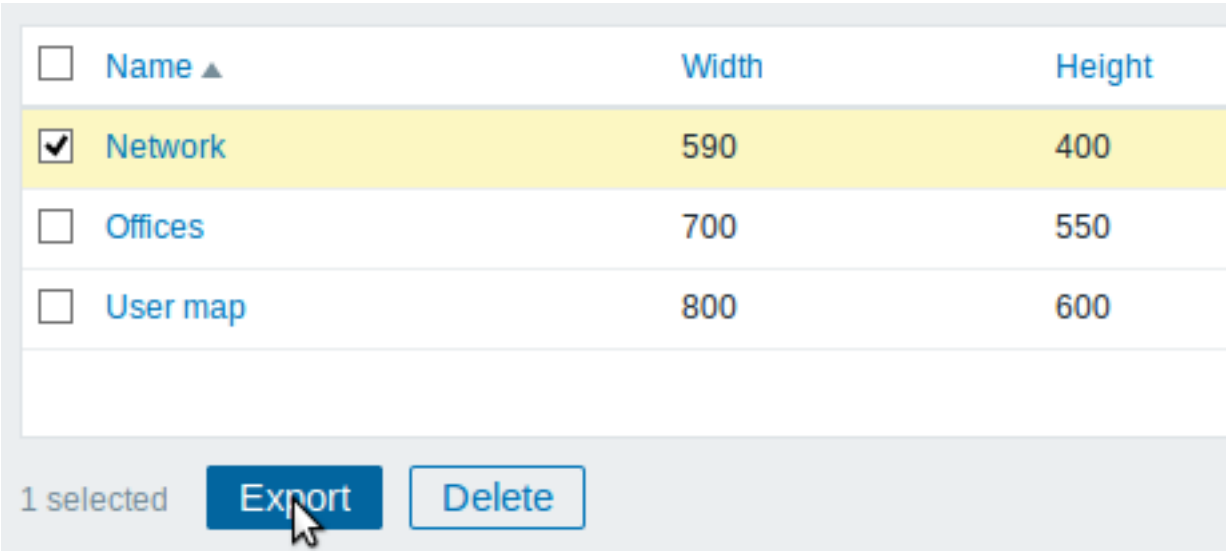
Not exported are host groups, hosts, triggers, other maps or any other elements that may be related to the exported map. Thus, if at least one of the elements the map refers to is missing, importing it will fail.

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

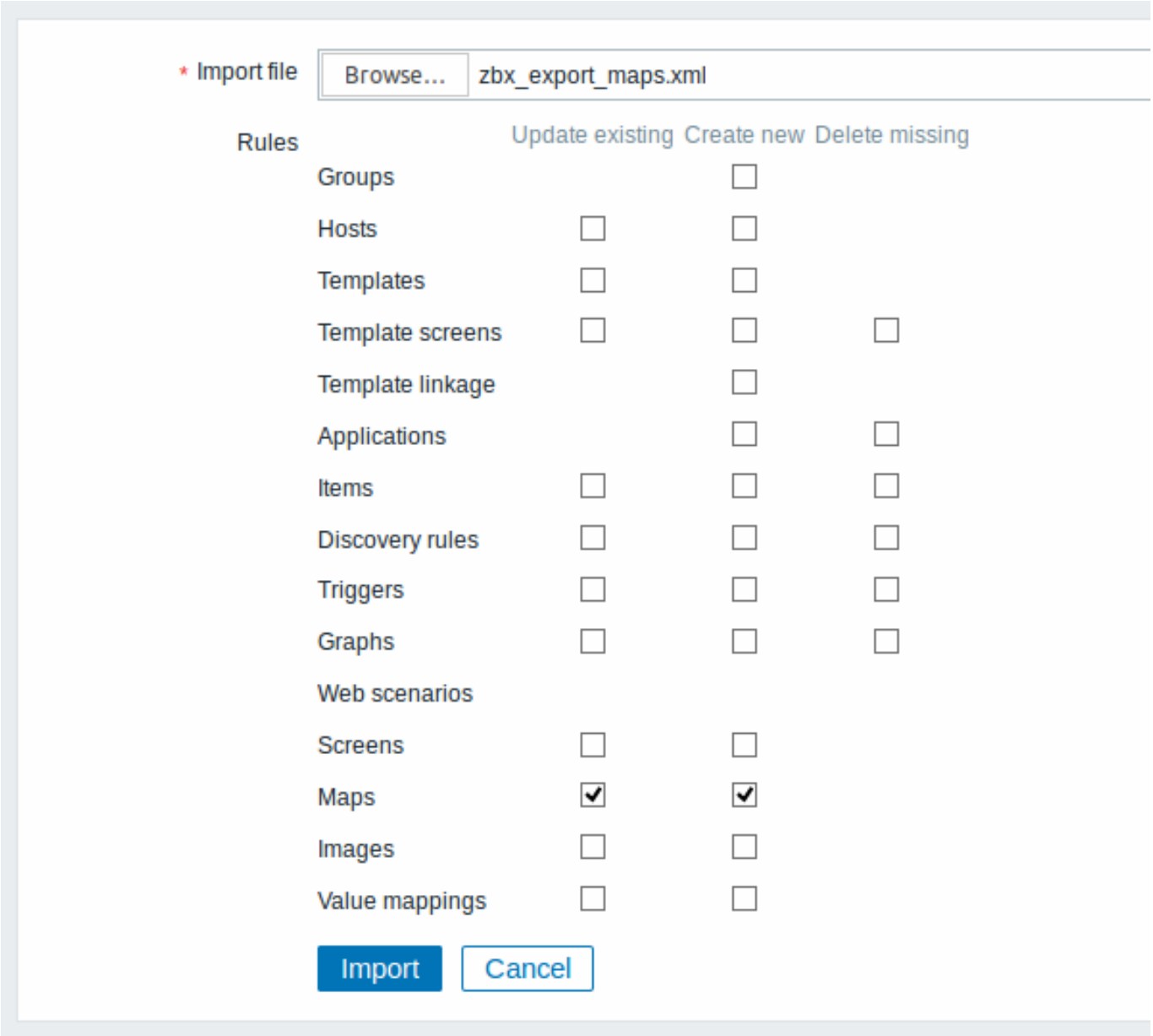


Selected maps are exported to a local XML file with default name *zabbix_export_maps.xml*.

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:

Rule	Description
Update existing	Existing maps will be updated with data taken from the import file. Otherwise they will not be updated.
Create new	The import will add new maps using data from the import file. Otherwise it will not add them.

If you uncheck both map options and check the respective options for images, images only will be imported. Image importing is only available to Zabbix Super Admin users.

Warning:
If replacing an existing image, it will affect all maps that are using this image.

Export format

Exporting a small network map with three elements, their images and some links between them. Note that images are truncated to save space.

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>5.0</version>
    <date>2021-08-31T12:46:49Z</date>
    <images>
        <image>
            <name>Zabbix_server_3D_(128)</name>
            <imagetype>1</imagetype>
            <encodedImage>iVBOR...CYII=</encodedImage>
        </image>
    </images>
    <maps>
        <map>
            <name>Local network</name>
            <width>680</width>
            <height>200</height>
            <label_type>0</label_type>
            <label_location>0</label_location>
            <highlight>1</highlight>
            <expandproblem>1</expandproblem>
            <markelements>1</markelements>
            <show_unack>0</show_unack>
            <severity_min>0</severity_min>
            <show_suppressed>0</show_suppressed>
            <grid_size>50</grid_size>
            <grid_show>1</grid_show>
            <grid_align>1</grid_align>
            <label_format>0</label_format>
            <label_type_host>2</label_type_host>
            <label_type_hostgroup>2</label_type_hostgroup>
            <label_type_trigger>2</label_type_trigger>
            <label_type_map>2</label_type_map>
            <label_type_image>2</label_type_image>
            <label_string_host/>
            <label_string_hostgroup/>
            <label_string_trigger/>
            <label_string_map/>
            <label_string_image/>
            <expand_macros>1</expand_macros>
            <background/>
            <iconmap/>
            <urls/>
            <selements>
                <selement>
                    <elementtype>0</elementtype>
                    <label>{HOST.NAME}
{HOST.CONN}</label>
                    <label_location>0</label_location>
                    <x>111</x>
                    <y>61</y>
                    <elementsubtype>0</elementsubtype>
                    <areatype>0</areatype>
                    <width>200</width>
                    <height>200</height>
                    <viewtype>0</viewtype>
                    <use_iconmap>0</use_iconmap>
                    <selementid>1</selementid>
                    <elements>
                        <element>
```

```
                        <host>Zabbix server</host>
                    </element>
                </elements>
                <icon_off>
                    <name>Zabbix_server_3D_(128)</name>
                </icon_off>
                <icon_on/>
                <icon_disabled/>
                <icon_maintenance/>
                <application/>
                <urls/>
            </selement>
        </selements>
        <shapes>
            <shape>
                <type>0</type>
                <x>0</x>
                <y>0</y>
                <width>680</width>
                <height>15</height>
                <border_type>0</border_type>
                <border_width>0</border_width>
                <border_color>000000</border_color>
                <text>{MAP.NAME}</text>
                <font>9</font>
                <font_size>11</font_size>
                <font_color>000000</font_color>
                <text_halign>0</text_halign>
                <text_valign>0</text_valign>
                <background_color/>
                <zindex>0</zindex>
            </shape>
        </shapes>
        <lines/>
        <links/>
    </map>
  </maps>
</zabbix_export>
```

Element tags

Element tag values are explained in the table below.

Element	Element property	Type	Range	Description
images				Root element for images.
image				Individual image.
	name	string		Unique image name.
	imagetype	integer	1 - image 2 - background	Image type.
	encodedImage			Base64 encoded image.
maps				Root element for maps.
map				Individual map.
	name	string		Unique map name.
	width	integer		Map width, in pixels.
	height	integer		Map height, in pixels.

Element	Element property	Type	Range	Description
	label_type	integer	0 - label 1 - host IP address 2 - element name 3 - status only 4 - nothing	Map element label type.
	label_location	integer	0 - bottom 1 - left 2 - right 3 - top	Map element label location by default.
	highlight	integer	0 - no 1 - yes	Enable icon highlighting for active triggers and host statuses.
	expandproblem	integer	0 - no 1 - yes	Display problem trigger for elements with a single problem.
	markelements	integer	0 - no 1 - yes	Highlight map elements that have recently changed their status.
	show_unack	integer	0 - count of all problems 1 - count of unacknowledged problems 2 - count of acknowledged and unacknowledged problems separately	Problem display.
	severity_min	integer	0 - not classified 1 - information 2 - warning 3 - average 4 - high 5 - disaster	Minimum trigger severity to show on the map by default.
	show_suppressed	integer	0 - no 1 - yes	Display problems which would otherwise be suppressed (not shown) because of host maintenance.
	grid_size	integer	20, 40, 50, 75 or 100	Cell size of a map grid in pixels, if ”grid_show=1”
	grid_show	integer	0 - yes 1 - no	Display a grid in map configuration.
	grid_align	integer	0 - yes 1 - no	Automatically align icons in map configuration.
	label_format	integer	0 - no 1 - yes	Use advanced label configuration.

Element	Element property	Type	Range	Description
	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_hostgroup	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		Custom label for host elements, if ”label_type_host=5”
	label_string_hostgroup	string		Custom label for host group elements, if ”label_type_hostgroup=5”
	label_string_trigger	string		Custom label for trigger elements, if ”label_type_trigger=5”
	label_string_map	string		Custom label for map elements, if ”label_type_map=5”
	label_string_image	string		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).

Element	Element property	Type	Range	Description
urls				Used by maps or each map element.
url				Individual URL.
	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.
selements				
selement				Individual map element.
	elementtype	`integer`	0 - host 1 - map 2 - trigger 3 - host group 4 - image	Map element type.
	label	`string`		Icon label.
	label_location	`integer`	-1 - use map default 0 - bottom 1 - left 2 - right 3 - top	
	x	`integer`		Location on the X axis.
	y	`integer`		Location on the Y axis.
	elementsubtype	`integer`	0 - single host group 1 - all host groups	Element subtype, if ”elementtype=3”
	areatype	`integer`	0 - same as whole map 1 - custom size	Area size, if ”elementsubtype=1”
	width	`integer`		Width of area, if ”areatype=1”
	height	`integer`		Height of area, if ”areatype=1”
	viewtype	`integer`	0 - place evenly in the area	Area placement algorithm, if ”elementsubtype=1”
	use_iconmap	`integer`	0 - no 1 - yes	Use icon mapping for this element. Relevant only if iconmapping is activated on map level.
	selementid	`id`		Unique element record ID.
	application	`string`		Application name filter. If an application name is given, only problems of triggers that belong to the given application will be displayed on the map.
elements				
element				Individual Zabbix entity that is represented on the map (map, hostgroup, host, etc).
	host			
icon_off				Image to use when element is in ’OK’ status.

Element	Element property	Type	Range	Description
icon_on				Image to use when element is in 'Problem' status.
icon_disabled				Image to use when element is disabled.
icon_maintenance				Image to use when element is in maintenance.
	name	`string`		Unique image name.
shapes				
shape				Individual shape.
	type	`integer`	0 - rectangle 1 - ellipse	Shape type.
	x	`integer`		X coordinates of the shape in pixels.
	y	`integer`		Y coordinates of the shape in pixels.
	width	`integer`		Shape width.
	height	`integer`		Shape height.
	border_type	`integer`	0 - none 1 - bold line 2 - dotted line 3 - dashed line	Type of the border for the shape.
	border_width	`integer`		Width of the border in pixels.
	border_color	`string`		Border color represented in hexadecimal code.
	text	`string`		Text inside of shape.

Element	Element property	Type	Range	Description
	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.

Element	Element property	Type	Range	Description
	background_color	string		Background (fill) color represented in hexadecimal code.
	zindex	integer		Value used to order all shapes and lines (z-index).
lines				
line				Individual line.
	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).
links				
link				Individual link between map elements.
	drawtype	integer	0 - line 2 - bold line 3 - dotted line 4 - dashed line	Link style.
	color	string		Link color (6 symbols, hex).
	label	string		Link label.
	selementid1	id		ID of one element to connect.
	selementid2	id		ID of the other element to connect.
linktriggers				
linktrigger				Individual link status indicator.
	drawtype	integer	0 - line 2 - bold line 3 - dotted line 4 - dashed line	Link style when trigger is in the 'Problem' state.
	color	string		Link color (6 symbols, hex) when trigger is in the 'Problem' state.
trigger				Trigger used for indicating link status.
	description	string		Trigger name.
	expression	string		Trigger expression.
	recovery_expression	string		Trigger recovery expression.

5 Screens

Overview

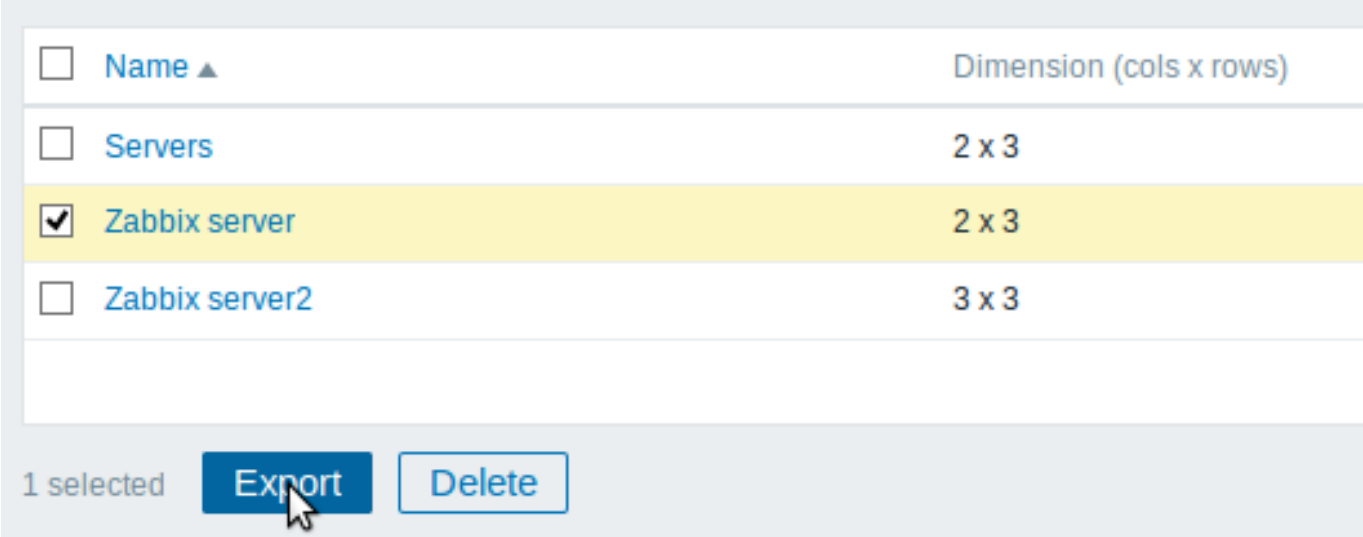
Screen export contains the screen structure - all screen settings and all screen elements along with their configuration.

Anything included in the screen itself (like a host, host group or any other data) is not exported. Thus, if at least one of the elements the screen refers to is missing, importing it will fail.

Exporting

To export screens, do the following:

- Go to: *Monitoring* → *Screens*
- Mark the checkboxes of the screens to export
- Click on *Export* below the list

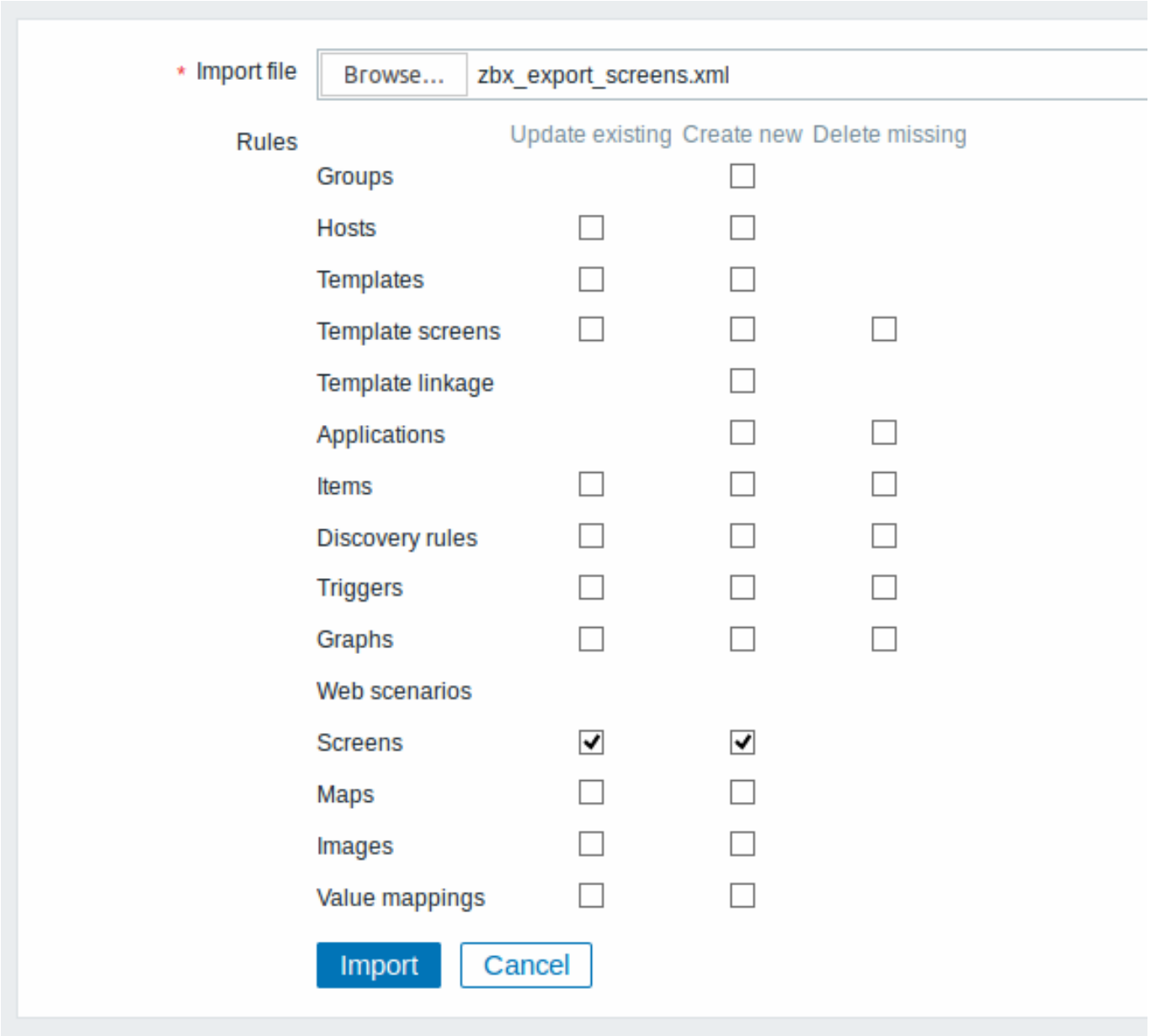


Selected screens are exported to a local XML file with default name *zabbix_export_screens.xml*.

Importing

To import screens, do the following:

- Go to: *Monitoring* → *Screens*
- 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:

Rule	Description
Update existing	Existing screens will be updated with data taken from the import file. Otherwise they will not be updated.
Create new	The import will add new screens using data from the import file. Otherwise it will not add them.

Export format

Exporting a small screen with two graphs taking up the first row of the screen.

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>5.0</version>
    <date>2020-04-22T09:22:17Z</date>
    <screens>
        <screen>
            <name>Zabbix server</name>
            <hsize>2</hsize>
```

```
            <vsize>3</vsize>
            <screen_items>
                <screen_item>
                    <resourcetype>0</resourcetype>
                    <width>300</width>
                    <height>80</height>
                    <x>0</x>
                    <y>0</y>
                    <colspan>1</colspan>
                    <rowspan>1</rowspan>
                    <elements>0</elements>
                    <valign>0</valign>
                    <halign>0</halign>
                    <style>0</style>
                    <url/>
                    <dynamic>1</dynamic>
                    <sort_triggers>0</sort_triggers>
                    <resource>
                        <name>CPU load</name>
                        <host>Zabbix host</host>
                    </resource>
                    <max_columns>3</max_columns>
                    <application/>
                </screen_item>
                <screen_item>
                    <resourcetype>0</resourcetype>
                    <width>300</width>
                    <height>80</height>
                    <x>1</x>
                    <y>0</y>
                    <colspan>1</colspan>
                    <rowspan>1</rowspan>
                    <elements>0</elements>
                    <valign>0</valign>
                    <halign>0</halign>
                    <style>0</style>
                    <url/>
                    <dynamic>1</dynamic>
                    <sort_triggers>0</sort_triggers>
                    <resource>
                        <name>CPU utilization</name>
                        <host>Zabbix host</host>
                    </resource>
                    <max_columns>3</max_columns>
                    <application/>
                </screen_item>
            </screen_items>
        </screen>
    </screens>
</zabbix_export>
```

Element tags

Element tag values are explained in the table below.

Element	Element property	Type	Range	Description
screens				
screen				
	name	string		Unique screen name.
	hsize	integer		Horizontal size, number of columns.
	vsize	integer		Vertical size, number of rows.

Element	Element property	Type	Range	Description
screen_items screen_item				
	resourcetype	`integer`	0 - graph 1 - simple graph 2 - map 3 - plain text 4 - host info 5 - trigger info 6 - server info 7 - clock 9 - trigger overview 10 - data overview 11 - URL 12 - history of actions 13 - history of events 14 - host group issues 15 - problems by severity 16 - host issues 19 - simple graph prototype 20 - graph prototype	Resource type.
	width	`integer`		Width of the screen item (in pixels) if ’resourcetype’ is 0, 1, 7, 11, 19 or 20.
	height	`integer`		Height of the screen item (in pixels) if ’resourcetype’ is 0, 1, 7, 11, 19 or 20.
	x	`integer`		X-coordinates of the screen item on the screen, from left to right. ’0’ means start from first column.
	y	`integer`		Y-coordinates of the screen item on the screen, from top to bottom. ’0’ means start from first row.
	colspan	`integer`		Number of columns the screen item will span across.
	rowspan	`integer`		Number or rows the screen item will span across.
	elements	`integer`		Number of lines to display on the screen item if ’resourcetype’ is 3, 12, 13, 14 or 16.

Element	Element property	Type	Range	Description
	valign	integer	0 - middle (default) 1 - top 2 - bottom	Vertical alignment.
	halign	integer	0 - center (default) 1 - left 2 - right	Horizontal alignment.
	style	integer	0 - plain text 1 - HTML	Option of displaying screen item if 'resourcetype' is 3.
		integer	0 - local time 1 - server time 2 - host time	Option of displaying screen item if 'resourcetype' is 7.
		integer	0 - horizontal 1 - vertical	Option of displaying screen item if 'resourcetype' is 4,5.
		integer	0 - left side 1 - top	Option of displaying screen item if 'resourcetype' is 9,10.
	url	string		Link URL if 'resourcetype' is 11.
	dynamic	integer	0 - no 1 - yes	Make screen item dynamic if 'resourcetype' is 0, 1, 3, 19 or 20.
	sort_triggers	integer	0 - last change (descending) 1 - severity (descending) 2 - host (ascending)	Option to sort triggers by if 'resourcetype' is 14,16.
		integer	3 - time (ascending) 4 - time (descending) 5 - type (ascending) 6 - type (descending) 7 - status (ascending) 8 - status (descending) 9 - retries left (ascending) 10 - retries left (descending) 11 - recipient (ascending) 12 - recipient (descending)	Option to sort triggers by if 'resourcetype' is 12.
	max_columns	integer		In how many columns generated graphs should be displayed in the screen cell if 'resourcetype' is 19 or 20. Useful when there are many LLD-generated graphs.

Element	Element property	Type	Range	Description
	application	`string`		Filter by application name if 'resourcetype' is 9 or 10.
resource				
	name	`string`		Resource name.
	host	`string`		Resource host.

6 Media types

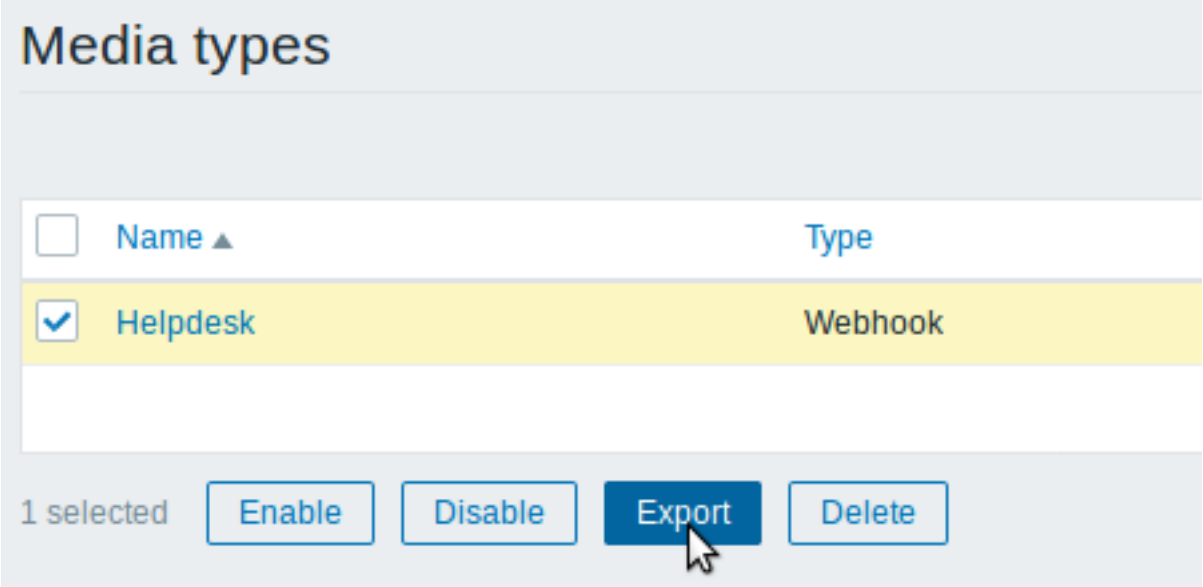
Overview

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

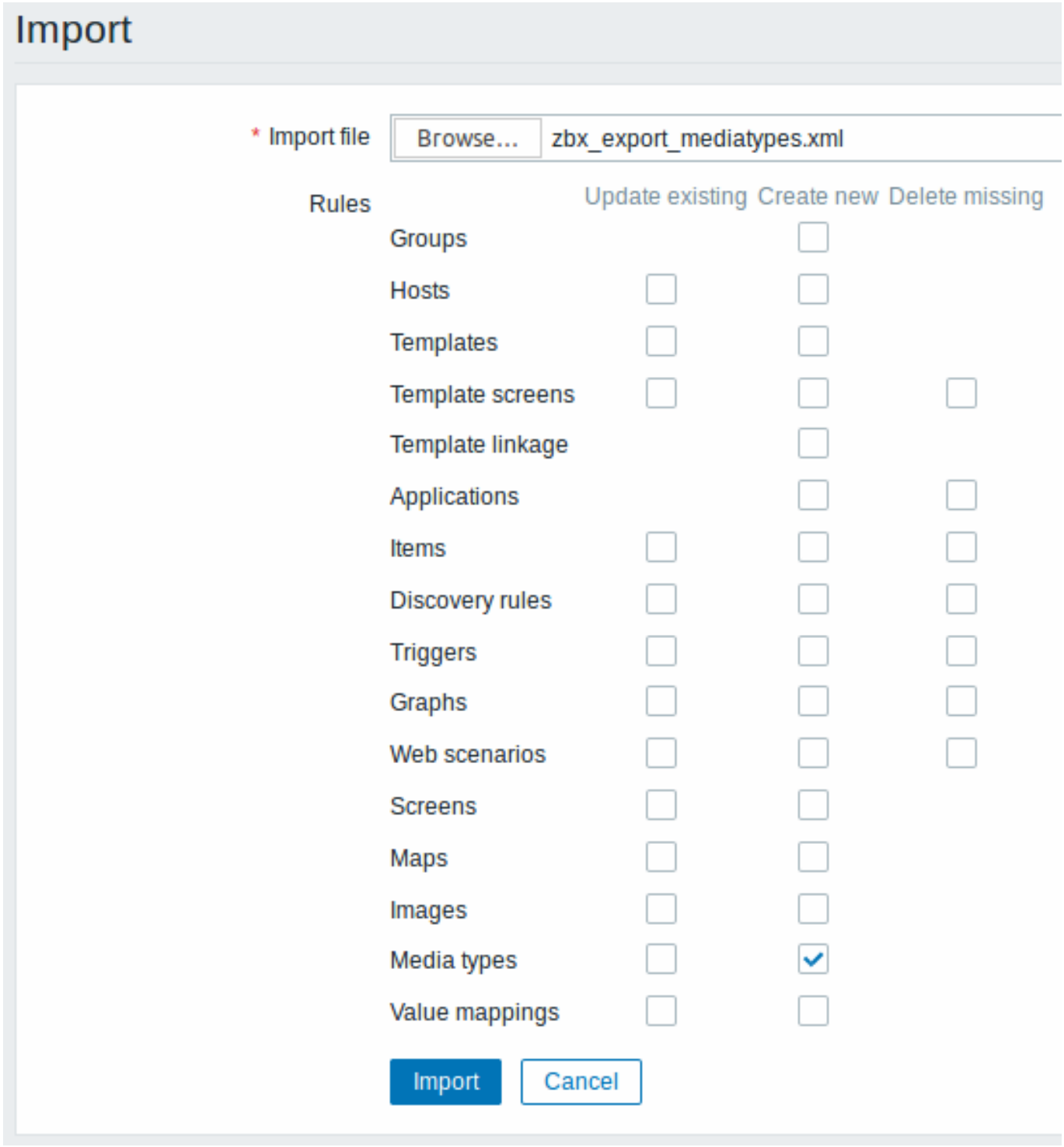


Selected media types are exported to a local XML file with default name *zbx_export_mediatypes.xml*.

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:

Rule	Description
Update existing	Existing elements will be updated with data taken from the import file. Otherwise they will not be updated.
Create new	The import will add new elements using data from the import file. Otherwise it will not add them.
Delete missing	The import will remove existing elements not present in the import file. Otherwise it will not remove them.

Export format

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
```

```
    <version>5.0</version>
    <date>2020-01-24T06:44:38Z</date>
    <media_types>
        <media_type>
            <name>Slack chat</name>
            <type>WEBHOOK</type>
            <parameters>
                <parameter>
                    <name>channel</name>
                    <value>{ALERT.SENDTO}</value>
                </parameter>
                <parameter>
                    <name>text</name>
                    <value>{ALERT.MESSAGE}</value>
                </parameter>
                <parameter>
                    <name>username</name>
                    <value>bot</value>
                </parameter>
            </parameters>
            <script>var req = new CurlHttpRequest();
req.AddHeader('Content-Type: application/x-www-form-urlencoded');

Zabbix.Log(127, 'webhook requqest value='+value);

req.Post('https://hooks.slack.com/services/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',
  'payload='+value
);

Zabbix.Log(127, 'response code: '+req.Status());

return JSON.stringify({
  'tags': {
    'delivered': 'slack'
  }
});</script>
            <process_tags>YES</process_tags>
            <show_event_menu>YES</show_event_menu>
            <event_menu_url>https://www.zabbix.com</event_menu_url>
            <event_menu_name>Slack message</event_menu_name>
            <description>Slack chat messages.</description>
        </media_type>
    </media_types>
</zabbix_export>
```

Element tags

Element tag values are explained in the table below.

Element	Element property	Required	Type	Range[1]	Description
media_types		-			Root element for media_types.
media_type		-			Individual media_type.
	name	x	string		Media type name.
	type	x	string	0 - EMAIL 1 - SMS 2 - SCRIPT 4 - WEBHOOK	Transport used by the media type.
	status	-	string	0 - ENABLED (default) 1 - DISABLED	Whether the media type is enabled.

Element	Element property	Required	Type	Range[1]	Description
	max_sessions	-	integer	Possible values for SMS: 1 - (default) Possible values for other media types: 0-100, 0 - unlimited	The maximum number of alerts that can be processed in parallel.
	attempts	-	integer	1-10 (default: 3)	The maximum number of attempts to send an alert.
	attempt_interval	-	string	0-60s (default: 10s)	The interval between retry attempts. Accepts seconds and time unit with suffix.
	description	-	string		Media type description.
message_templates		-			Root element for media type message templates.
message_template		-			Individual message template.
	event_source	x	string	0 - TRIGGERS 1 - DISCOVERY 2 - AUTOREGISTRATION 3 - INTERNAL	Event source.
	operation_mode	x	string	0 - PROBLEM 1 - RECOVERY 2 - UPDATE	Operation mode.
	subject	-	string		Message subject.
	message	-	string		Message body.
Used only by e-mail media type					
	smtp_server	x	string		SMTP server.
	smtp_port	-	integer	Default: 25	SMTP server port to connect to.
	smtp_helo	x	string		SMTP helo.
	smtp_email	x	string		Email address from which notifications will be sent.
	smtp_security	-	string	0 - NONE (default) 1 - STARTTLS 2 - SSL_OR_TLS	SMTP connection security level to use.
	smtp_verify_host	-	string	0 - NO (default) 1 - YES	SSL verify host for SMTP. Optional if smtp_security is STARTTLS or SSL_OR_TLS.

Element	Element property	Required	Type	Range[1]	Description
	smtp_verify_peer	-	string	0 - NO (default) 1 - YES	SSL verify peer for SMTP. Optional if smtp_security is STARTTLS or SSL_OR_TLS.
	smtp_authentication	-	string	0 - NONE (default) 1 - PASSWORD	SMTP authentication method to use.
	username	-	string		Username.
	password	-	string		Authentication password.
	content_type	-	string	0 - TEXT 1 - HTML (default)	Message format.
Used only by SMS media type					
	gsm_modem	x	string		Serial device name of the GSM modem.
Used only by script media type					
	script name	x	string		Script name.
parameters		-			Root element for script parameters.
parameter		-			Individual script parameter.
Used only by webhook media type					
	script	x	string		Script.
	timeout	-	string	1-60s (default: 30s)	Javascript script HTTP request timeout interval.
	process_tags	-	string	0 - NO (default) 1 - YES	Whether to process returned tags.
	show_event_menu	-	string	0 - NO (default) 1 - YES	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.
	event_menu_url	-	string		URL of the event menu entry. Supports {EVENT.TAGS.*} macro.

Element	Element property	Required	Type	Range[1]	Description
	event_menu_name	-	`string`		Name of the event menu entry. Supports {EVENT.TAGS.*} macro.
parameters		-			Root element for webhook media type parameters.
parameter		-			Individual webhook media type parameter.
	name	x	`string`		Webhook parameter name.
	value	-	`string`		Webhook parameter value.

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.

Note:
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.

Event	Check of service result
Service Discovered	The service is 'up' after it was 'down' or when discovered for the first time.
Service Up	The service is 'up', after it was already 'up'.
Service Lost	The service is 'down' after it was 'up'.
Service Down	The service is 'down', after it was already 'down'.
Host Discovered	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.
Host Up	At least one service of a host is 'up', after at least one service was already 'up'.
Host Lost	All services of a host are 'down' after at least one was 'up'.
Host Down	All services of a host are 'down', after they were already 'down'.

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.

Note:
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.

The IP address of the discovered device is the criterion for finding a host in the system. If a host with that IP address and interface type already exists, that host will be the target for performing operations.

If the IP address of the discovered host is changed or the interface is deleted, a new host will be created upon the next discovery.

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

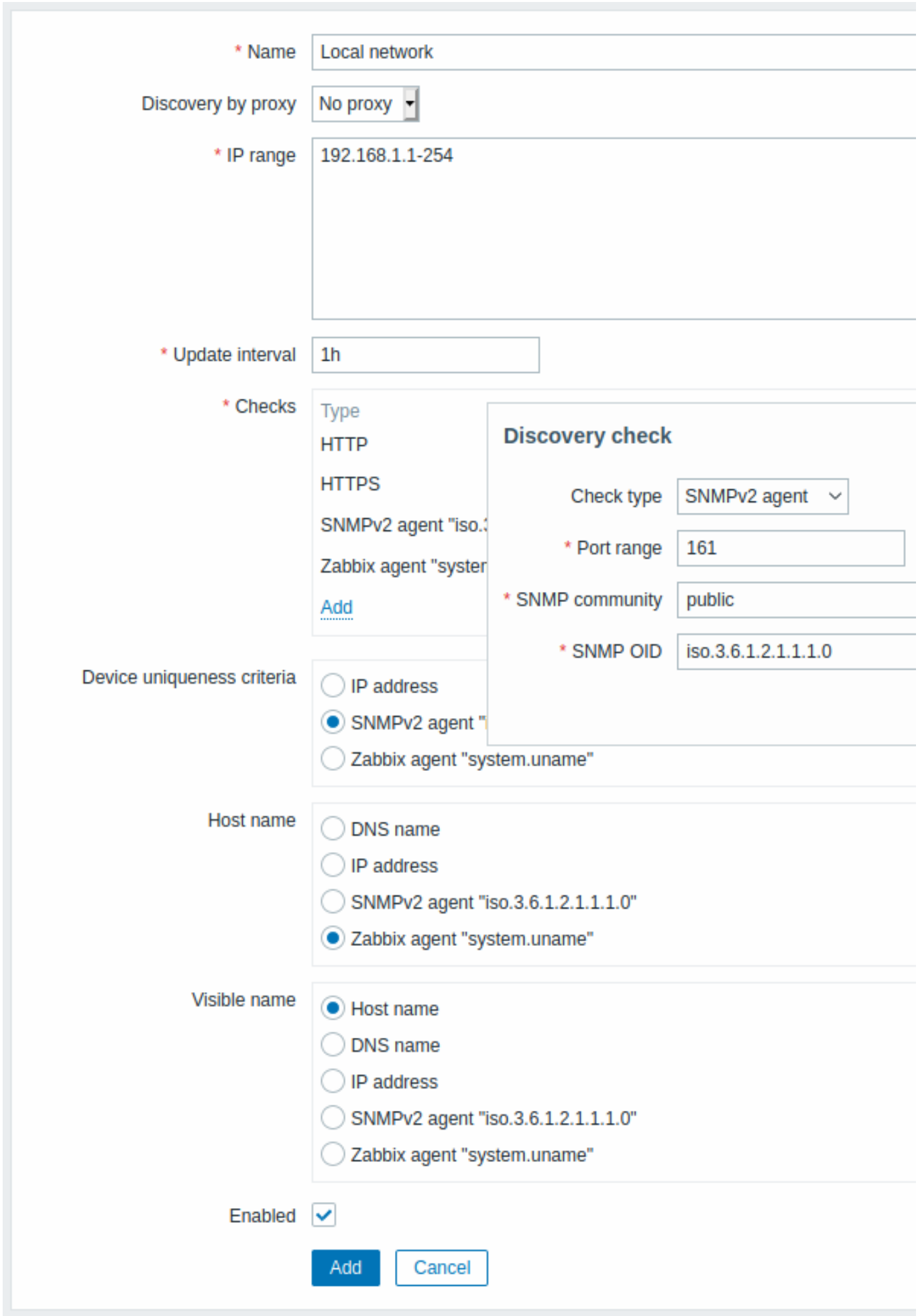
1 Configuring a network discovery rule

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.

Parameter	Description
Name	Unique name of the rule. For example, ”Local network”.
Discovery by proxy	What performs discovery: **no proxy** - Zabbix server is doing discovery **<proxy name>** - this proxy performs discovery
IP range	The range of IP addresses for discovery. It may have the following formats: Single IP: 192.168.1.33 Range of IP addresses: 192.168.1-10.1-255. The range is limited by the total number of covered addresses (less than 64K). IP mask: 192.168.4.0/24 supported IP masks: /16 - /30 for IPv4 addresses /112 - /128 for IPv6 addresses List: 192.168.1.1-255, 192.168.2.1-100, 192.168.2.200, 192.168.4.0/24 Since Zabbix 3.0.0 this field supports spaces, tabulation and multiple lines.
Update interval	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. Time suffixes are supported, e.g. 30s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. *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).
Checks	Zabbix will use this list of checks for discovery. Click on Add to configure a new check in a popup window. Supported checks: SSH, LDAP, SMTP, FTP, HTTP, HTTPS, POP, NNTP, IMAP, TCP, Telnet, Zabbix agent, SNMPv1 agent, SNMPv2 agent, SNMPv3 agent, ICMP ping. A protocol-based discovery uses the **net.tcp.service[]** 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 agent items for more details. The ’Ports’ parameter may be one of following: Single port: 22 Range of ports: 22-45 List: 22-45,55,60-70
Device uniqueness criteria	Uniqueness criteria may be: **IP address** - do not process 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. **<discovery check>** - either Zabbix agent or SNMP agent check. Note that uniqueness criteria used during discovery is not the same as host identification in the system when performing actions. Uniqueness criteria during discovery define whether two or more discovered devices are the same (or different), whereas only the IP address is the criterion for host identification in Zabbix (see Host creation).
Host name	Set the technical host name of a created host using: **DNS name** - DNS name (default) **IP address** - IP address **<discovery check>** - received string value of the discovery check (e.g. Zabbix agent, SNMP agent check) See also: Host naming. This option is supported since 4.2.0.

Parameter	Description
Visible name	Set the visible host name of a created host using: **Host name** - technical host name (default) **DNS name** - DNS name **IP address** - IP address **<discovery check>** - received string value of the discovery check (e.g. Zabbix agent, SNMP agent check) See also: Host naming. This option is supported since 4.2.0.
Enabled	With the check-box marked the rule is active and will be executed by Zabbix server. If unmarked, the rule is not active. It won't be executed.

A real life scenario

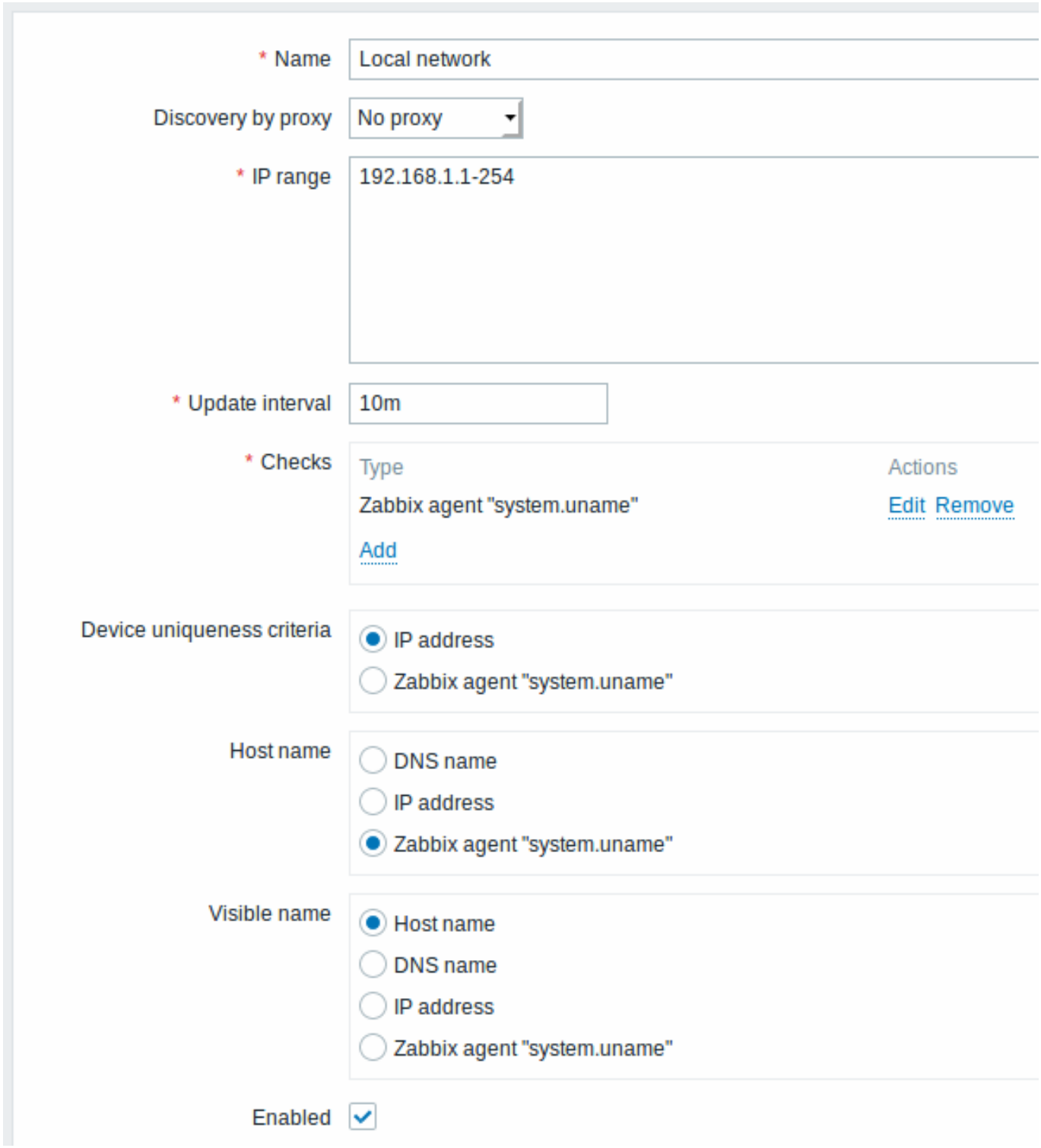
In this example we would like to set up network discovery for the local network having an IP range of 192.168.1.1-192.168.1.254.

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 *Template OS Linux* for Linux hosts
- use *Template OS Windows* for Windows hosts

Step 1

Defining a network discovery rule for our IP range.



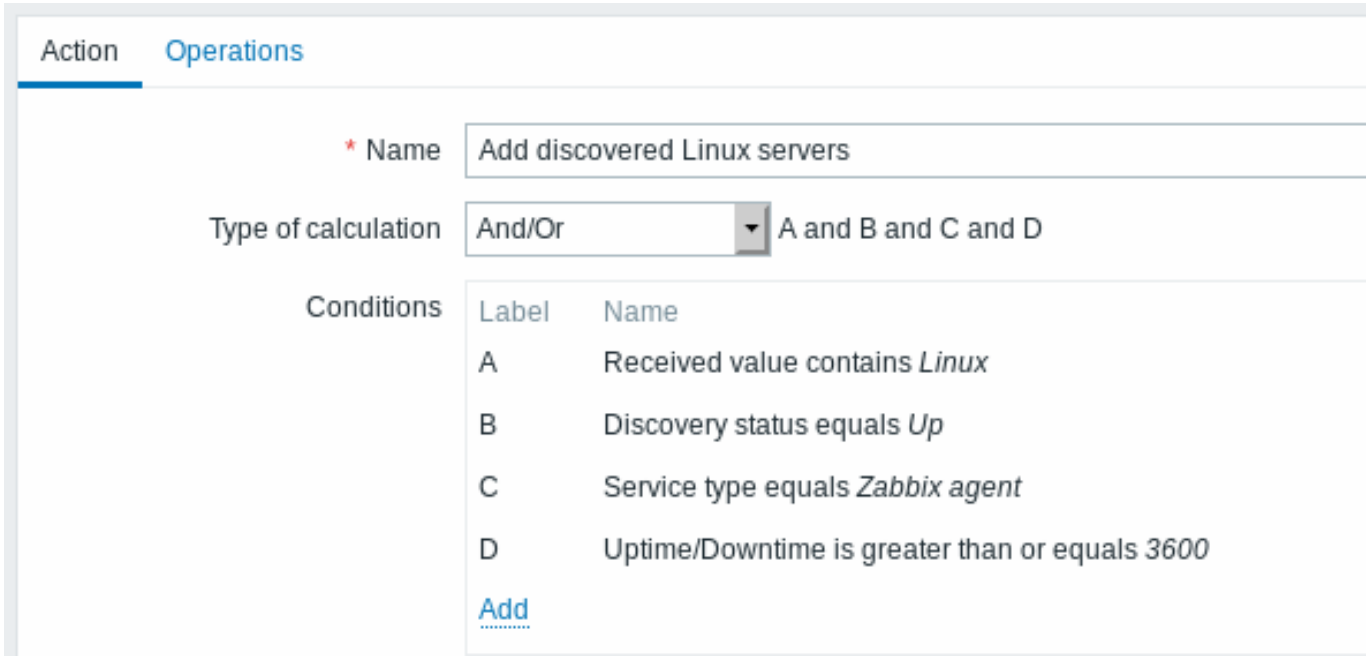
Zabbix will try to discover hosts in the IP range of 192.168.1.1-192.168.1.254 by connecting to Zabbix agents and getting the value from the **system.uname** key. The value received from the agent can be used to name the hosts and also to apply different actions for different operating systems. For example, link Windows servers to Template OS Windows, Linux servers to Template OS Linux.

The rule will be executed every 10 minutes.

When this rule is added, Zabbix will automatically start the discovery and generating discovery-based events for further processing.

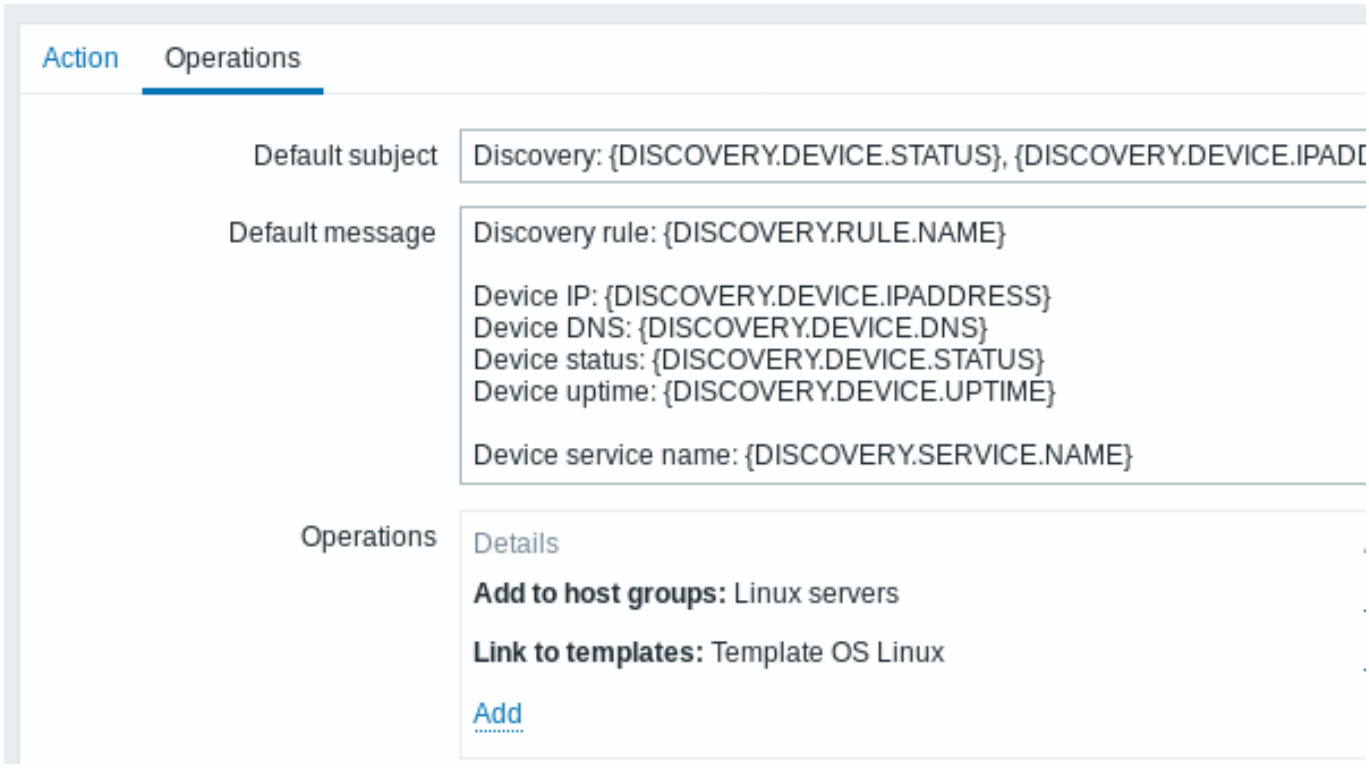
Step 2

Defining a discovery action for adding the discovered Linux servers to the respective group/template.



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

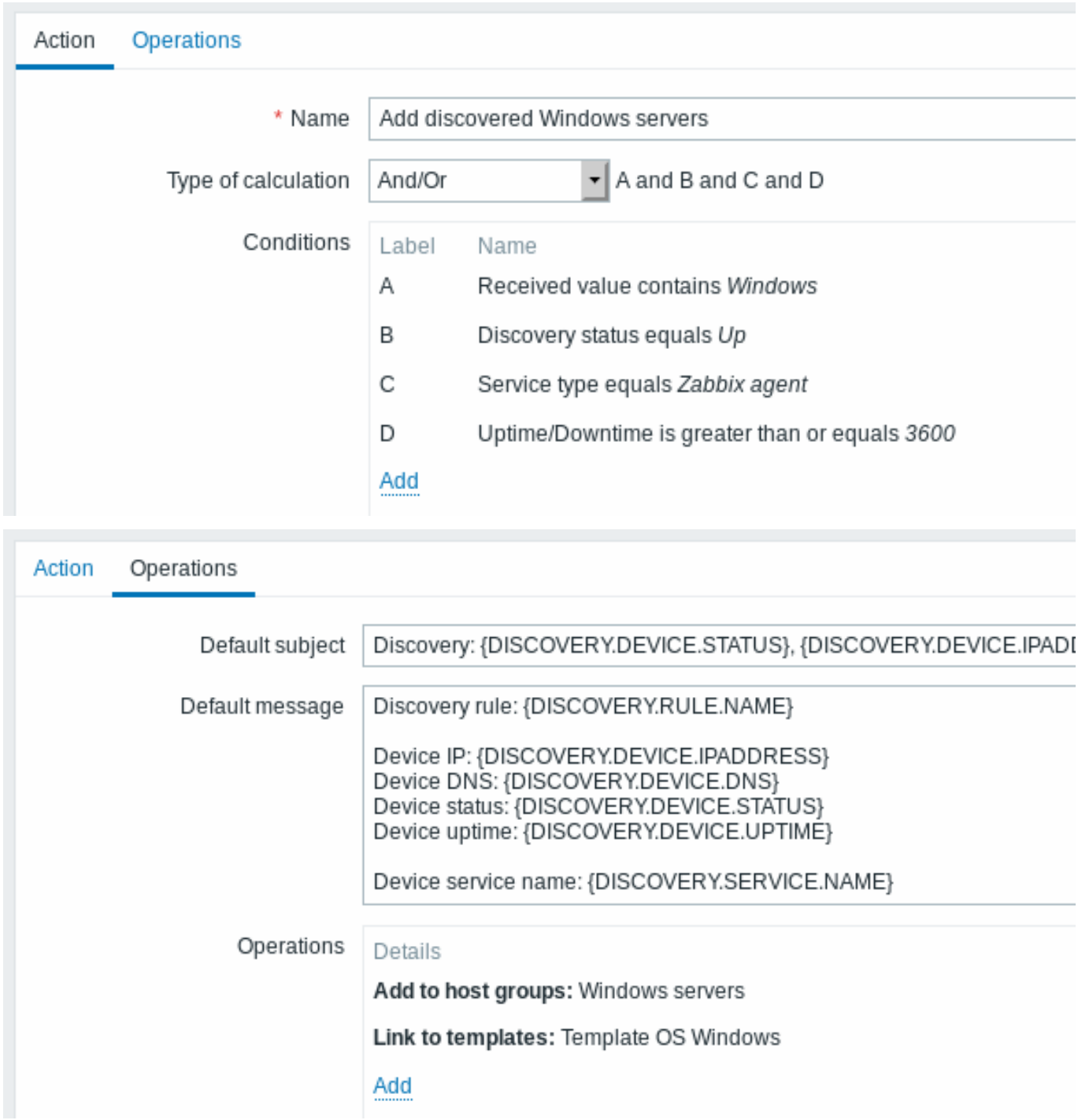


The action will execute the following operations:

- add the discovered host to the ”Linux servers” group (and also add host if it wasn’t added previously)
- link host to the ”Template OS Linux” template. Zabbix will automatically start monitoring the host using items and triggers from ”Template OS Linux”.

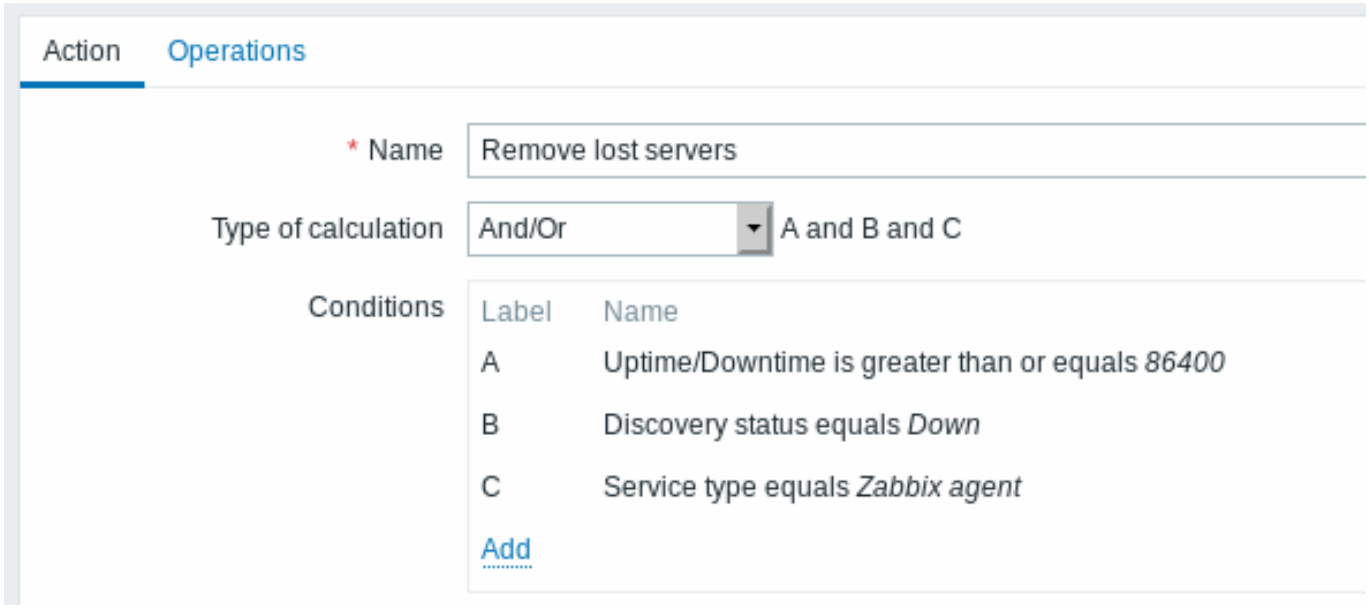
Step 3

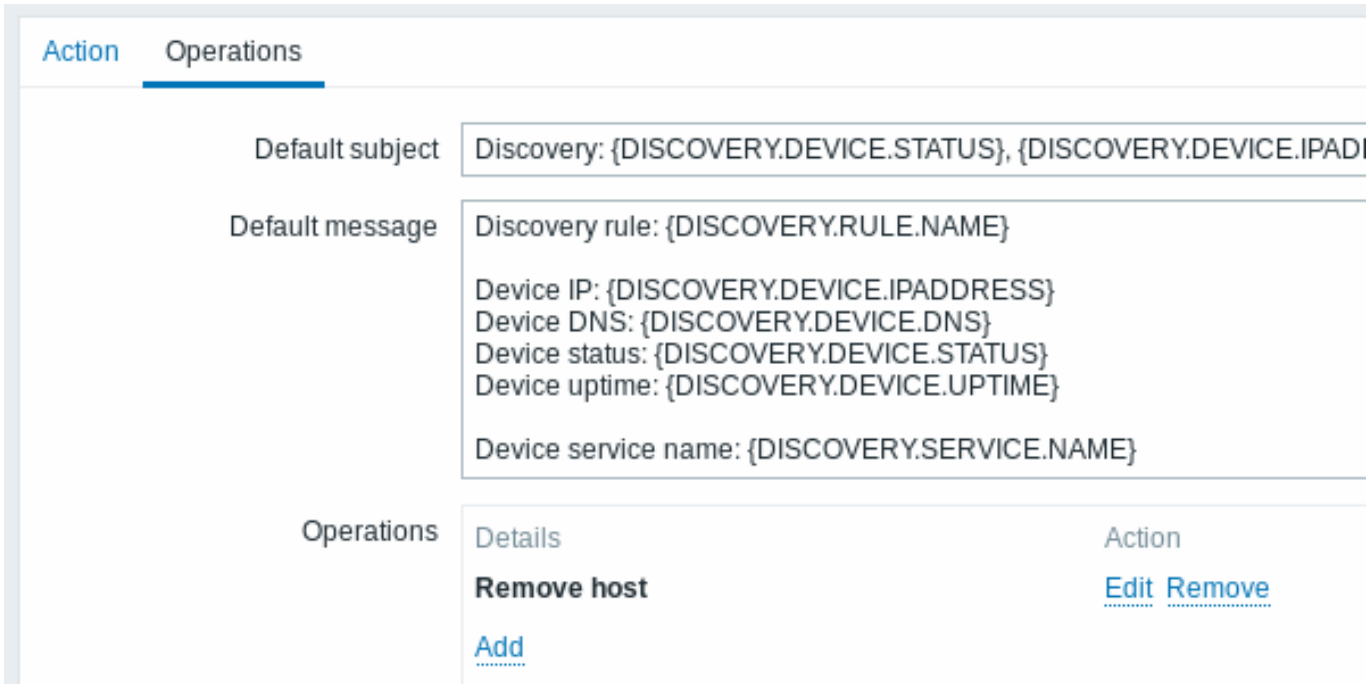
Defining a discovery action for adding the discovered Windows servers to the respective group/template.



Step 4

Defining a discovery action for removing lost servers.





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.

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.

Note:
Setting up network discovery is not required to have active agents autoregister.

In the Zabbix frontend, go to *Configuration* → *Actions*, select *Autoregistration actions* 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.

Note:
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.

Attention:
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.

Note:
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.

Attention:
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 "Template OS Linux" and "Template OS Windows" 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: Template OS Linux

Note:
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: Template OS 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: Template OS 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 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08
* Operations:
    * Send message to users: Admin via all media
    * Add to host groups: Linux servers
    * Link to templates: Template OS Linux
    * Link to templates: Template DB MySQL
```

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”.

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.

Warning:
As a result of the changes above, newer agents no longer will be able to work with an older Zabbix server.

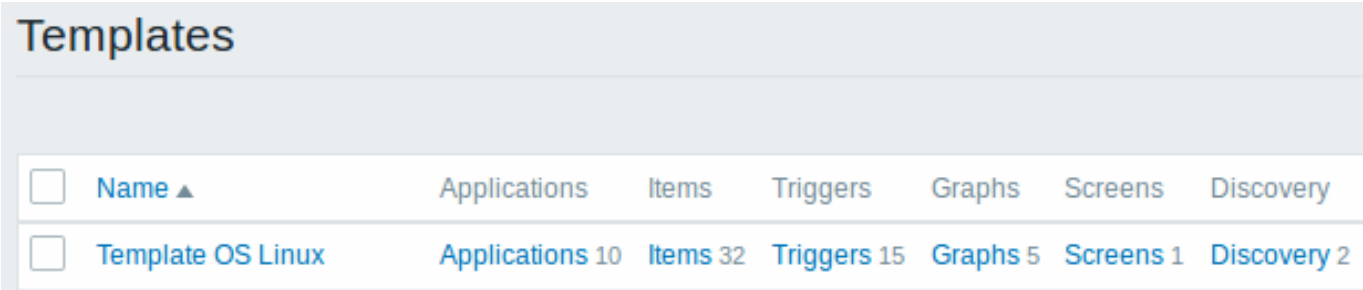
See also: Discovered entities

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



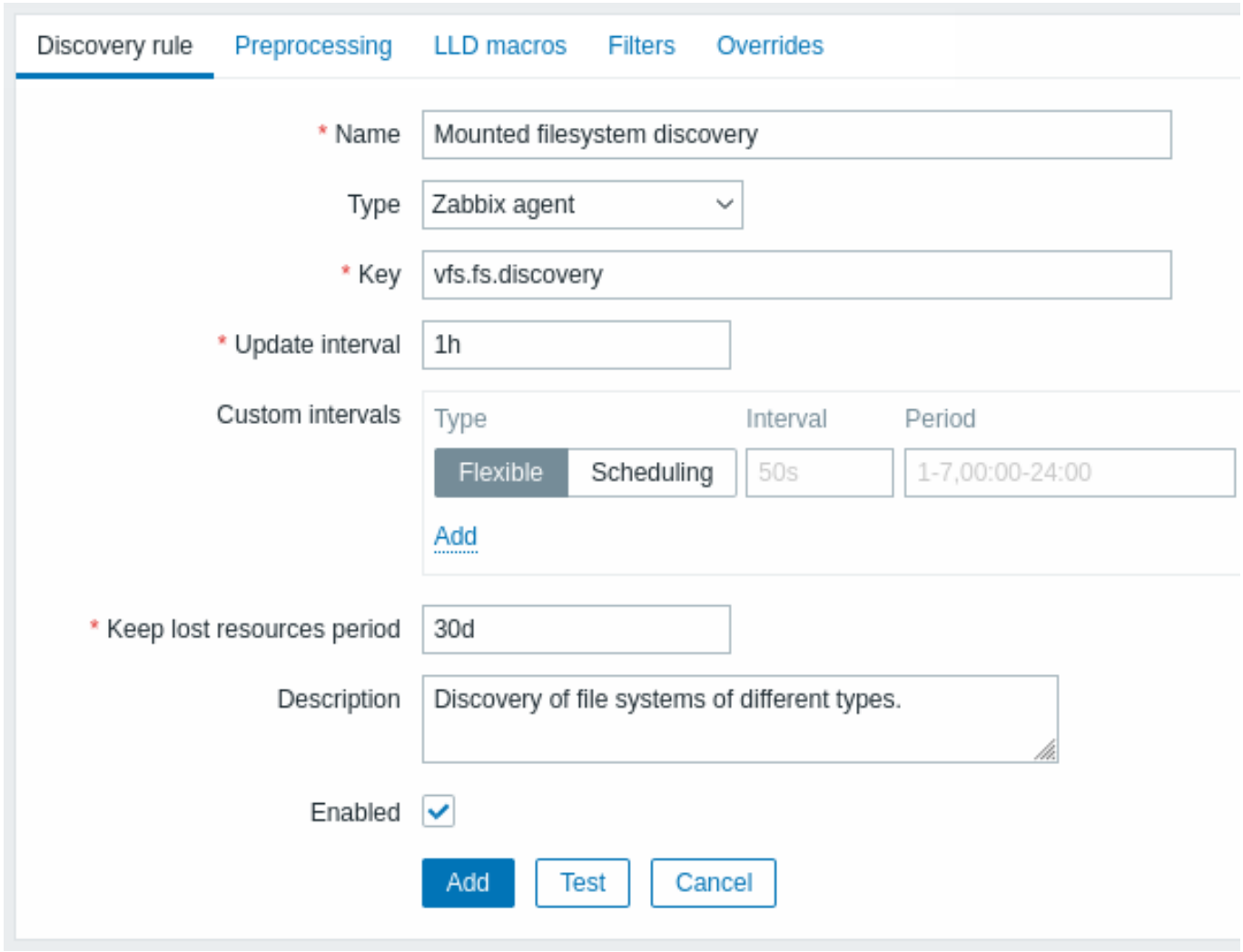
- Click on *Create discovery rule* in the upper right corner of the screen
- Fill in the discovery rule form with the required details

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.

Parameter	Description
Name	Name of discovery rule.
Type	The type of check to perform discovery. In this example we are using a *Zabbix agent* item key. The discovery rule can also be a dependent item, depending on a regular item. It cannot depend on another discovery rule. For a dependent item, select the respective type (*Dependent item*) and specify the master item in the 'Master item' field. The master item must exist.
Key	Enter the discovery item key (up to 2048 characters). For example, you may use the built-in "vfs.fs.discovery" item key to return a JSON with the list of file systems present on the computer and their types. Note that another option for filesystem discovery is using discovery results by the "vfs.fs.get" agent key, supported since Zabbix 4.4.5 (see example).

Parameter	Description
Update interval	This field specifies how often Zabbix performs discovery. In the beginning, when you are just setting up file system discovery, you might wish to set it to a small interval, but once you know it works you can set it to 30 minutes or more, because file systems usually do not change very often. Time suffixes are supported, e.g. 30s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. *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. *Note* that for an existing discovery rule the discovery can be performed immediately by pushing the *Check now* button.
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.

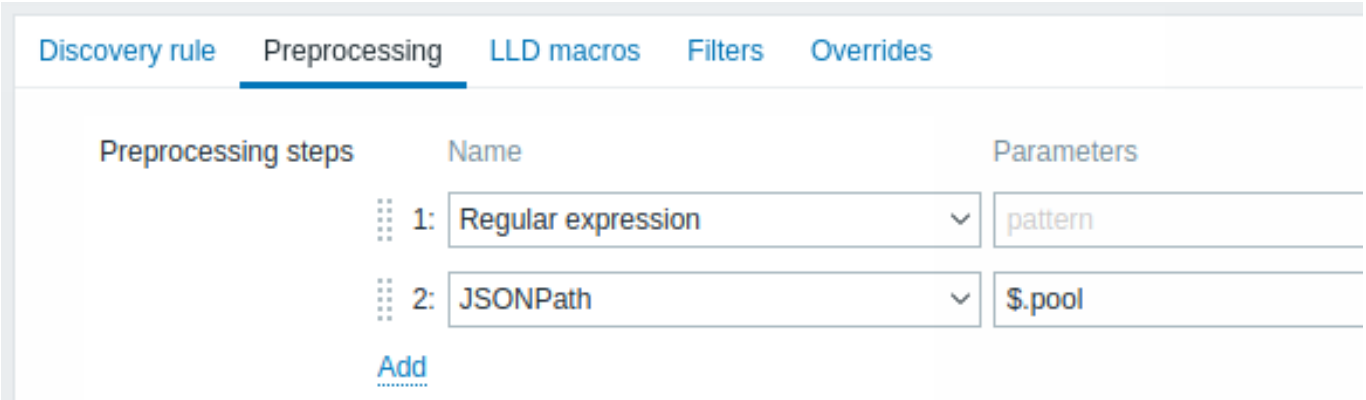
Note:
Discovery rule history is not preserved.

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



Type	Transformation	Description
Text		
	Regular expression	Match the received value to the <pattern> regular expression and replace value with the extracted <output>. The regular expression supports extraction of maximum 10 captured groups with the \N sequence. 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. 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.
	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.

Type	Transformation	Description

	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></do`
		`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></do`
		Note that namespaces are not supported.
		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.

	CSV to JSON	Convert CSV file data into JSON format. For more information, see: CSV to JSON preprocessing. Supported since 4.4.0.
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	Transformation	Description
	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.

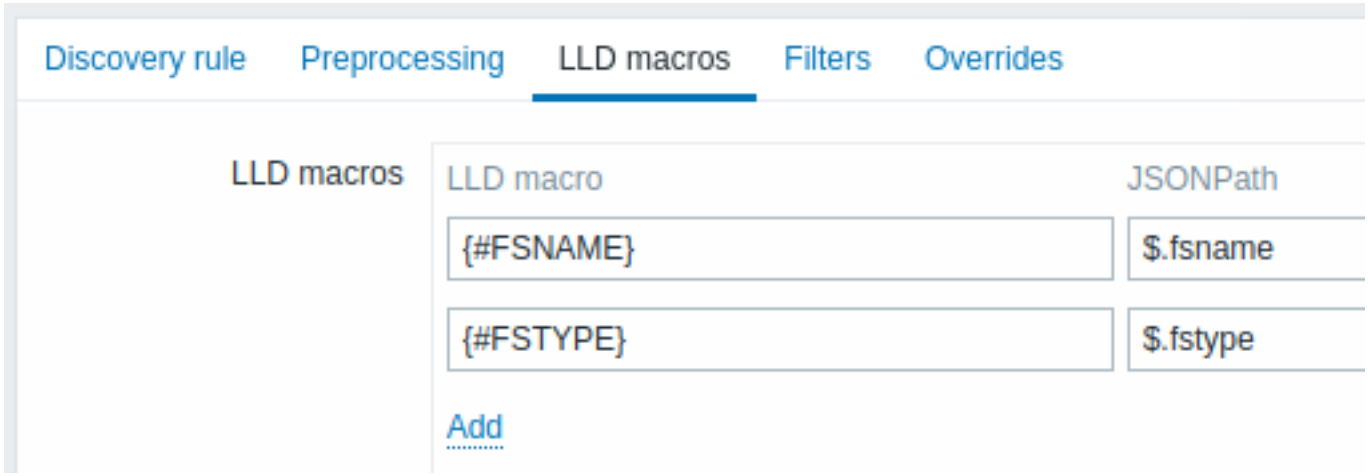
Note that if the discovery rule has been applied to the host via template then the content of this tab is read-only.

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:

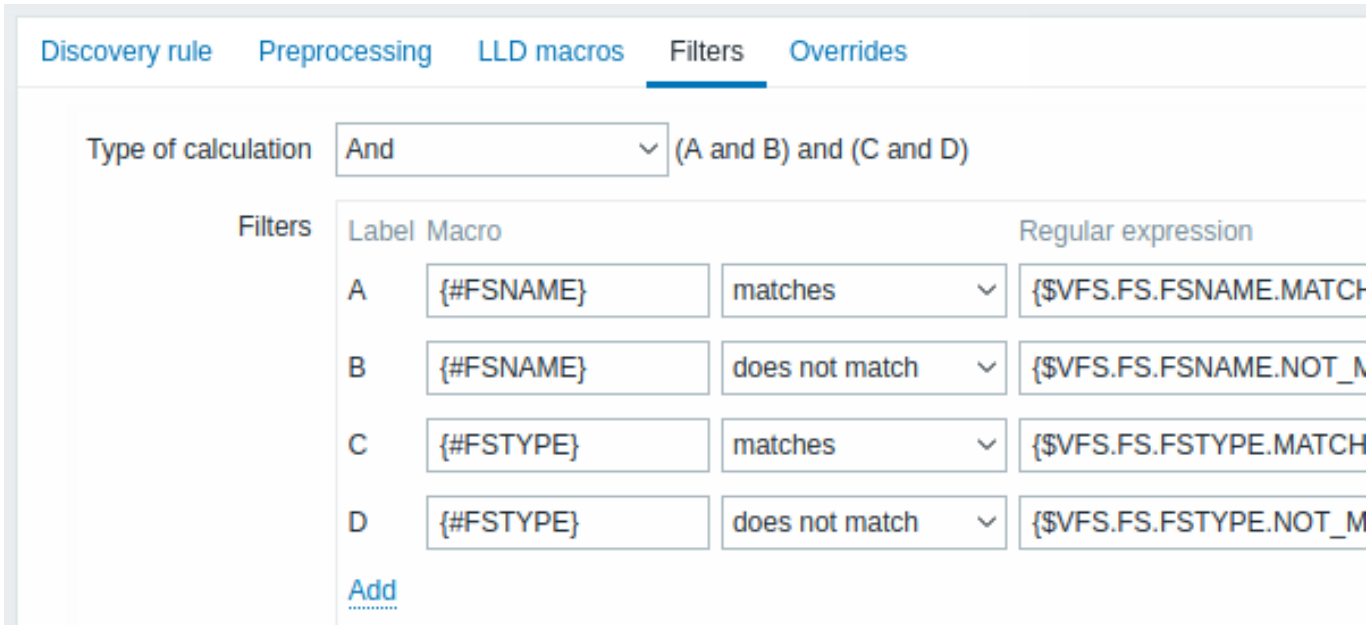


The extracted values can be used in discovered items, triggers, etc. Note that values will be extracted from the result of discovery and any preprocessing steps so far.

Parameter	Description
LLD macro	Name of the low-level discovery macro, using the following syntax: {#MACRO}.
JSONPath	Path that is used to extract LLD macro value from a LLD row, using JSONPath syntax. 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 `$['unicode + special chars #1']['unicode + special chars #2']`.

Filter

The **Filters** tab contains discovery rule filter definitions allowing to filter discovery values:



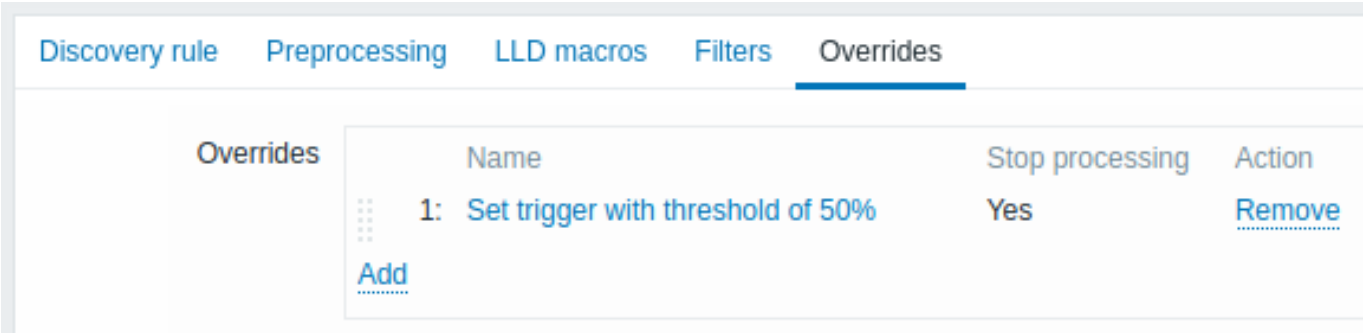
Parameter	Description
Type of calculation	The following options for calculating filters are available: **And** - all filters must be passed; **Or** - enough if one filter is passed; **And/Or** - uses *And* with different macro names and *Or* with the same macro name; **Custom expression** - offers the possibility to define a custom calculation of filters. The formula must include all filters in the list. Limited to 255 symbols.
Filters	A filter can be used to generate real items, triggers, and graphs only for certain file systems. It expects 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` macro on Windows is supported since Zabbix **3.0.0**. Defining several filters is supported since Zabbix **2.4.0**. Note that if some macro from the filter is missing in the response, the found entity will be ignored. Filter drop-down offers two values to specify whether a macro matches a regular expression or does not match.

Warning:
A mistake or typo in the regular expression used in LLD rule may cause deleting thousands of configuration elements, historical values and events for many hosts. For example, an incorrect "File systems for discovery" regular expression may cause deleting thousands of items, triggers, historical values and events.

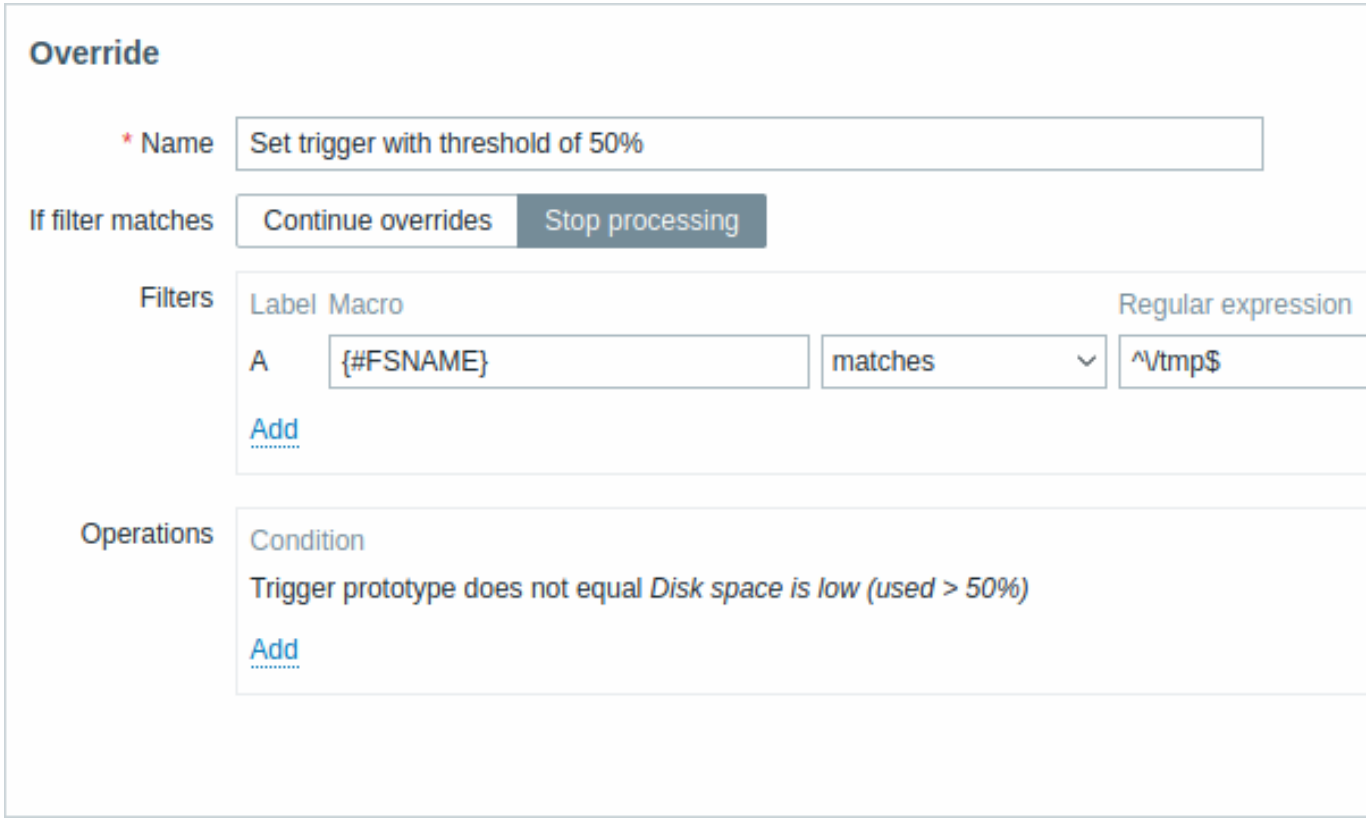
Attention:
Zabbix database in MySQL must be created as case-sensitive if file system names that differ only by case are to be discovered correctly.

Override

The **Override** tab allows to set 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 Add 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.

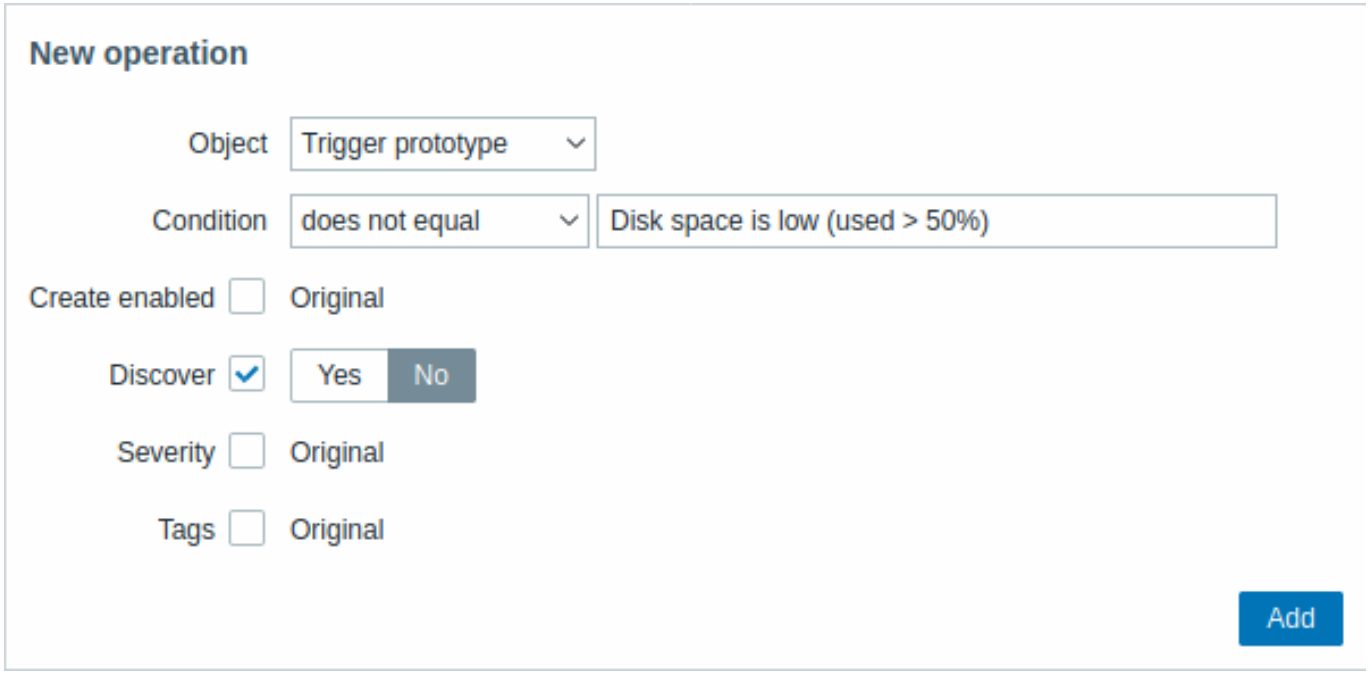


All mandatory parameters are marked with red asterisks.

Parameter	Description
Name	A unique (per LLD rule) override name.
If filter matches	Defines whether subsequent 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 Add in the Operations block. To edit an existing operation, click on Edit next to the operation. A popup window where you can edit the operation details will open.



Parameter		Description
Object		Four types of objects are available: Item prototype Trigger prototype Graph prototype Host prototype
Condition		Allows to filter entities to which the operation should be applied.
	Operator	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
	Pattern	A regular expression or a string to search for.
	Object: *Item prototype*	
	Create enabled	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.
	Discover	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.

Parameter		Description
	Update interval	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 Add to specify flexible/scheduling intervals. For detailed information see Custom intervals.
	History storage period	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.
	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.
	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.

Parameter		Description
	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.
	Host inventory	When the checkbox is marked, the buttons will appear, allowing to select different inventory mode for the host prototype: *Disabled* - do not populate host inventory *Manual* - provide details manually *Automated* - auto-fill host inventory data based on collected metrics.

Form buttons

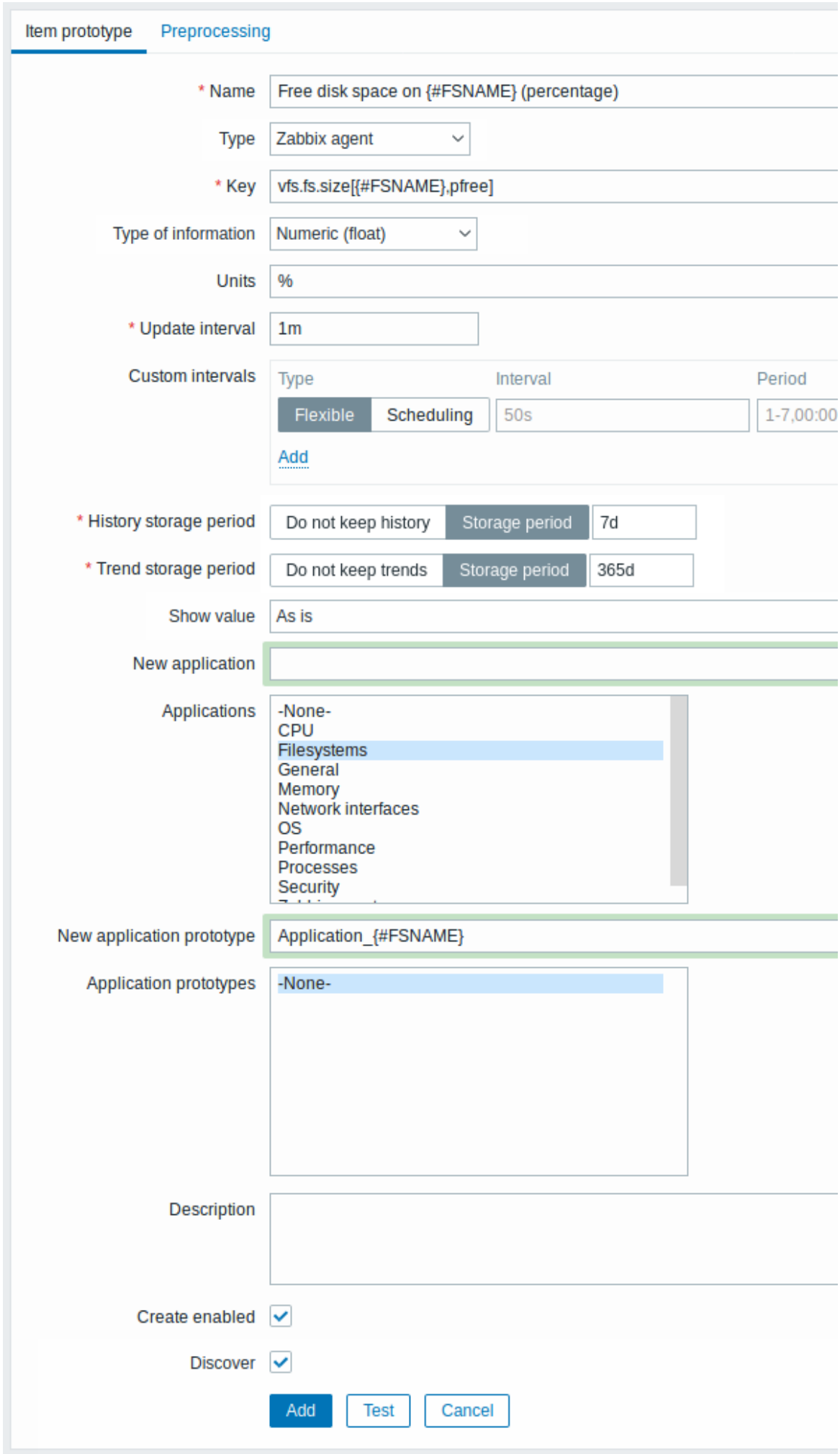
Buttons at the bottom of the form allow to perform several operations.

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.

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 the {#FSNAME} macro is used where a file system name is required. The use of a low-level discovery macro is mandatory in the item key to make sure that the discovery is processed correctly. When the discovery rule is processed, this macro will be substituted with the discovered file system.


Item prototype
Preprocessing
* Name
Free disk space on {#FSNAME} (percentage)
Type
Zabbix agent
* Key
vfs.fs.size[{#FSNAME},pfree]
Type of information
Numeric (float)
Units
%
* Update interval
1m
Custom intervals
Type
Interval
Period
Flexible
Scheduling
50s
1-7,00:00
Add
* History storage period
Do not keep history
Storage period
7d
* Trend storage period
Do not keep trends
Storage period
365d
Show value
As is
New application
Applications
-None-
CPU
Filesystems
General
Memory
Network interfaces
OS
Performance
Processes
Security
New application prototype
Application_{#FSNAME}
Application prototypes
-None-
Description
Create enabled
Discover
Add
Test
Cancel

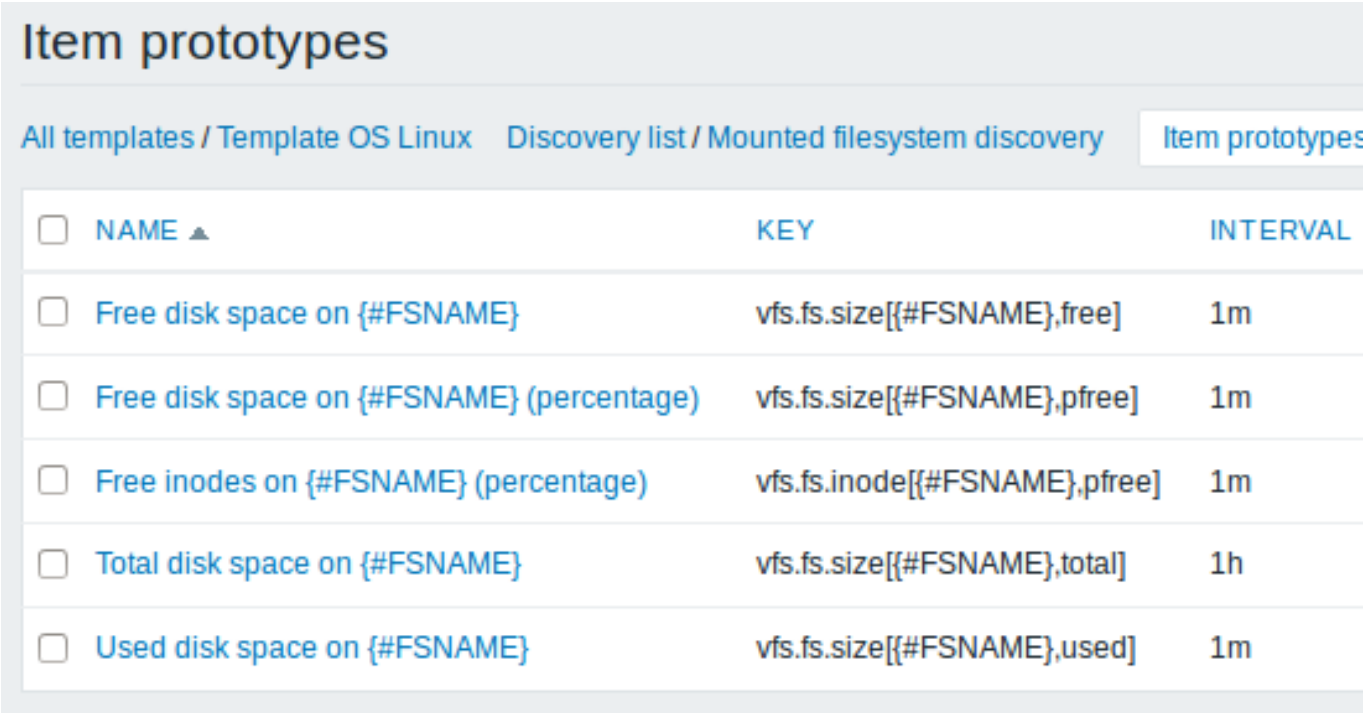
Low-level discovery macros and user macros are supported in item prototype configuration and item value preprocessing parameters. Note that when used in update intervals, a single macro has to fill the whole field. Multiple macros in one field or macros mixed with text are not supported.

Note:
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:

Parameter	Description
New application prototype	You may define a new application prototype. In application prototypes you can use low-level discovery macros that, after discovery, will be substituted with real values to create applications that are specific for the discovered entity. See also application discovery notes for more specific information.
Application prototypes	Select from the existing application prototypes.
Create enabled	If checked the item will be added in an enabled state. If unchecked, the item will be added to a discovered entity, but in a disabled state.
Discover	If checked (default) the item will be added to a discovered entity. If unchecked, the item will not be added to a discovered entity, unless this setting is overridden in the discovery rule.

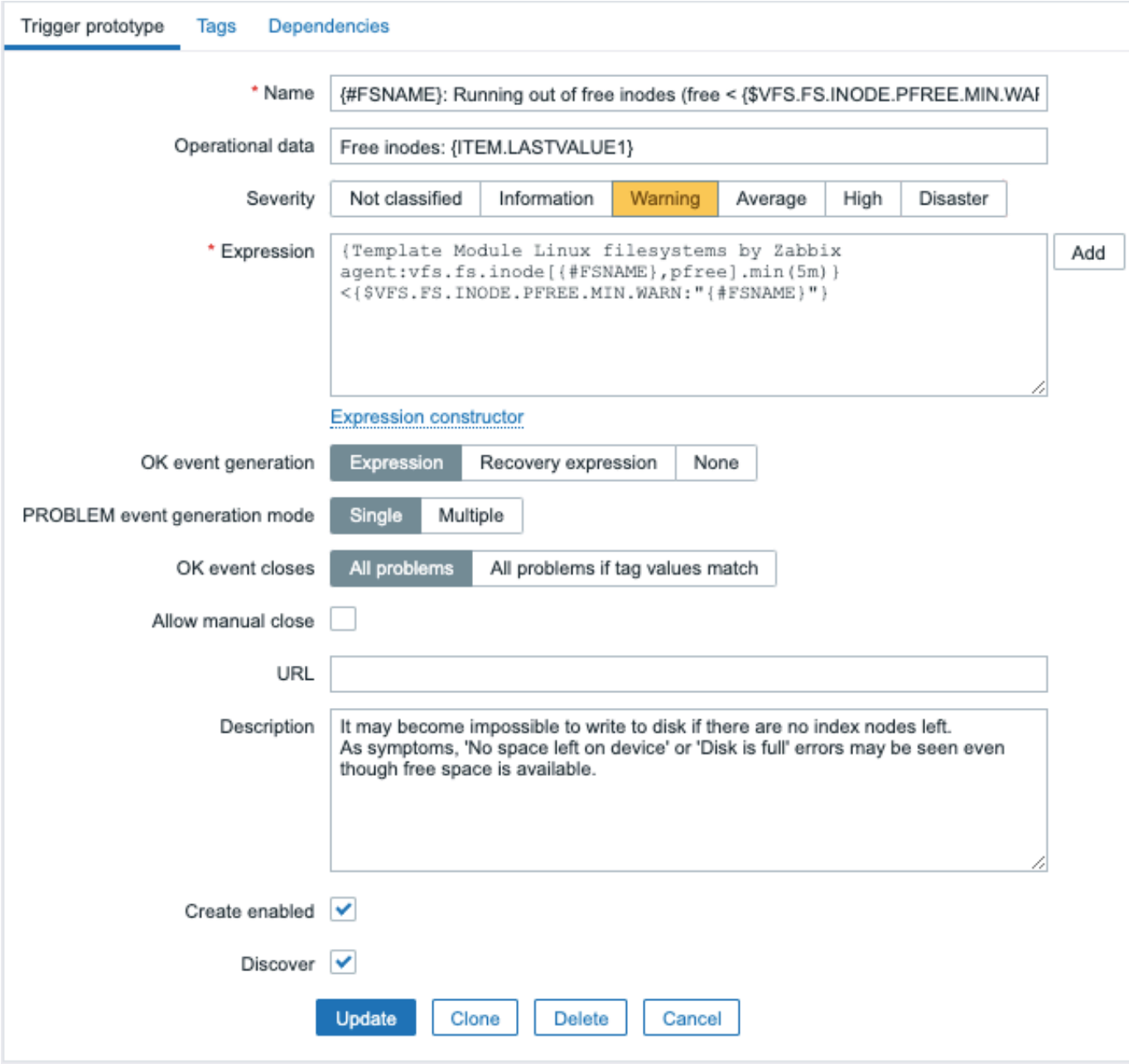
We can create several item prototypes for each file system metric we are interested in:



Mass update option is available if you want to update properties of several item prototypes at once.

Trigger prototypes

We create trigger prototypes in a similar way as item prototypes:

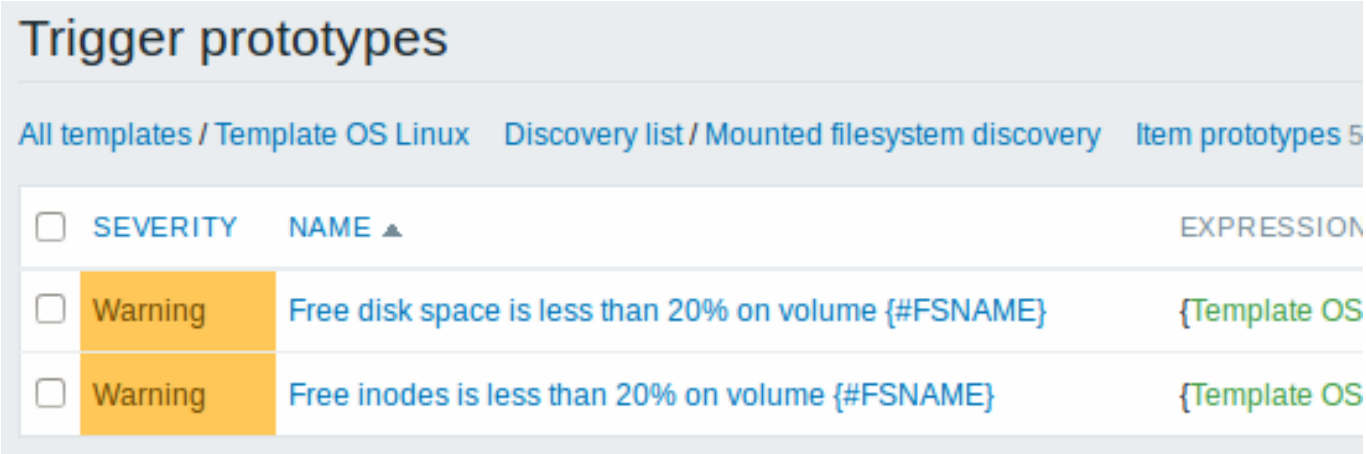


Attributes that are specific for trigger prototypes:

Parameter	Description
Create enabled	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.
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.

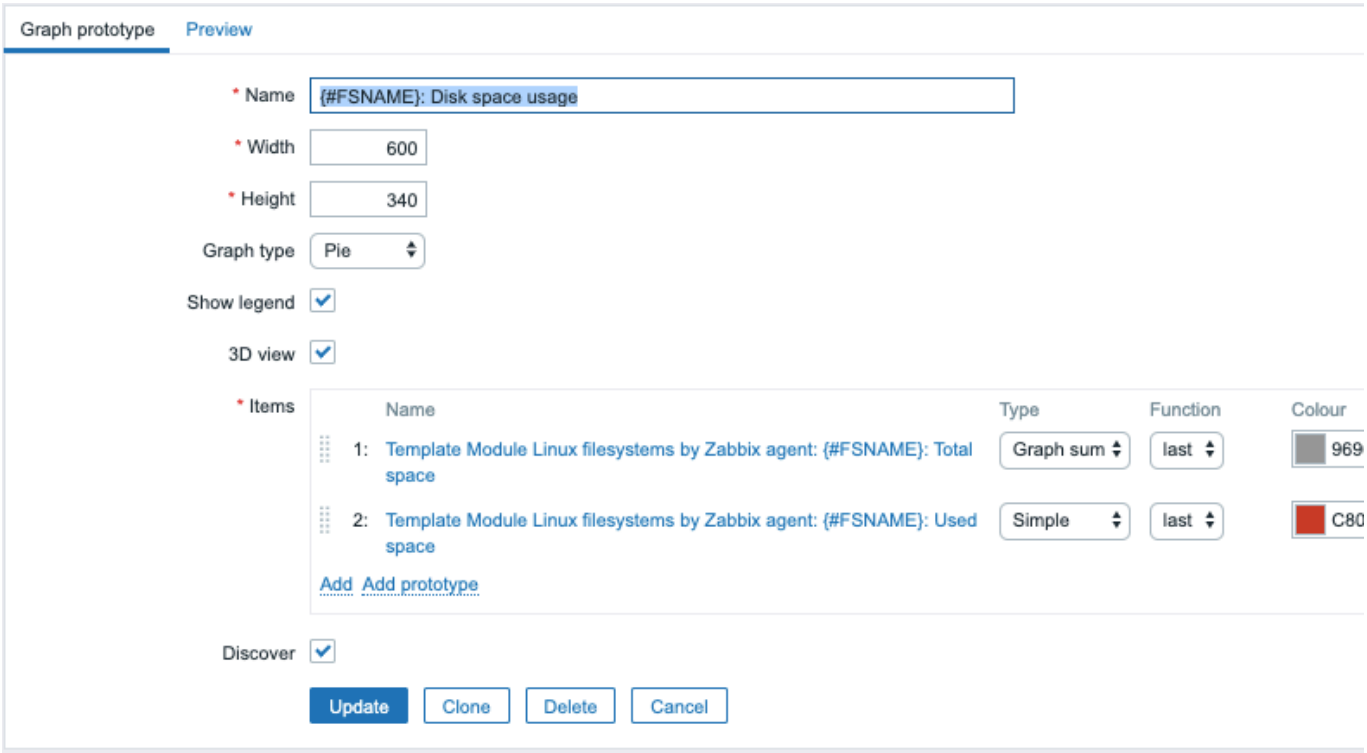
When real triggers are created from the prototypes, there may be a need to be flexible as to what constant ('20' in our example) is used for comparison in the expression. See how user macros with context can be useful to accomplish such flexibility.

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.



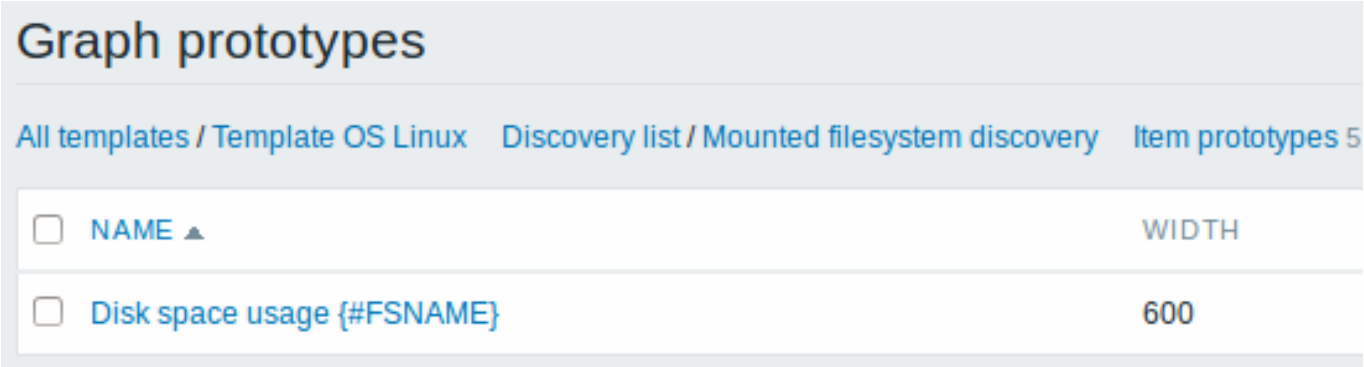
Graph prototypes

We can create graph prototypes, too:

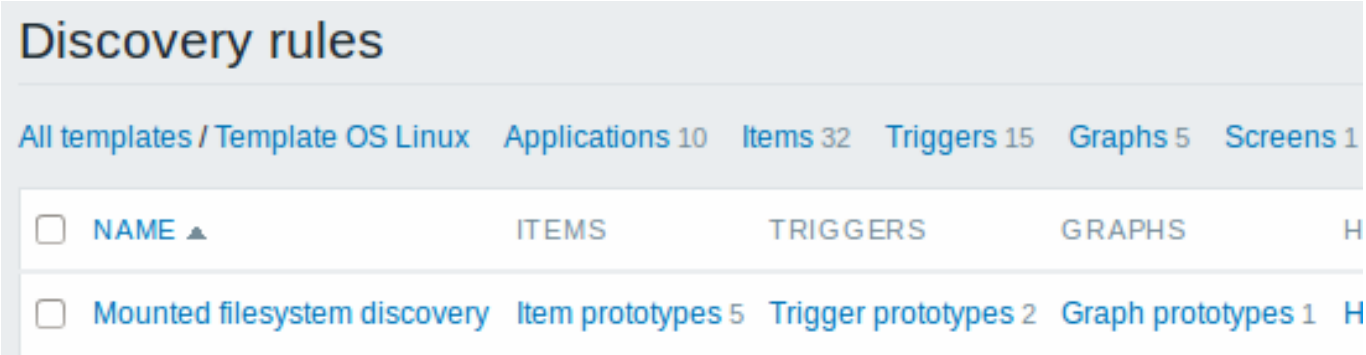


Attributes that are specific for graph prototypes:

Parameter	Description
Discover	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.



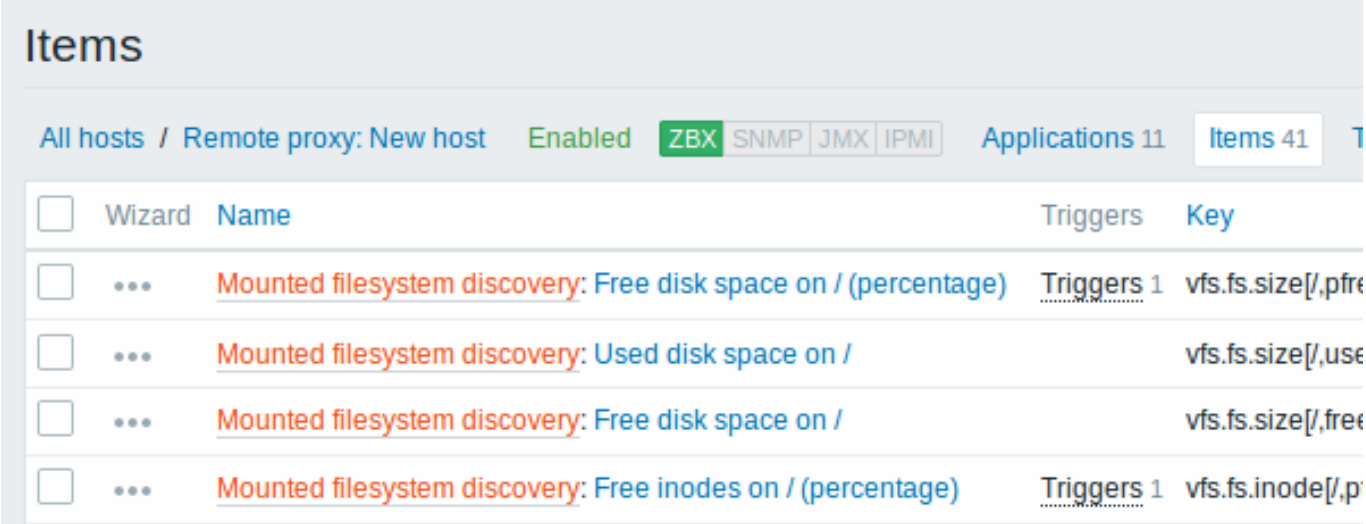
Finally, we have created a discovery rule that looks as shown below. It has five item prototypes, two trigger prototypes, and one graph prototype.



Note: For configuring host prototypes, see the section about host prototype configuration in virtual machine monitoring.

Discovered entities

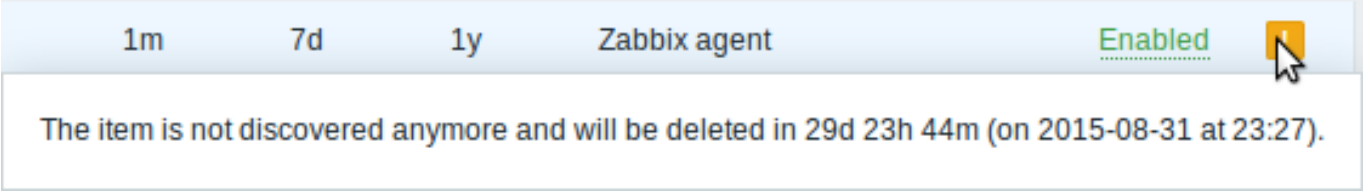
The screenshots below illustrate how discovered items, triggers, and graphs look like in the host's configuration. Discovered entities are prefixed with an orange link to a discovery rule they come from.



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.

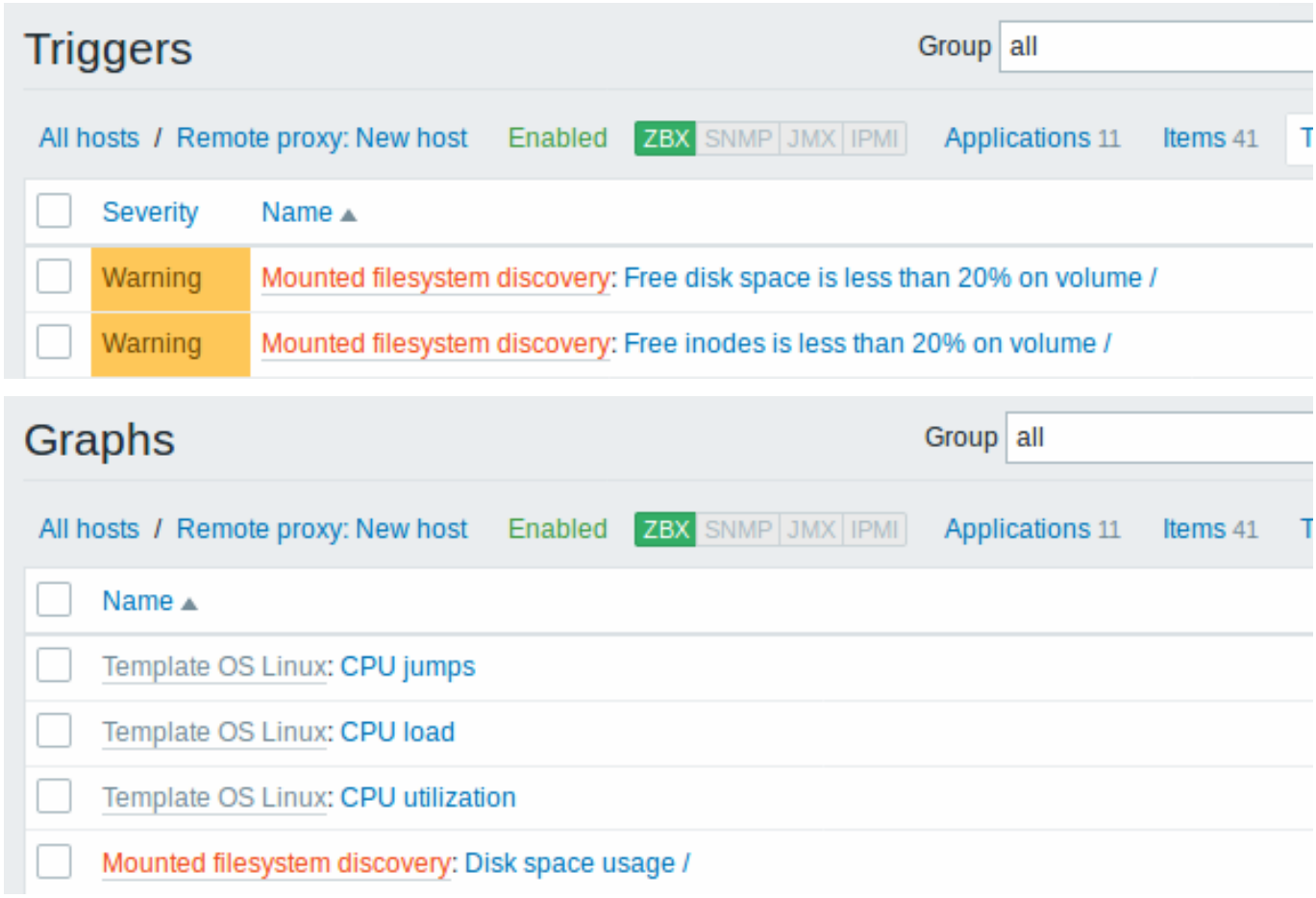
Items (similarly, triggers and graphs) created by a low-level discovery rule will be deleted automatically if a discovered entity (file system, interface, etc) stops being discovered (or does not pass the filter anymore). In this case the items, triggers and graphs will be deleted after the days defined 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.

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 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.

Multiple LLD rules for 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.

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 file systems and outputs JSON, which includes both file system name and type. One way to use it would be as a UserParameter with key "vfs.fs.discovery_perl":

```
###!/usr/bin/perl

$first = 1;

print "[\n";

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}\":\"$fsname\",\n";
    print "\t\t\"{#FSTYPE}\":\"$fstype\"\n";
    print "\t}\n";
}

print "]\n";
```

Attention:
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.

```
[
    { "{#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:

```
###!/usr/bin/perl

$first = 1;

print "[\n";

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\":\"$fsname\",\n";
    print "\t\t\"fstype\":\"$fstype\"\n";
    print "\t}\n";
}

print "]\n";
```

An example of its output (reformatted for clarity) is shown below. JSON for custom discovery checks has to follow the same format.

```
[
    { "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"  }
]
```

Then, in the discovery rule's "Filter" field, we could specify "{#FSTYPE}" as a macro and "rootfs|ext3" as a regular expression.

Note:
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.

Using LLD macros in user macro contexts

LLD macros may be used inside user macro context, for example, in trigger prototypes.

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:

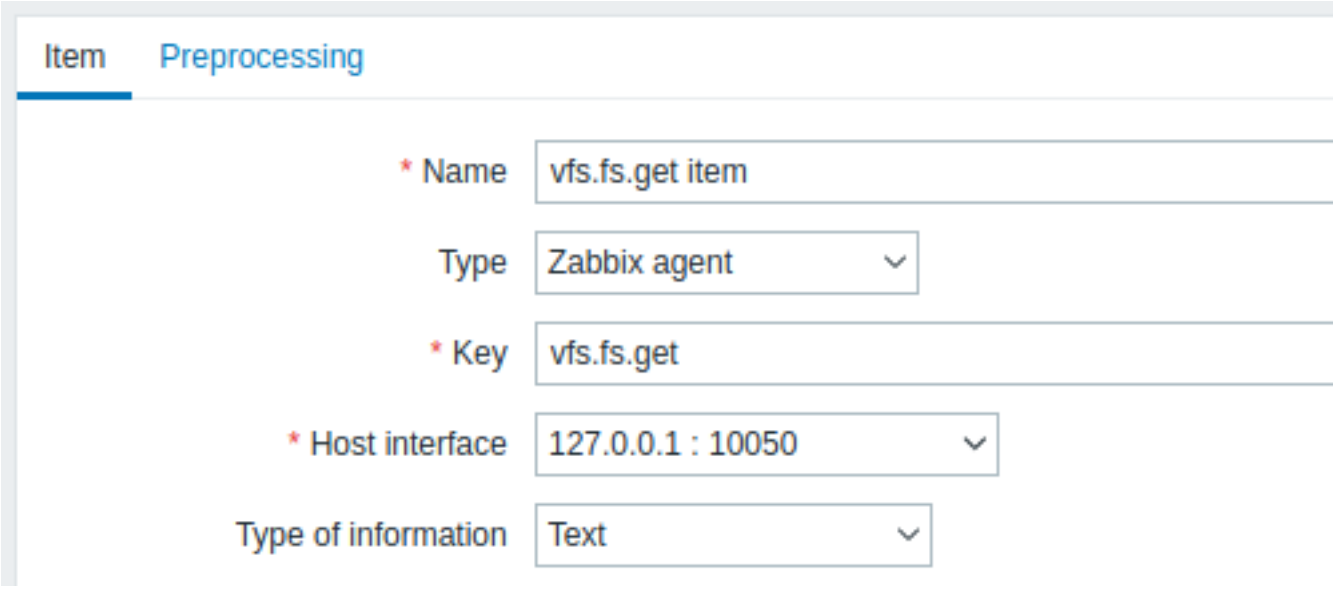
- the `vfs.fs.get` agent item as the master item
- dependent low-level discovery rule and item prototypes

Configuration

Master item

Create a Zabbix agent item using the following key:

`vfs.fs.get`



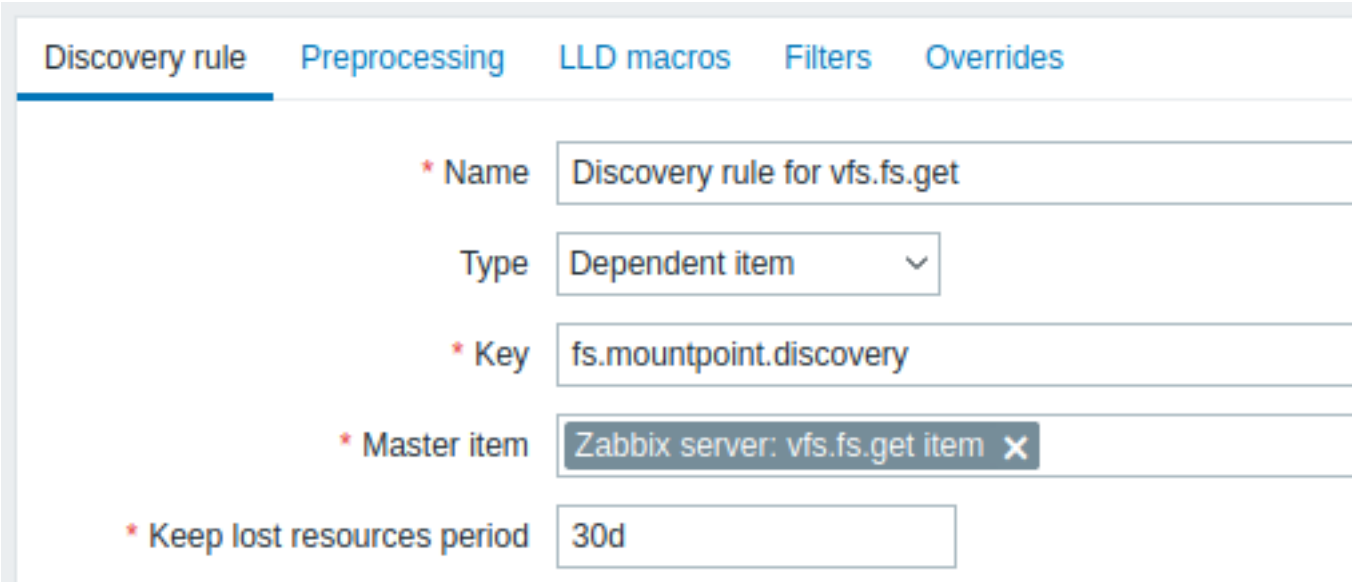
Set the type of information to ”Text” for possibly big JSON data.

The data returned by this item will contain something like the following for a mounted filesystem:

```
{
  "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
  }
}
```

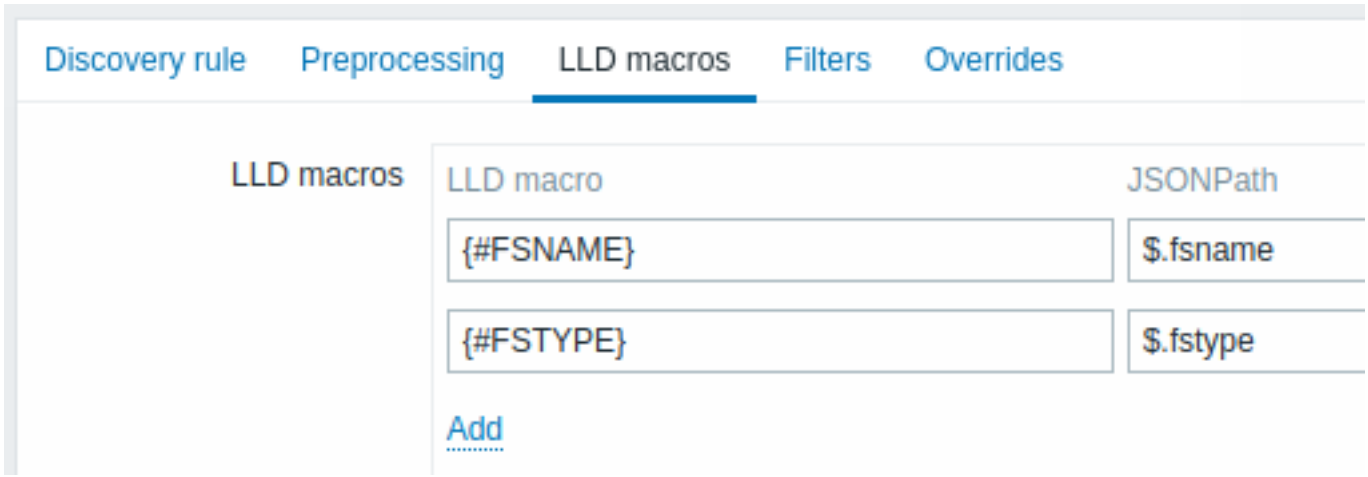
Dependent LLD rule

Create a low-level discovery rule as ”Dependent item” type:



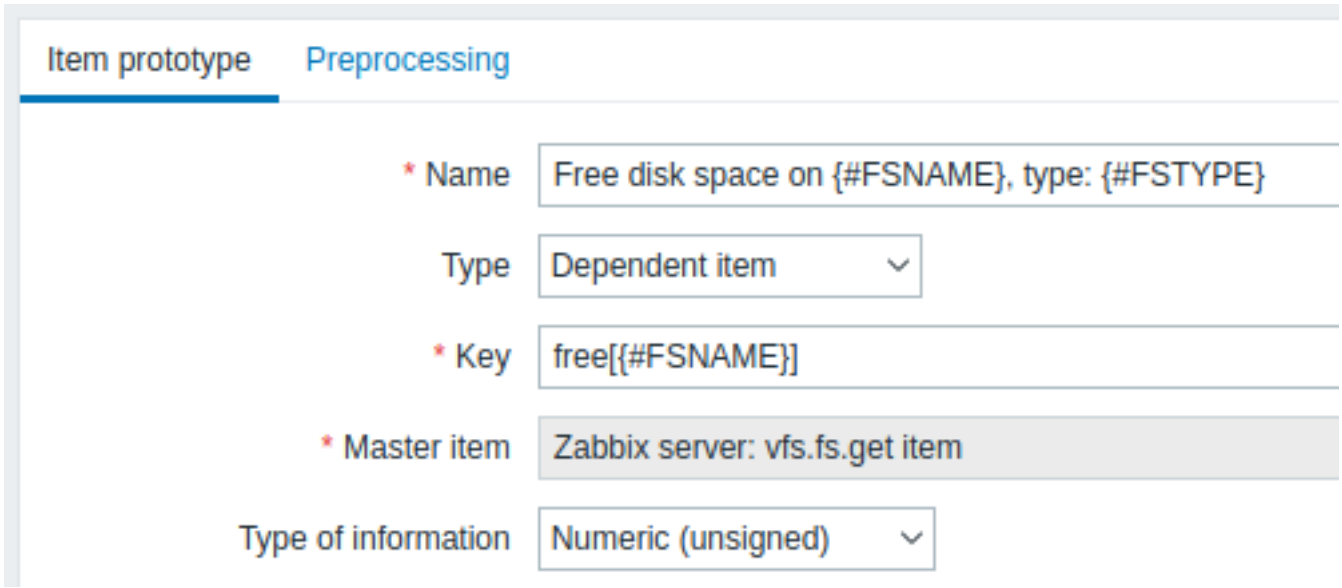
As master item select the `vfs.fs.get` item we created.

In the "LLD macros" tab define custom macros with the corresponding JSONPath:



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.



Note the use of custom macros in the item prototype name and key:

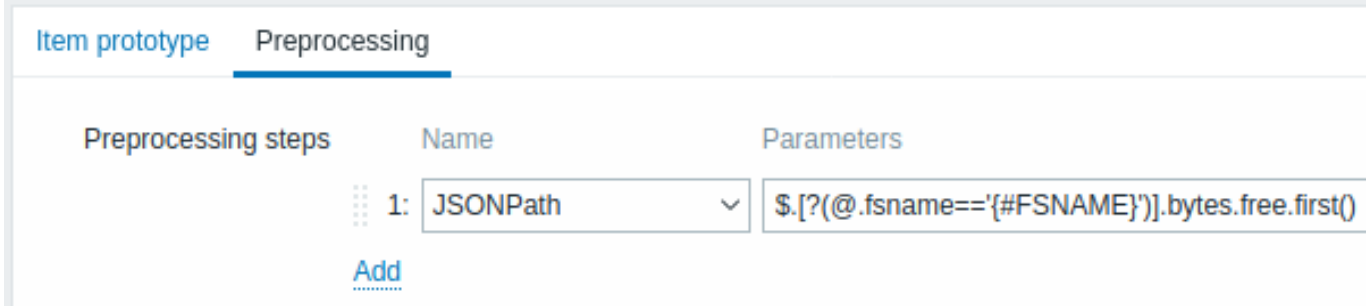
- *Name*: Free disk space on `{#FSNAME}`, type: `{#FSTYPE}`
- *Key*: Free`[{#FSNAME}]`

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 since Zabbix 5.0.16.

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:

- `system.cpu.util[{#CPU.NUMBER},<type>,<mode>]`
- `system.hw.cpu[{#CPU.NUMBER},<info>]`

For detailed item key description, see Zabbix agent item keys.

4 Discovery of SNMP OIDs

Overview

In this section we will perform an SNMP discovery on a switch.

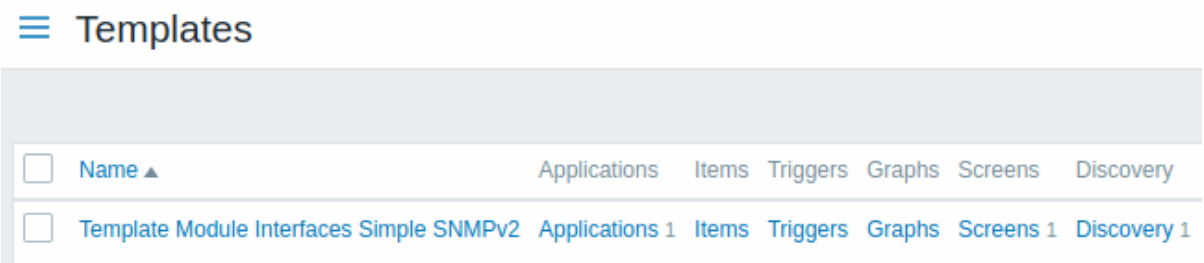
Item key

Unlike with file system and network interface discovery, the item does not necessarily has to have an "snmp.discovery" key - item type of SNMP agent is sufficient.

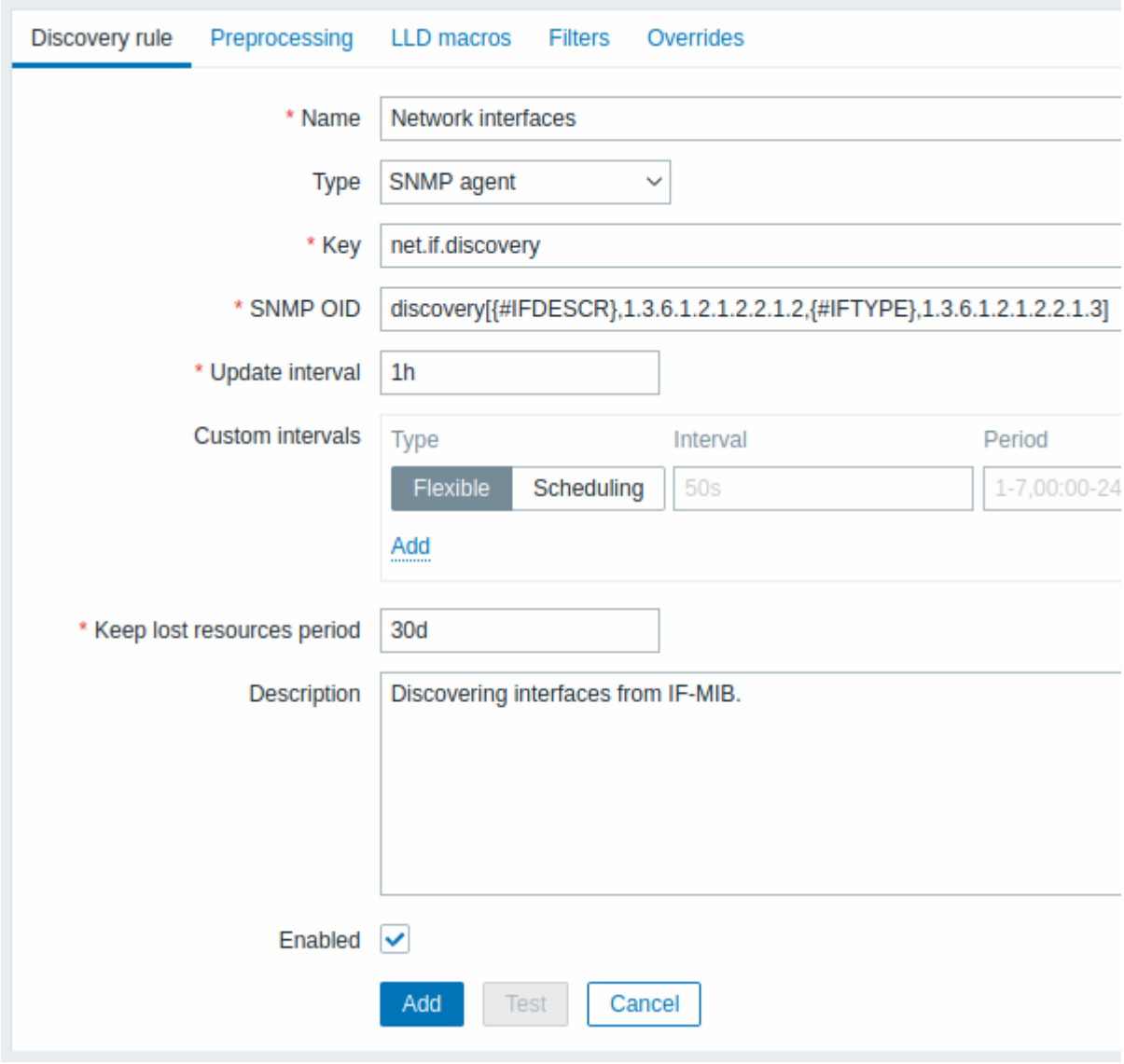
Discovery of SNMP OIDs is supported since Zabbix server/proxy 2.0.

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",
        "{#IFDESCR}": "WAN",
        "{#IFPHYSADDRESS}": "8:0:27:90:7a:75"
    },
    {
        "{#SNMPINDEX}": "2",
        "{#IFDESCR}": "LAN1",
        "{#IFPHYSADDRESS}": "8:0:27:90:7a:76"
    },
    {
        "{#SNMPINDEX}": "3",
        "{#IFDESCR}": "LAN2",
        "{#IFPHYSADDRESS}": "8:0:27:2b:af:9e"
    }
]
```

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:

```
ifDescr.1 "Interface #1"
ifDescr.2 "Interface #2"
ifDescr.4 "Interface #4"

ifAlias.1 "eth0"
ifAlias.2 "eth1"
ifAlias.3 "eth2"
ifAlias.5 "eth4"
```

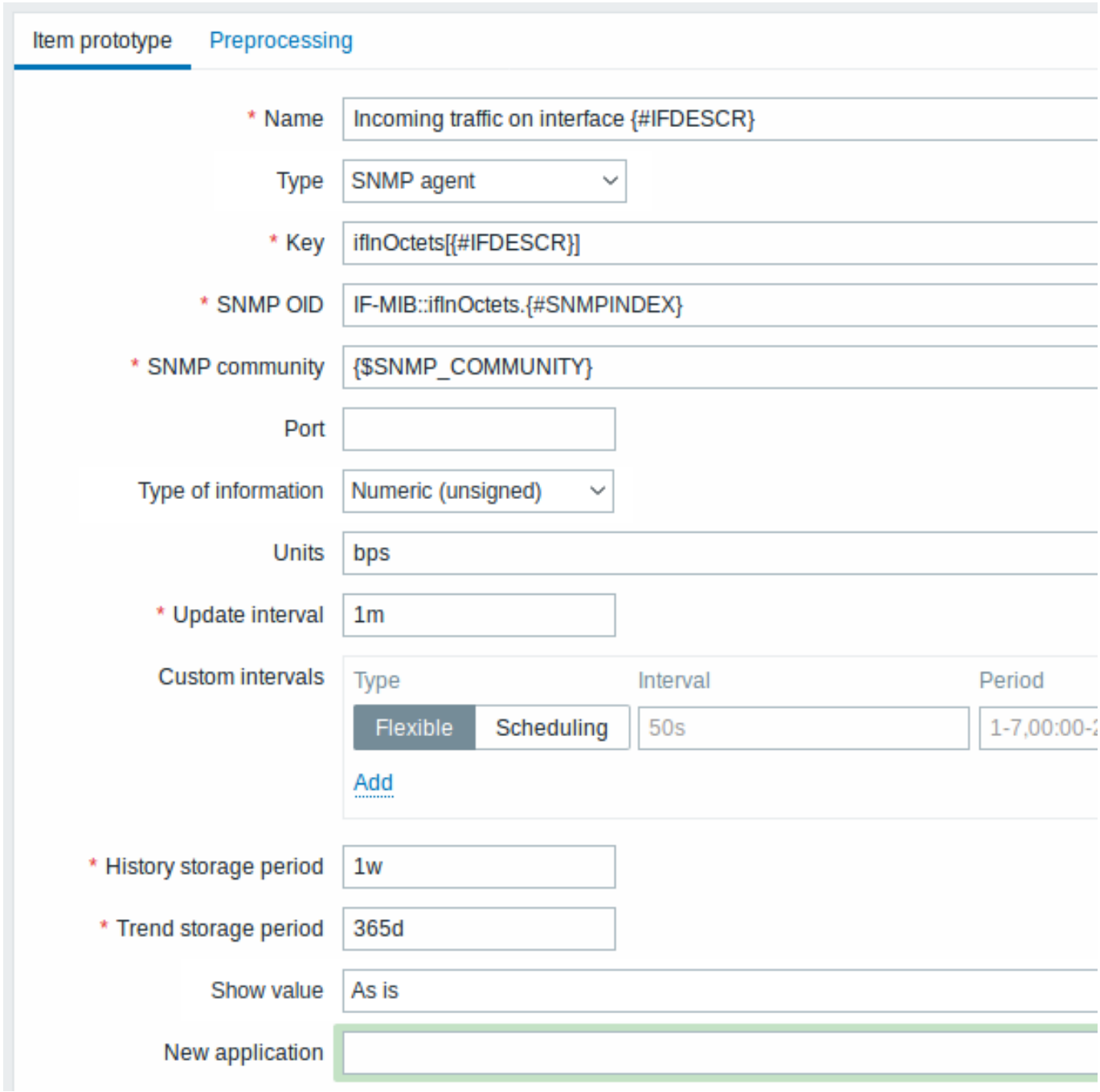
Then in this case SNMP discovery `discovery[{#IFDESCR}, ifDescr, {#IFALIAS}, ifAlias]` will return the following structure:

```
[
    {
        "{#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"
    }
```

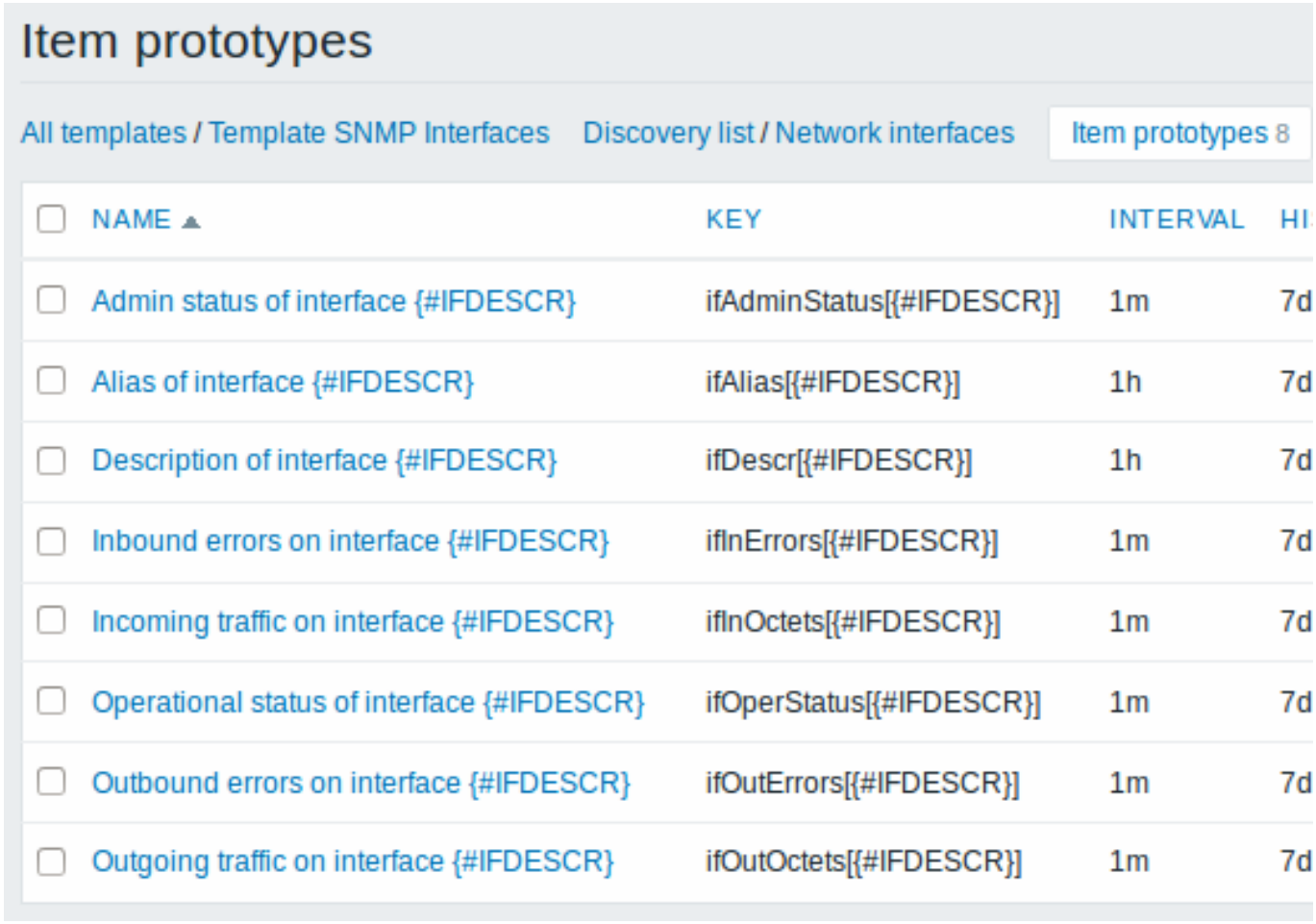
```
]
```

Item prototypes

The following screenshot illustrates how we can use these macros in item prototypes:

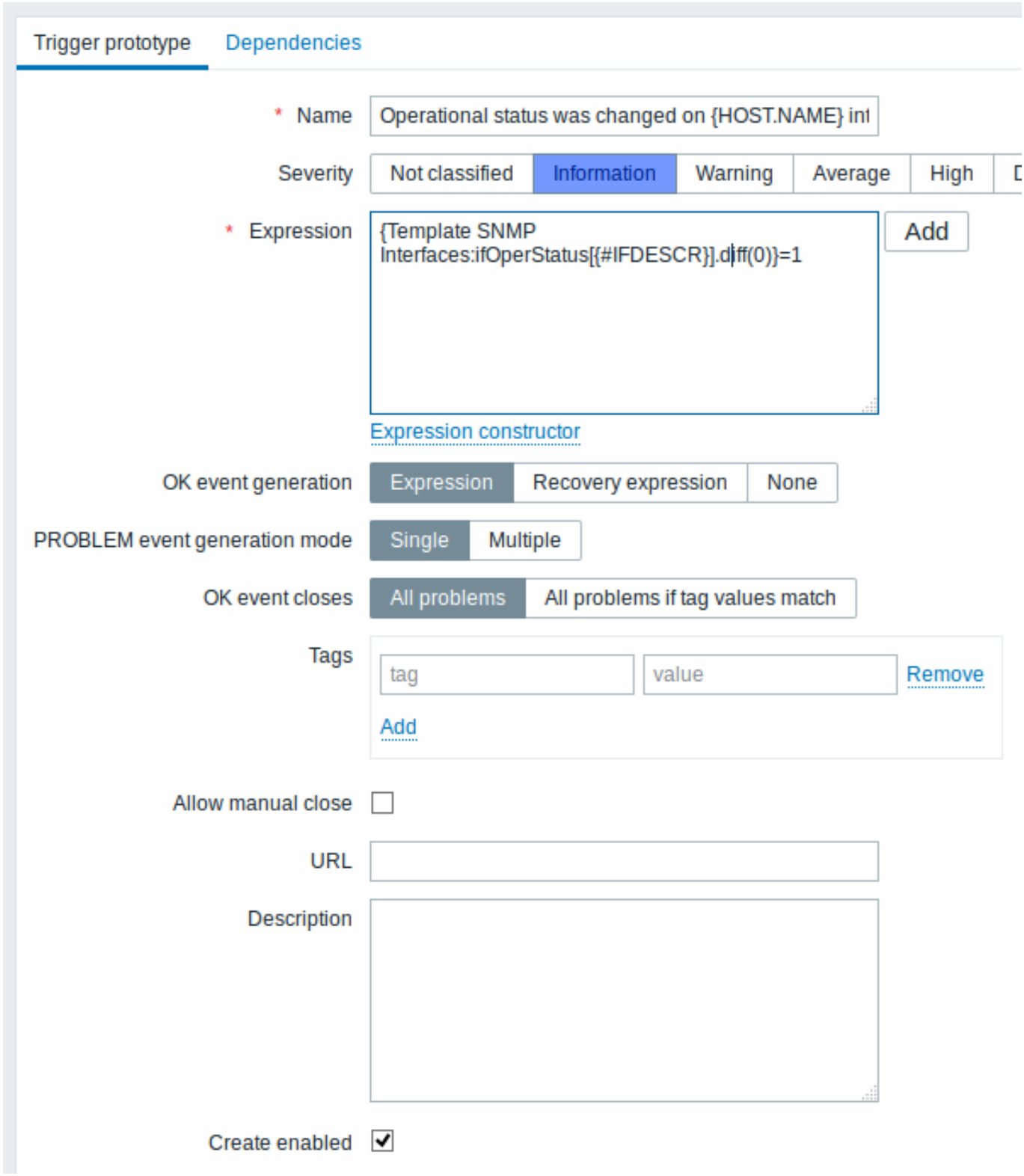


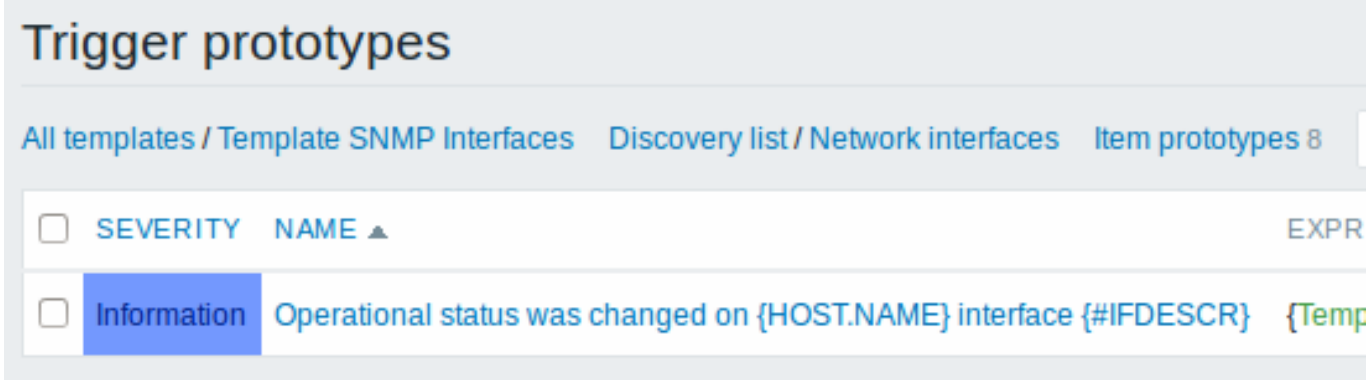
Again, creating as many item prototypes as needed:



Trigger prototypes

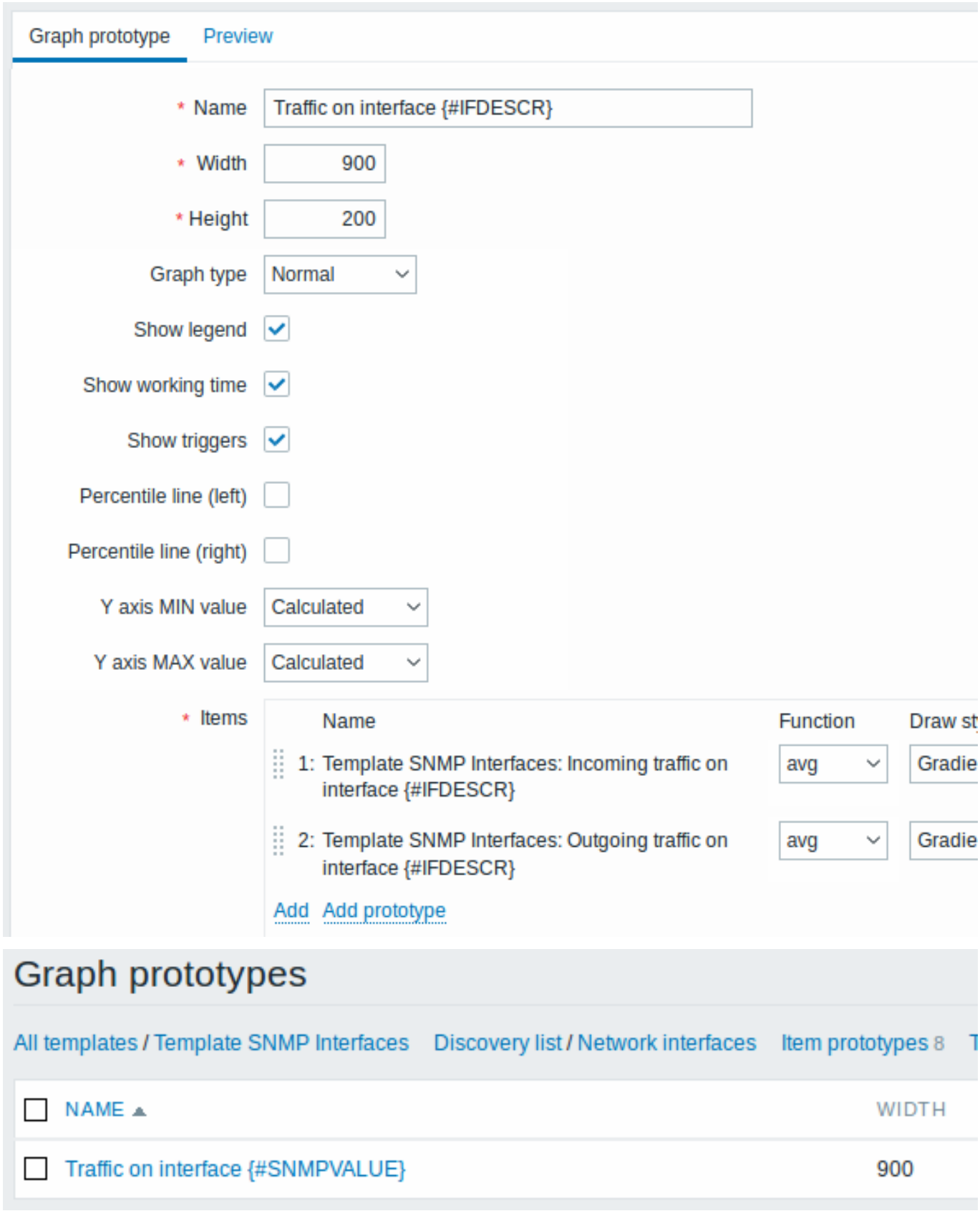
The following screenshot illustrates how we can use these macros in trigger prototypes:



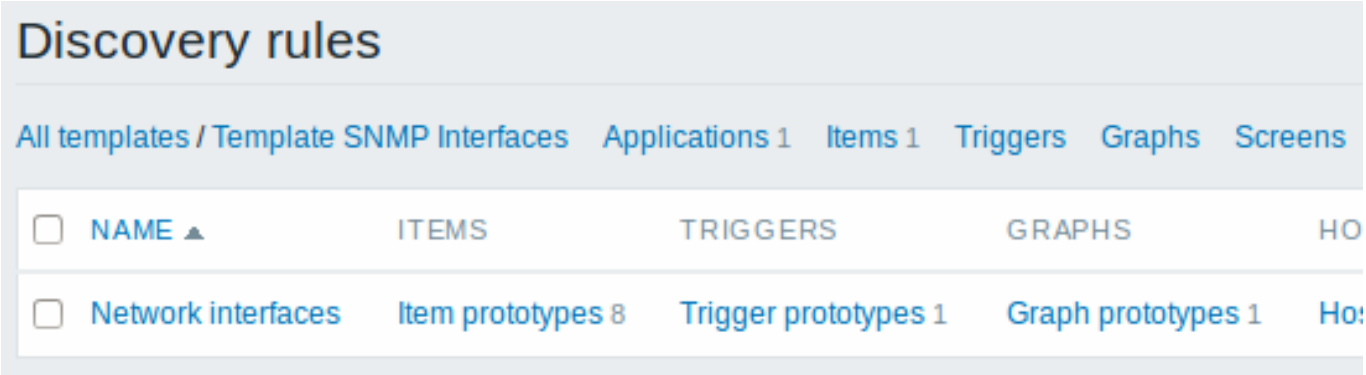


Graph prototypes

The following screenshot illustrates how we can use these macros in graph prototypes:

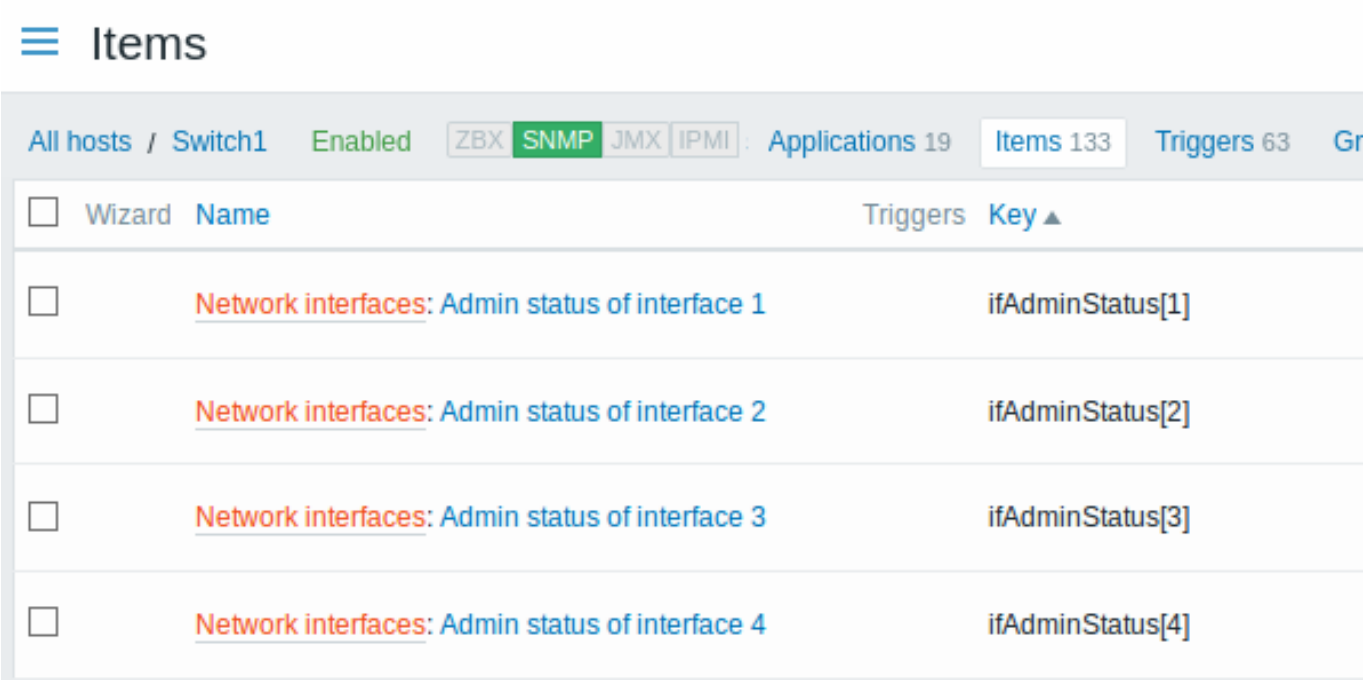


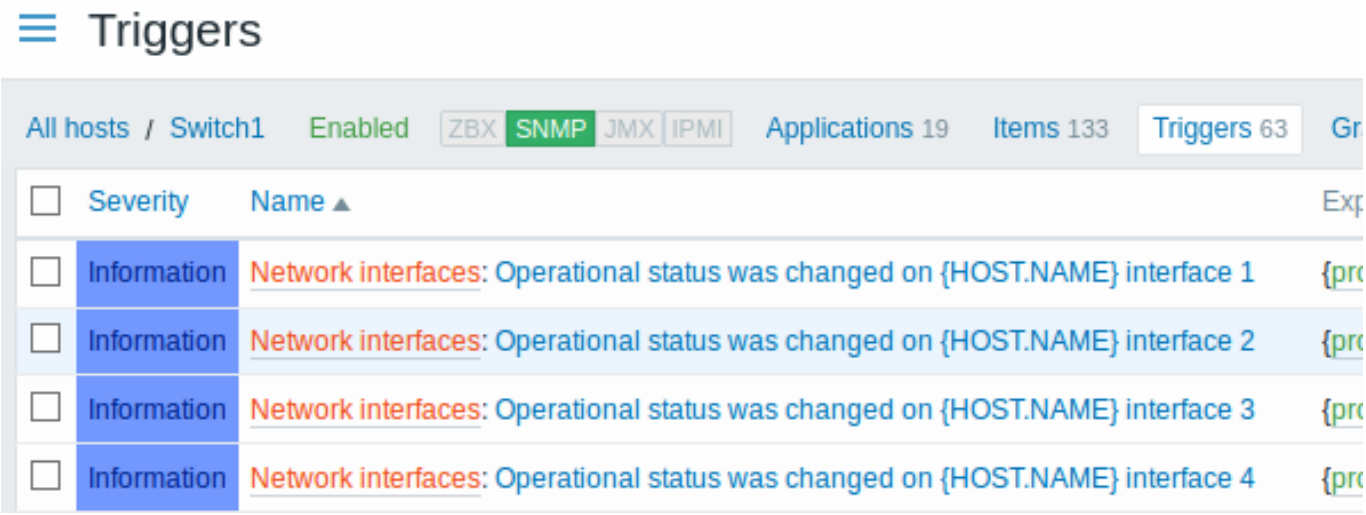
A summary of our discovery rule:

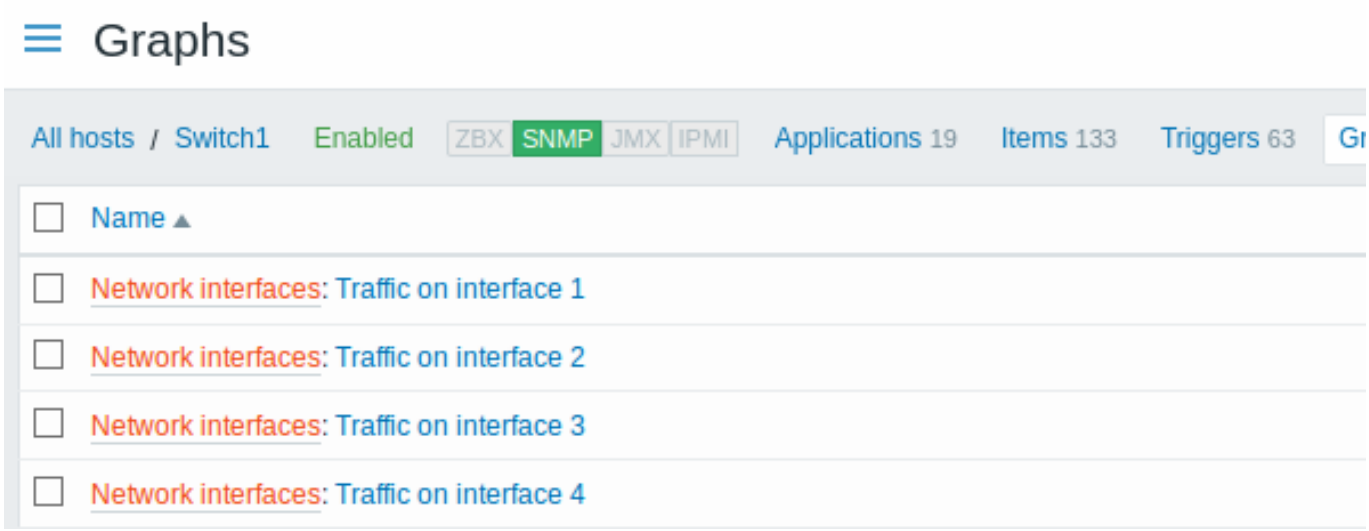


Discovered entities

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[]:

Item key	Return value	Parameters	Comment
jmx.discovery[<discovery mode>,<object name>]			

Item key			
	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 documentation) identifying the MBean names to be retrieved (empty by default, retrieving all registered beans)	Examples: → jmx.discovery - retrieve all JMX MBean attributes → jmx.discovery[beans] - retrieve all JMX MBeans → jmx.discovery[attribu - retrieve all garbage collector attributes → jmx.discovery[beans, - retrieve all garbage collectors 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 de-

Item key			
jmx.get[<discovery mode>,<object name>]	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* (retrieve JMX MBean attributes, default) or *beans* (retrieve JMX MBeans) **object name** - object name pattern (see documentation) identifying the MBean names to be retrieved (empty by default, retrieving all registered beans)	When using this item, it is needed to define custom low-level discovery macros, pointing to values extracted from the returned JSON using JSONPath. Supported since Zabbix Java gateway 4.4.

Attention:
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):

```
[
    {
        "{#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):

```
[
    {
        "{#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 (some of which will be ignored; see below):

```
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:

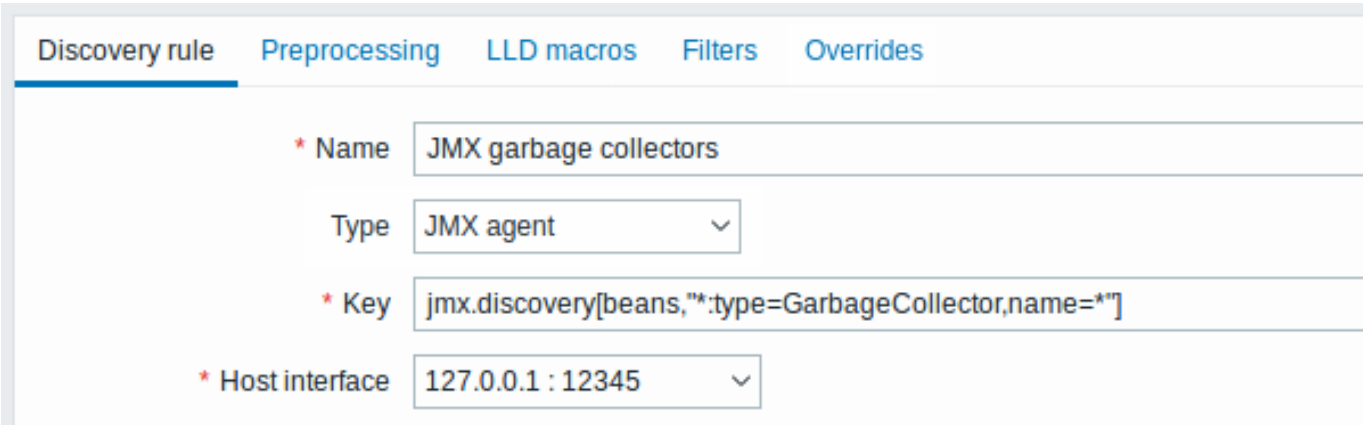
MBean1	**MBean2**	**MBean3**
MBean1Attribute1	MBean2Attribute1	MBean3Attribute1
MBean1Attribute2	MBean2Attribute2	MBean3Attribute2
MBean1Attribute3	MBean2Attribute3	MBean3Attribute3

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.

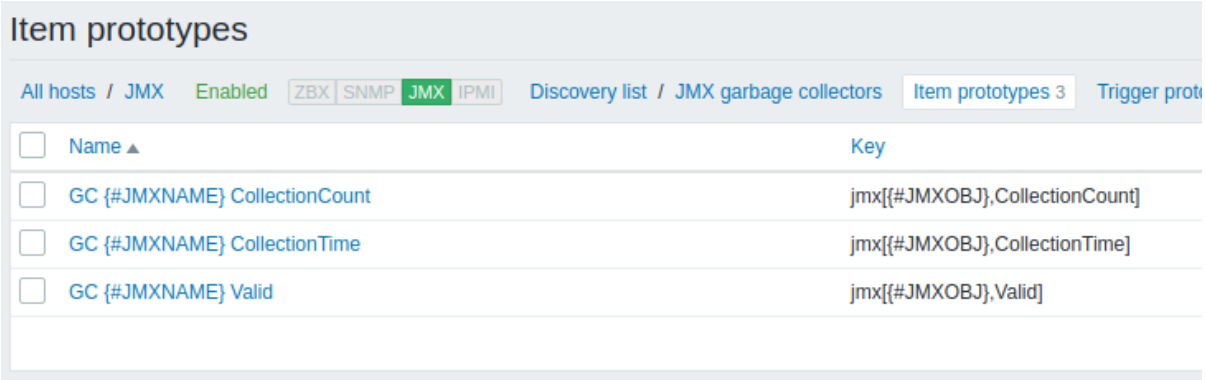
Discovery rule configuration collecting Mbeans (without the attributes) looks like the following:



Key used:

```
jmx.discovery[beans,"*:type=GarbageCollector,name=*"]
```

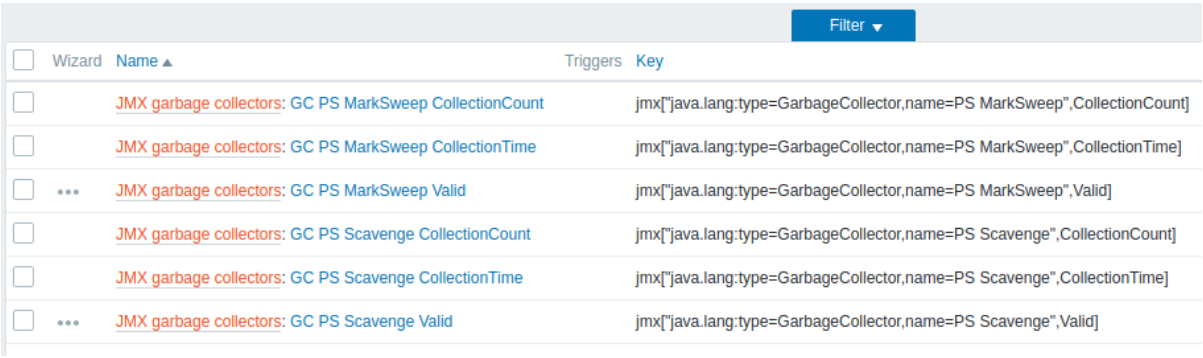
All the garbage collectors without attributes will be discovered. As Garbage collectors have the same attribute set, we can use desired attributes in item prototypes the following way:



Keys used:

```
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):

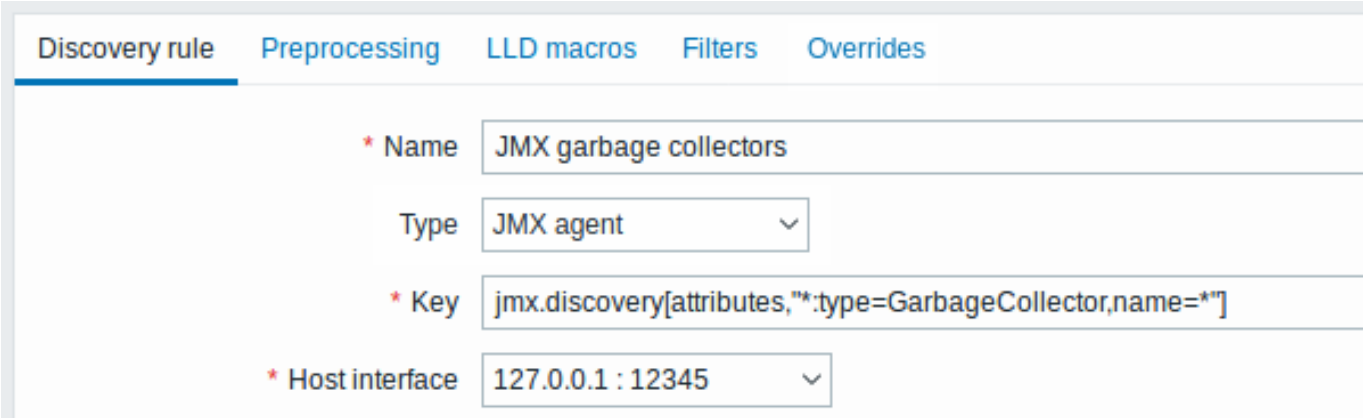


Example 2: Discovering Mbean attributes

This rule will return 9 objects with the following fields: MBean1Attribute1, MBean2Attribute1, Mbean3Attribute1,MBean1Attribute2,MBean2Attr
Mbean3Attribute2, MBean1Attribute3, MBean2Attribute3, Mbean3Attribute3.

For more information about objects please refer to supported macros table, *Discovery of MBean attributes* section.

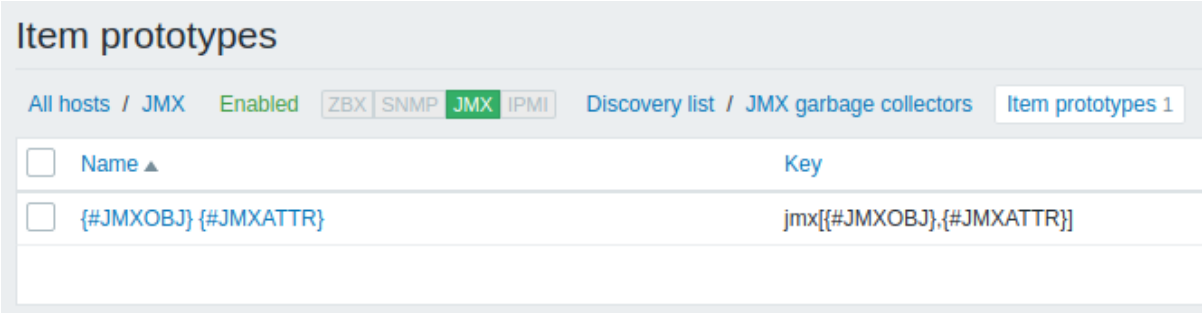
Discovery rule configuration collecting Mbean attributes looks like the following:



Key used:

```
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 it 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.

Discovering 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:

```
[
    {
        "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:

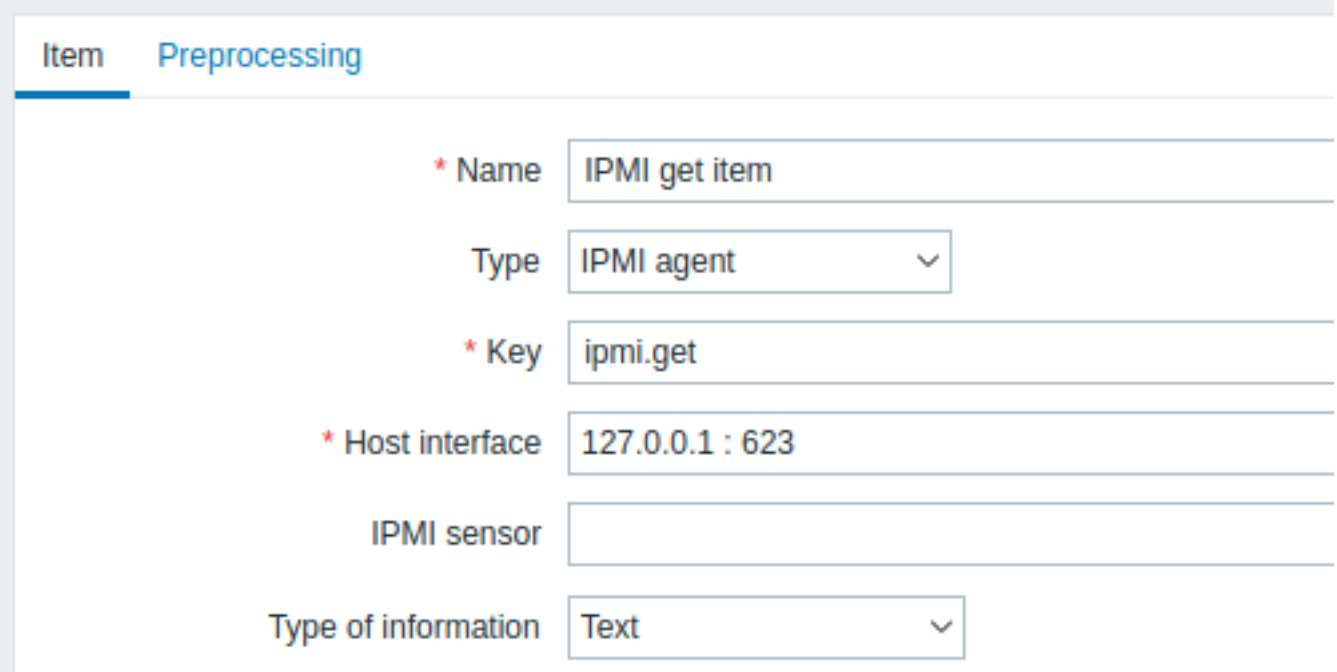
- the `ipmi.get` IPMI item (supported since Zabbix **5.0.0**) as the master item
- dependent low-level discovery rule and item prototypes

Configuration

Master item

Create an IPMI item using the following key:

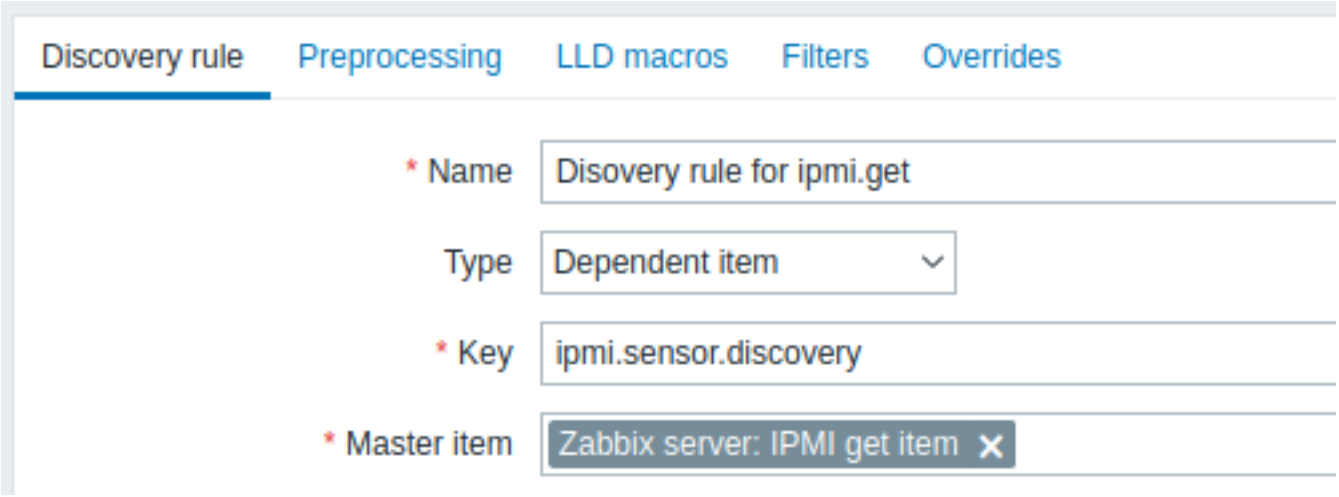
`ipmi.get`



Set the type of information to ”Text” for possibly big JSON data.

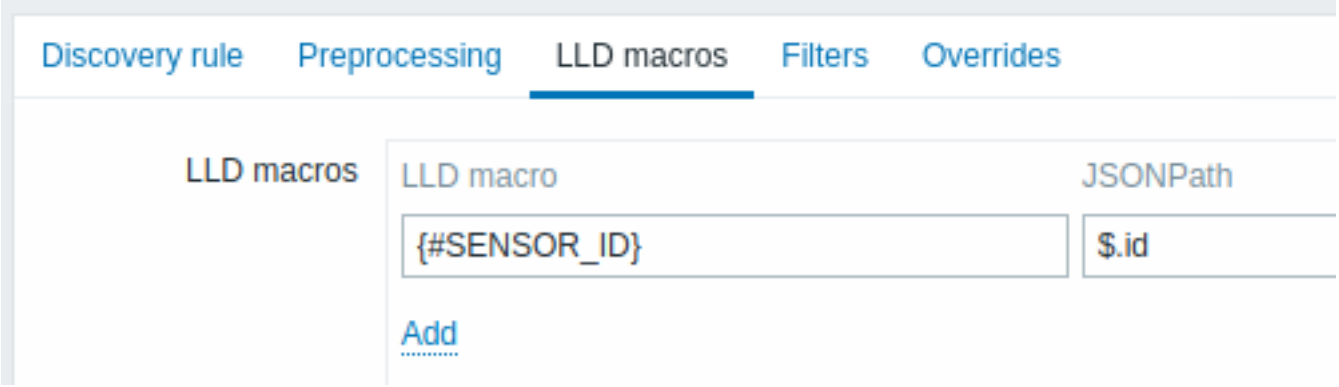
Dependent LLD rule

Create a low-level discovery rule as ”Dependent item” type:



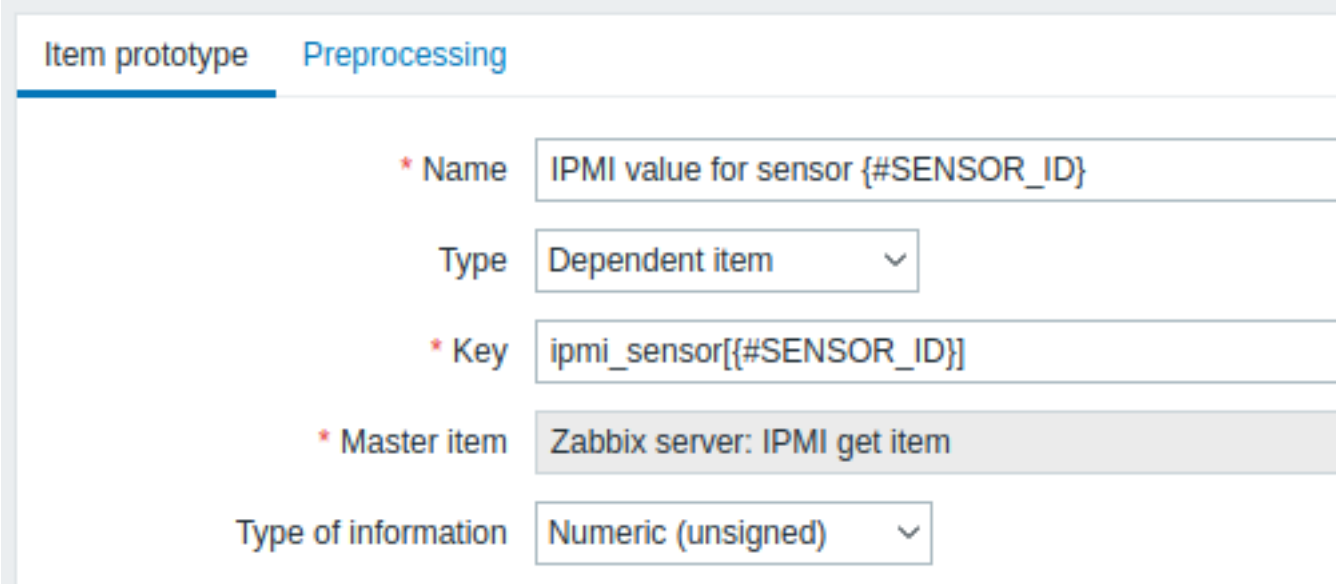
As master item select the `ipmi.get` item we created.

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.



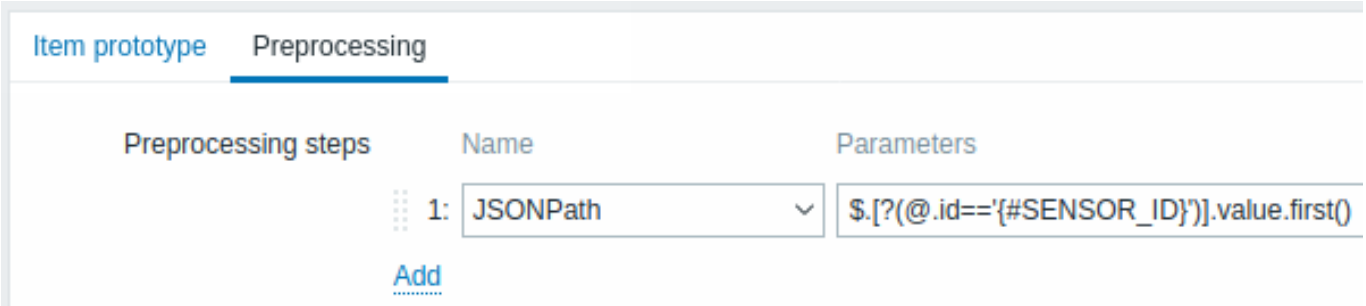
Note the use of the {#SENSOR_ID} macro in the item prototype name and key:

- *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`

> **Attention:**
> This item key is only supported in Zabbix agent 2.

This item returns a JSON with information about systemd units, for example:

```
[{
    "{#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}": "enabled"
}]
```

Supported macros

The following macros are supported for use in the discovery rule filter and prototypes of items, triggers and graphs:

Macro	Description
{#UNIT.NAME}	Primary unit name.
{#UNIT.DESCRIPTION}	Human readable description.
{#UNIT.LOADSTATE}	Load state (i.e. whether the unit file has been loaded successfully)
{#UNIT.ACTIVESTATE}	Active state (i.e. whether the unit is currently started or not)

Macro	Description
{#UNIT.SUBSTATE}	Sub state (a more fine-grained version of the active state that is specific to the unit type, which the active state is not)
{#UNIT.FOLLOWED}	Unit that is being followed in its state by this unit, if there is any; otherwise an empty string.
{#UNIT.PATH}	Unit object path.
{#UNIT.JOBID}	Numeric job ID if there is a job queued for the job unit; 0 otherwise.
{#UNIT.JOBTYPE}	Job type.
{#UNIT.JOBPATH}	Job object path.
{#UNIT.UNITFILESTATE}	The install state of the unit file (supported since 5.0.6).

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:

Macro	Description
{#SERVICE.NAME}	Service name.
{#SERVICE.DISPLAYNAME}	Displayed service name.
{#SERVICE.DESCRIPTION}	Service description.
{#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*).

Macro	Description
{#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_counter and perf_counter_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

`wmi.getall[<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",
        "Index" : 0,
        "InterfaceType" : "IDE"
    },
    {
        "DeviceID" : "\\.\PHYSICALDRIVE1",
        "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",
        "Index" : 1,
        "InterfaceType" : "IDE"
    }
]
```

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.

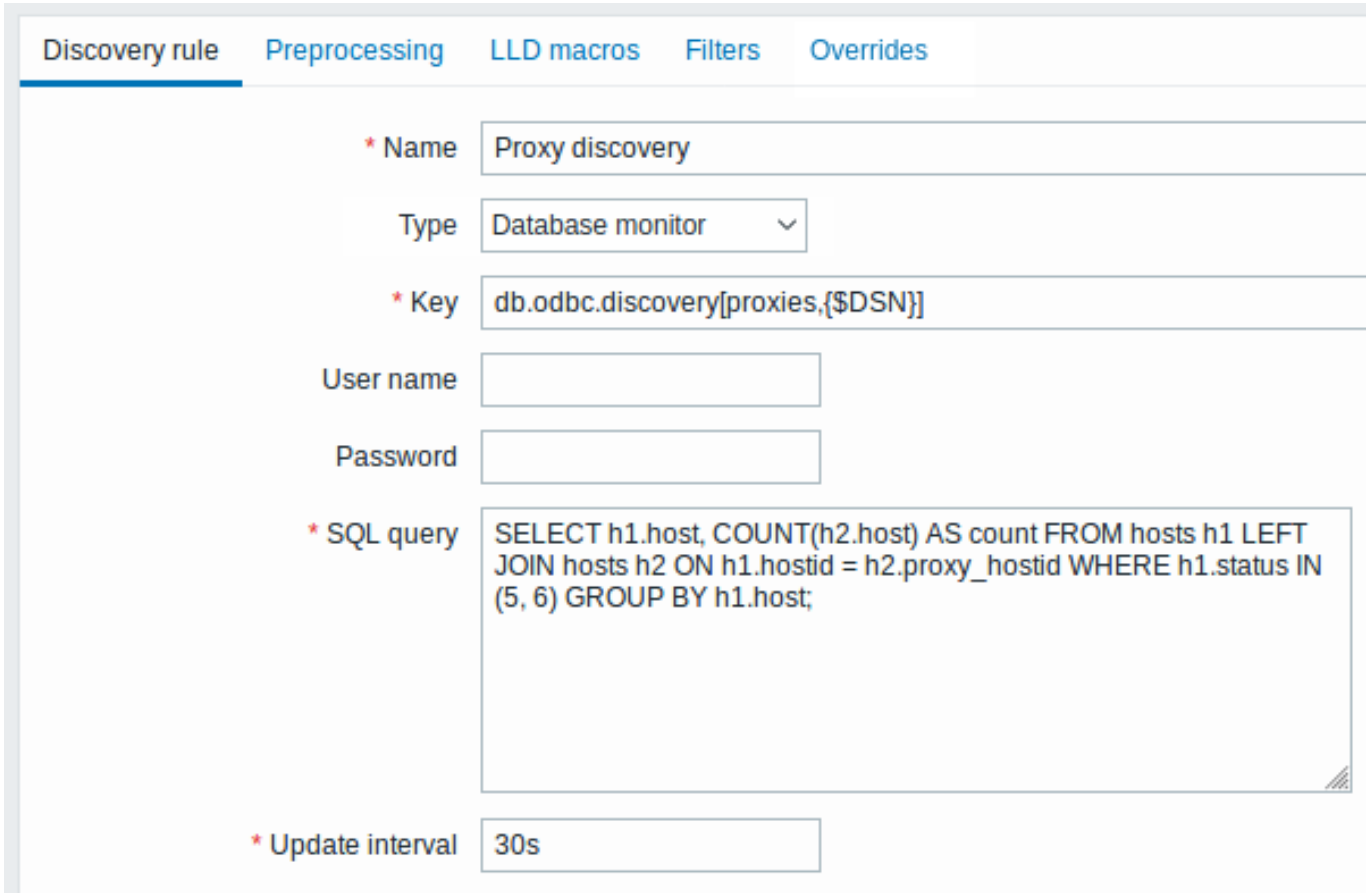
Two item keys may be used in "Database monitor" discovery rules:

- **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:

```
mysql> SELECT h1.host, COUNT(h2.host) AS count FROM hosts h1 LEFT JOIN hosts h2 ON h1.hostid = h2.proxy_ho
+---------+-------+
| 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:

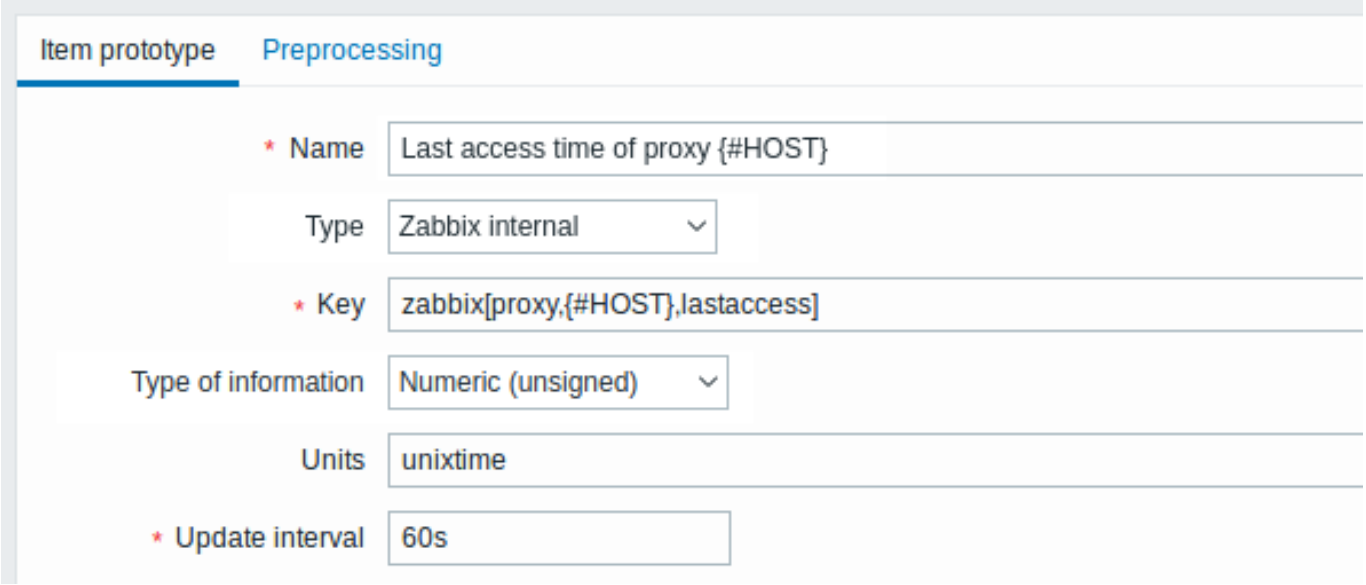
```
[
    {
        "{#HOST}": "Japan 1",
        "{#COUNT}": "5"
    },
    {
        "{#HOST}": "Japan 2",
        "{#COUNT}": "12"
    },
    {
        "{#HOST}": "Latvia",
        "{#COUNT}": "3"
    }
]
```

It can be seen that column names become macro names and selected rows become the values of these macros.

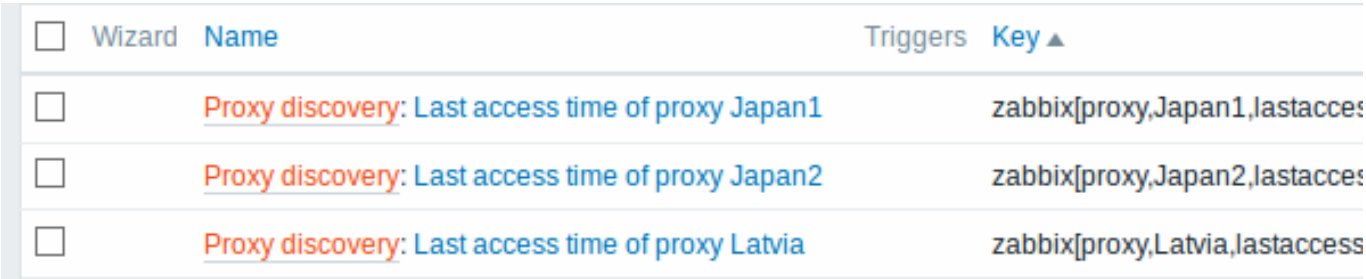
Note:
If it is not obvious how a column name would be transformed into a macro name, it is suggested to use column aliases like "COUNT(h2.host) AS count" in the example above.
In case a column name cannot be converted into a valid macro name, the discovery rule becomes not supported, with the error message 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 L
 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
```

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_ho
+---------+-------+
| host    | count |
+---------+-------+
| Japan 1 |     5 |
| Japan 2 |    12 |
| Latvia  |     3 |
+---------+-------+
3 rows in set (0.01 sec)
```

this JSON will be returned:

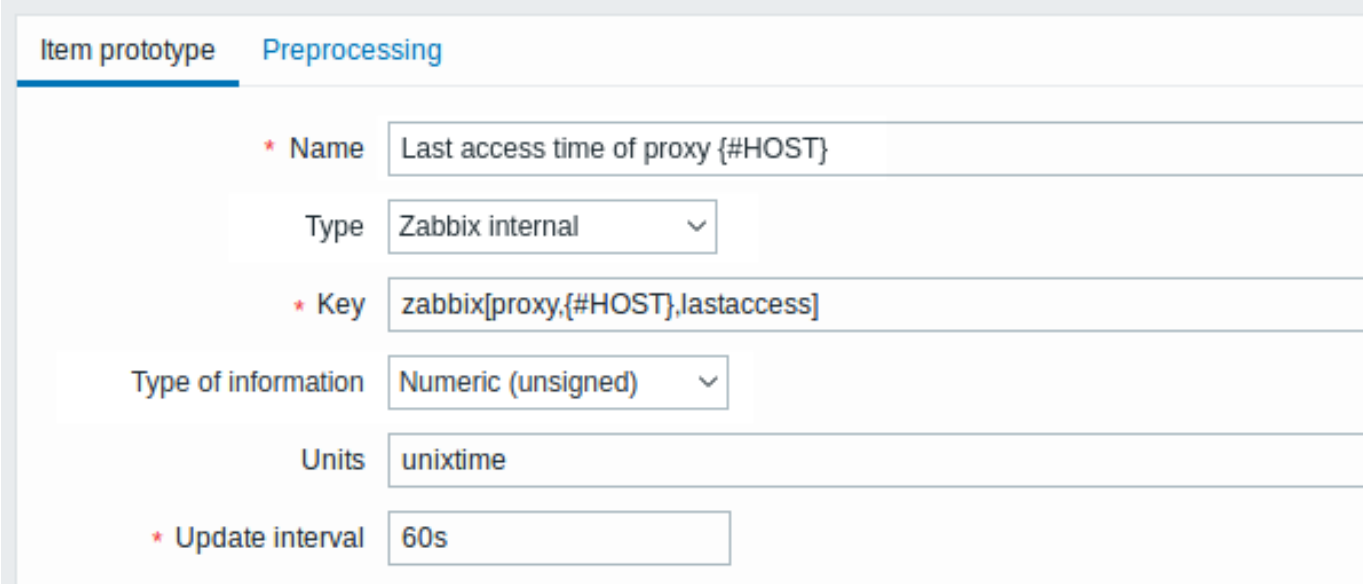
```
[
    {
        "host": "Japan 1",
```

```
        "count": "5"
    },
    {
        "host": "Japan 2",
        "count": "12"
    },
    {
        "host": "Latvia",
        "count": "3"
    }
]
```

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

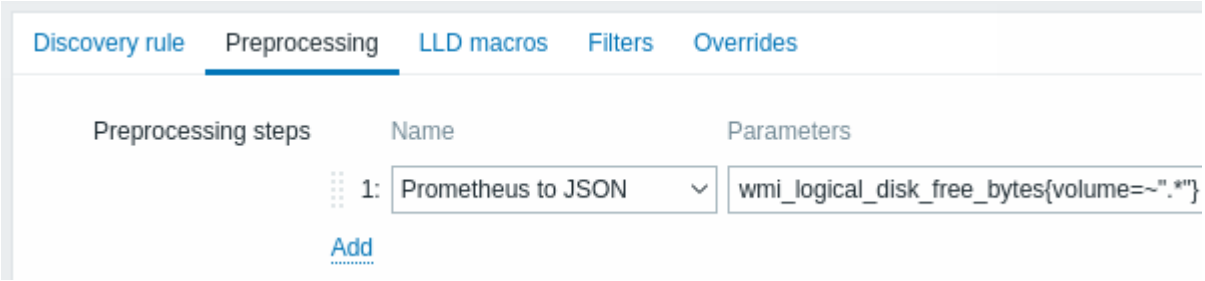
The low-level discovery rule should be created as a dependent item to the HTTP master item that collects Prometheus data.

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`:



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"
    }
]
```

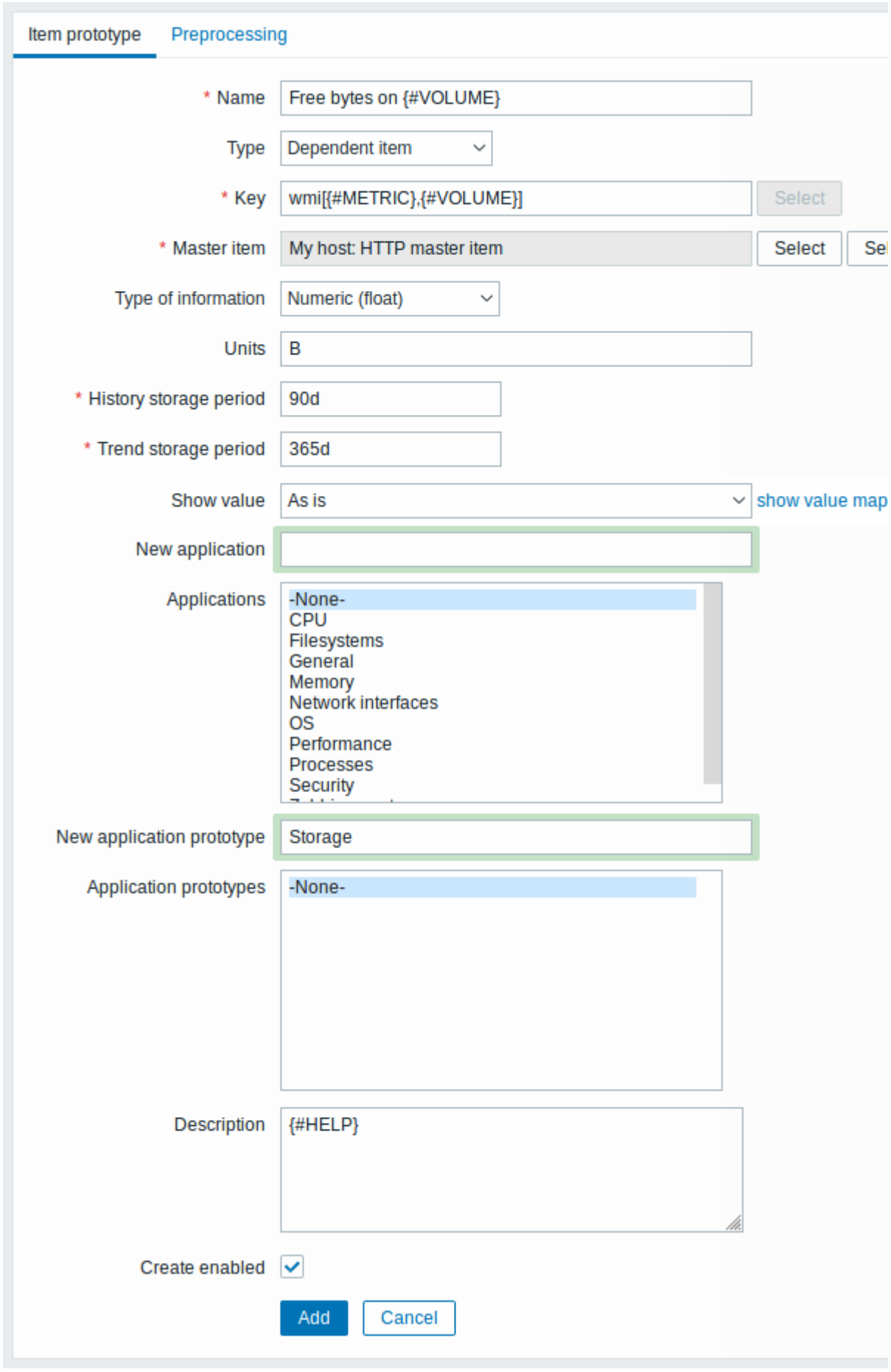
Mapping LLD macros

Next you have to go to the LLD macros tab and make the following mappings:

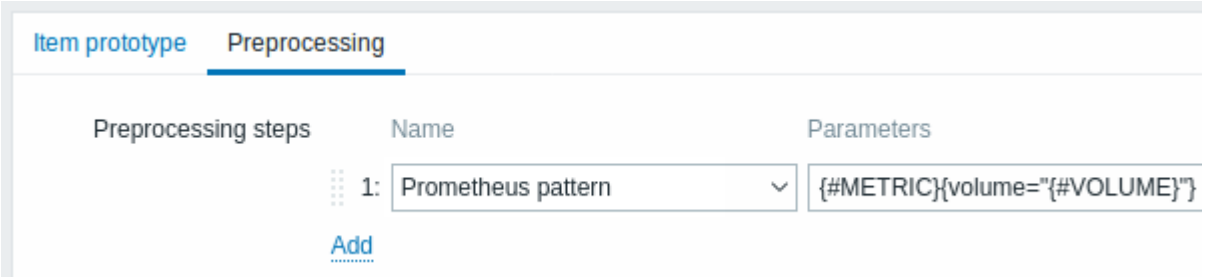
```
{#VOLUME}=$.labels['volume']
{#METRIC}=$['name']
{#HELP}=$['help']
```

Item prototype

You may want to create an item prototype like this:



with preprocessing options:



13 Discovery of block devices

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:

- 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.:

```
[
   {
      "{#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:

`[{"{#IF.CONN}":"192.168.3.1","{#IF.IP}":"192.168.3.1","{#IF.DNS}":"","{#IF.PORT}":"10050","{#IF.TYPE}":"AG`

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:

Macro	Description
{#IF.CONN}	Interface IP address or DNS host name.
{#IF.IP}	Interface IP address.
{#IF.DNS}	Interface DNS host name.
{#IF.PORT}	Interface port number.
{#IF.TYPE}	Interface type (”AGENT”, ”SNMP”, ”JMX”, or ”IPMI”).
{#IF.DEFAULT}	Default status for the interface: 0 - not default interface 1 - default interface
{#IF.SNMP.BULK}	SNMP bulk processing status for the interface: 0 - disabled 1 - enabled This macro is returned only if interface type is “SNMP”.

Notes on low-level discovery

Application discovery

Application prototypes support LLD macros.

One application prototype can be used by several item prototypes of the same discovery rule.

If created application prototype is not used by any item prototype it gets removed from ’Application prototypes’ list automatically.

Like other discovered entities applications follow the lifetime defined in discovery rule (’keep lost resources period’ setting) - they are removed after not being discovered for the specified number of days.

If an application is not discovered anymore, the application itself may not be removed because of the ’lost resources period’ setting, however:

- items that are still discovered are automatically removed from it;
- items that are no longer discovered are not removed from it.

Application prototypes defined by one discovery rule can’t discover the same application. In this situation only the first prototype discovery will succeed, the rest will report appropriate LLD error. Only application prototypes defined in different discovery rules can result in discovering the same application.

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.

	Proxy
Lightweight	**Yes**
GUI	No
Works independently	**Yes**
Easy maintenance	**Yes**
Automatic DB creation	**Yes**[1]
Local administration	No
Ready for embedded hardware	**Yes**
One way TCP connections	**Yes**
Centralized configuration	**Yes**
Generates notifications	No

[1] Automatic DB creation feature works only with SQLite. Other databases require manual setup.

1 Proxies

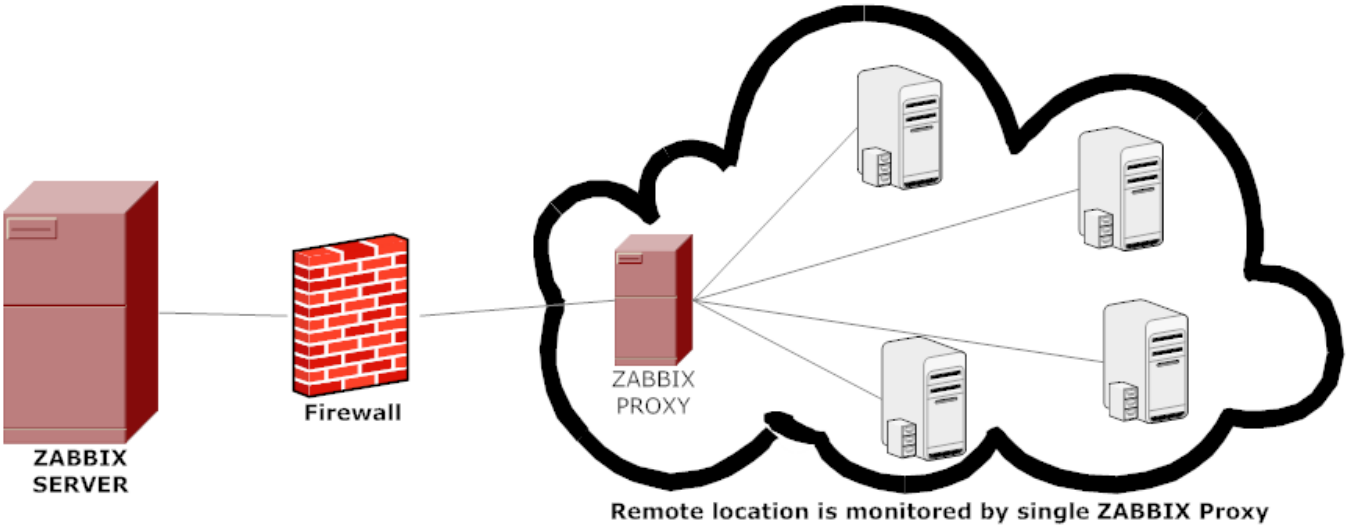
Overview

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.

Attention:
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.

Attention:
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:

Function		Supported by proxy
Items		
	Zabbix agent checks	**Yes**
	Zabbix agent checks (active)	**Yes** [1]
	Simple checks	**Yes**
	Trapper items	**Yes**
	SNMP checks	**Yes**
	SNMP traps	**Yes**
	IPMI checks	**Yes**
	JMX checks	**Yes**
	Log file monitoring	**Yes**
	Internal checks	**Yes**
	SSH checks	**Yes**
	Telnet checks	**Yes**
	External checks	**Yes**
	Dependent items	**Yes**
Built-in web monitoring		**Yes**
Item value preprocessing		**Yes**
Network discovery		**Yes**
Active agent autoregistration		**Yes**
Low-level discovery		**Yes**
Remote commands		**Yes**
Calculating triggers		*No*
Processing events		*No*
Event correlation		*No*
Sending alerts		*No*

Note:
[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.

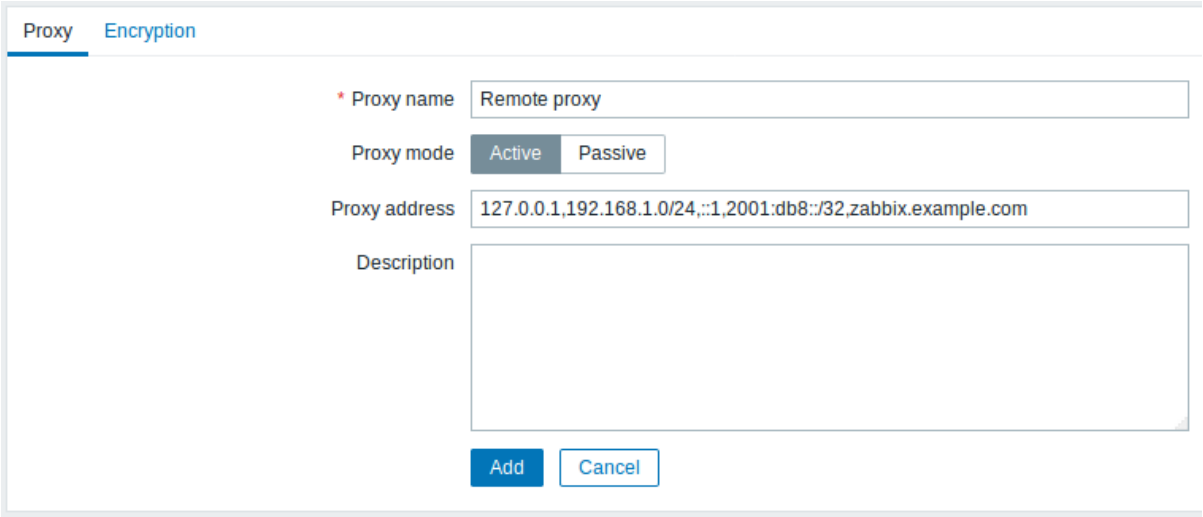
Configuration

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*



Parameter		Description
Proxy name		Enter the proxy name. It must be the same name as in the *Hostname* parameter in the proxy configuration file.
Proxy mode		Select the proxy mode. **Active** - the proxy will connect to the Zabbix server and request configuration data **Passive** - 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.
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.

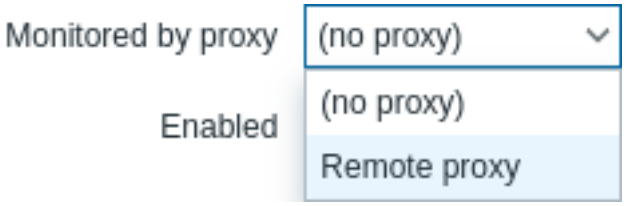
The **Encryption** tab allows you to require encrypted connections with the proxy.

Parameter	Description
Connections to proxy	How the server connects to the passive proxy: no encryption (default), using PSK (pre-shared key) or certificate.
Connections from proxy	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 ”No encryption”.

Parameter	Description
Issuer	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.
Subject	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.
PSK identity	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.
PSK	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

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.



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. Note that 3.0.x is supported since Zabbix 5.0.28.
- 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.

Note:
You can find out more about setting up SSL for Zabbix frontend by referring to these best practices.

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 --wit
```

Different Zabbix components may be compiled with different crypto libraries (e.g. a server with *OpenSSL*, an agent with *GnuTLS*).

Attention:
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:

Example	Connections to host	Allowed connections from host	Rejected connections from host
NONE	Unencrypted	Unencrypted	Encrypted, certificate and PSK-based encrypted
CERT NONE PSK CERT	Encrypted, certificate-based	Encrypted, certificate-based	Unencrypted and PSK-based encrypted
PSK NONE PSK CERT	Encrypted, PSK-based	Encrypted, PSK-based	Unencrypted and certificate-based encrypted
PSK NONE PSK CERT	Encrypted, PSK-based	Unencrypted and PSK-based encrypted	Certificate-based encrypted
CERT NONE PSK CERT	Encrypted, certificate-based	Unencrypted, PSK or certificate-based encrypted	-

Attention:
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):

Library	Certificate ciphersuites	PSK ciphersuites
GnuTLS 3.1.18	TLS_ECDHE_RSA_AES_128_GCM_SHA256 TLS_ECDHE_RSA_AES_128_CBC_SHA256 TLS_ECDHE_RSA_AES_128_CBC_SHA1 TLS_RSA_AES_128_GCM_SHA256 TLS_RSA_AES_128_CBC_SHA256 TLS_RSA_AES_128_CBC_SHA1	TLS_ECDHE_PSK_AES_128_CBC_SHA256 TLS_ECDHE_PSK_AES_128_CBC_SHA1 TLS_PSK_AES_128_GCM_SHA256 TLS_PSK_AES_128_CBC_SHA256 TLS_PSK_AES_128_CBC_SHA1
OpenSSL 1.0.2c	ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-SHA256 AES128-SHA	PSK-AES128-CBC-SHA

Library	Certificate ciphersuites	PSK ciphersuites
OpenSSL 1.1.0	ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-CCM8 AES128-CCM AES128-SHA256 AES128-SHA	ECDHE-PSK-AES128-CBC-SHA256 ECDHE-PSK-AES128-CBC-SHA PSK-AES128-GCM-SHA256 PSK-AES128-CCM8 PSK-AES128-CCM PSK-AES128-CBC-SHA256 PSK-AES128-CBC-SHA
OpenSSL 1.1.1d	TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-CCM8 AES128-CCM AES128-SHA256 AES128-SHA	TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-PSK-AES128-CBC-SHA256 ECDHE-PSK-AES128-CBC-SHA PSK-AES128-GCM-SHA256 PSK-AES128-CCM8 PSK-AES128-CCM PSK-AES128-CBC-SHA256 PSK-AES128-CBC-SHA

User-configured ciphersuites The built-in ciphersuite selection criteria can be overridden with user-configured ciphersuites.

> **Attention:**
> 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:

Override scope	Parameter	Value	Description
Ciphersuite selection for certificates	TLSCipherCert13	Valid OpenSSL 1.1.1 cipher strings for TLS 1.3 protocol (their values are passed to the OpenSSL function SSL_CTX_set_ciphersuites()).	Certificate-based ciphersuite selection criteria for TLS 1.3 Only OpenSSL 1.1.1 or newer.
	TLSCipherCert	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.	Certificate-based ciphersuite selection criteria for TLS 1.2/1.3 (GnuTLS), TLS 1.2 (OpenSSL)

Override scope	Parameter	Value	Description
Ciphersuite selection for PSK	TLSCipherPSK13	Valid OpenSSL 1.1.1 cipher strings for TLS 1.3 protocol (their values are passed to the OpenSSL function SSL_CTX_set_ciphersuites()).	PSK-based ciphersuite selection criteria for TLS 1.3 Only OpenSSL 1.1.1 or newer.
	TLSCipherPSK	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.	PSK-based ciphersuite selection criteria for TLS 1.2/1.3 (GnuTLS), TLS 1.2 (OpenSSL)
Combined ciphersuite list for certificate and PSK	TLSCipherAll13	Valid OpenSSL 1.1.1 cipher strings for TLS 1.3 protocol (their values are passed to the OpenSSL function SSL_CTX_set_ciphersuites()).	Ciphersuite selection criteria for TLS 1.3 Only OpenSSL 1.1.1 or newer.
	TLSCipherAll	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.	Ciphersuite selection criteria for TLS 1.2/1.3 (GnuTLS), TLS 1.2 (OpenSSL)

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**:

Agent connection setup	Cipher configuration
TLSConnect=cert	TLSCipherCert, TLSCipherCert13
TLSConnect=psk	TLSCipherPSK, TLSCipherPSK13
TLSAccept=cert	TLSCipherCert, TLSCipherCert13
TLSAccept=psk	TLSCipherPSK, TLSCipherPSK13
TLSAccept=cert,psk	TLSCipherAll, TLSCipherAll13

For Zabbix **server** and **proxy**:

Connection setup	Cipher configuration
Outgoing connections using PSK	TLSCipherPSK, TLSCipherPSK13
Incoming connections using certificates	TLSCipherAll, TLSCipherAll13
Incoming connections using PSK if server has no certificate	TLSCipherPSK, TLSCipherPSK13
Incoming connections using PSK if server has certificate	TLSCipherAll, TLSCipherAll13

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
TLSCipherCert	NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509
TLSCipherPSK	NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL
TLSCipherAll	NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509

Parameter	OpenSSL 1.1.1d [1]
TLSCipherCert13	
TLSCipherCert	EECDH+aRSA+AES128:RSA+aRSA+AES128
TLSCipherPSK13	TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
TLSCipherPSK	kECDHEPSK+AES128:kPSK+AES128
TLSCipherAll13	
TLSCipherAll	EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+AES128

[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:

```
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:

```
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_
zabbix_sender [41271]: DEBUG: zbx_tls_init_child() PSK ciphersuites: TLS_CHACHA20_POLY1305_SHA256 TLS_AE
zabbix_sender [41271]: DEBUG: zbx_tls_init_child() certificate and PSK ciphersuites: TLS_AES_256_GCM_SHA
```

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 `TLSCipher*` parameters.

> **Attention:**
> 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:

```
TLSCipherCert=EECDH+aRSA+AES128
TLSCipherPSK=kECDHEPSK+AES128
```

and test again:

```
$ zabbix_sender -vv -c /home/zabbix/test.conf -k nonexisting_item -o 1 2>&1 | grep ciphersuites
zabbix_sender [42892]: DEBUG: zbx_tls_init_child() certificate ciphersuites: TLS_AES_256_GCM_SHA384 TLS_
zabbix_sender [42892]: DEBUG: zbx_tls_init_child() PSK ciphersuites: TLS_CHACHA20_POLY1305_SHA256 TLS_AE
zabbix_sender [42892]: DEBUG: zbx_tls_init_child() certificate and PSK ciphersuites: TLS_AES_256_GCM_SHA
```

The "certificate ciphersuites" and "PSK ciphersuites" lists have changed - they are shorter than before, only containing TLS 1.3 ciphersuites and TLS 1.2 ECDHE-* ciphersuites as expected.

2. TLSCipherAll and TLSCipherAll13 cannot be tested with zabbix_sender; they do not affect "certificate and PSK ciphersuites" value shown in the example above. To tweak TLSCipherAll and TLSCipherAll13 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

```
TLSCipherCert=EECDH+aRSA+AES128
TLSCipherPSK=kECDHEPSK+AES128
TLSCipherAll=EECDH+aRSA+AES128:kECDHEPSK+AES128
```

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:

```
TLSCipherPSK=kECDHEPSK+AES128
```

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 `TLSCipher*` parameter values:

```
$ openssl ciphers EECDH+aRSA+AES128 | sed 's/:/ /g'
TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-RSA-AES128-GCM-SHA256 E
$ openssl ciphers kECDHEPSK+AES128 | sed 's/:/ /g'
TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-PSK-AES128-CBC-SHA256 E
$ openssl ciphers EECDH+aRSA+AES128:kECDHEPSK+AES128 | sed 's/:/ /g'
TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-RSA-AES128-GCM-SHA256 E
```

You may prefer `openssl ciphers` with option `-V` for a more verbose output:

```
$ openssl ciphers -V EECDH+aRSA+AES128:kECDHEPSK+AES128
           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
           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(128) Mac=AEAD
           0xC0,0x27 - ECDHE-RSA-AES128-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(128)  Mac=SHA256
           0xC0,0x13 - ECDHE-RSA-AES128-SHA    TLSv1 Kx=ECDH     Au=RSA  Enc=AES(128)  Mac=SHA1
           0xC0,0x37 - ECDHE-PSK-AES128-CBC-SHA256 TLSv1 Kx=ECDHEPSK Au=PSK  Enc=AES(128)  Mac=SHA256
           0xC0,0x35 - ECDHE-PSK-AES128-CBC-SHA TLSv1 Kx=ECDHEPSK Au=PSK  Enc=AES(128)  Mac=SHA1
```

Similarly, you can test the priority strings for GnuTLS:

```
$ gnutls-cli -l --priority=NONE:+VERS-TLS1.2:+ECDHE-RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+CURVE-A
Cipher suites for NONE:+VERS-TLS1.2:+ECDHE-RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+CURVE-ALL:+COMP-
TLS_ECDHE_RSA_AES_128_GCM_SHA256                        0xc0, 0x2f      TLS1.2
TLS_ECDHE_RSA_AES_128_CBC_SHA256                        0xc0, 0x27      TLS1.2

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, GR
PK-signatures: SIGN-RSA-SHA256, SIGN-RSA-PSS-SHA256, SIGN-RSA-PSS-RSAE-SHA256, SIGN-ECDSA-SHA256, SIGN-E
```

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
```

Attention:
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, OpenSSL PKI Tutorial v1.1 .

Carefully consider and test your certificate extensions - see Limitations on using X.509 v3 certificate extensions.

Certificate configuration parameters

Parameter	Mandatory	Description
TLSCAFile	*	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.
TLSCRLFile		Full pathname of a file containing Certificate Revocation Lists. See notes in Certificate Revocation Lists (CRL).
TLSCertFile	*	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).
TLSKeyFile	*	Full pathname of a file containing private key. Set access rights to this file - it must be readable only by Zabbix user.
TLSServerCertIssuer		Allowed server certificate issuer.
TLSServerCertSubject		Allowed server certificate subject.

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:
    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+MRMwEQYKCZImiZPyLGQB
....
9wEzdN8uTrqoyU78gi12npLj08LegRKjb5hFTVmO
-----END 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-----
MIID3DCCAsSgAwIBAgIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLGQB
...
vdGNYoSfvu41GQAR5Vj5FnRJRzv5XQOZ3B6894GY1zY=
-----END CERTIFICATE-----
```

2. Put Zabbix server certificate chain into file, for example, `/home/zabbix/zabbix_server.crt`:

```
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-----
MIIECDCCAvCgAwIBAgIBATANBgkqhkiG9w0BAQUFADCBgTETMBEGCgmSJomT8ixk
...
h02u1GHiy46GI+xfR3LsPwFKlkTaaLaL/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+MRMwEQYKCZImiZPyLGQB
...
dyCeWnvL7u5sd6ffo8iRny0QzbHKmQt/wUtcVIvWXdMIFJM0Hw==
-----END CERTIFICATE-----
```

Here the first is Zabbix server certificate, followed by intermediate CA certificate.

Note:
Use of any attributes except of the ones mentioned above is discouraged for both client and server certificates, because it may affect certificate verification process. For example, OpenSSL might fail to establish encrypted connection if *X509v3 Extended Key Usage* or *Netscape Cert Type* are set. See also: Limitations on using X.509 v3 certificate extensions.

3. Put Zabbix server private key into file, for example, `/home/zabbix/zabbix_server.key`:

```
-----BEGIN PRIVATE KEY-----
MIIEwAIBADANBgkqhkiG9w0BAQEFAASCBKowggSmAgEAAoIBAQC9tIXIJoVnNXDl
...
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:

```
TLSAccept=cert
```

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:

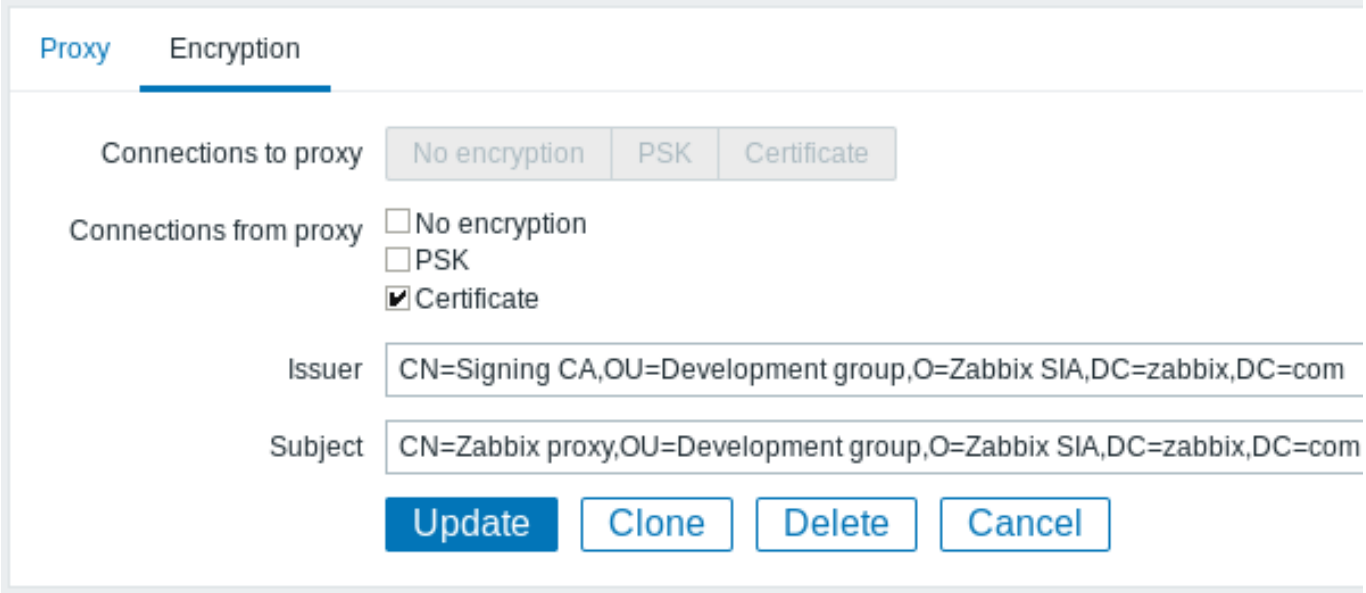
```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix server,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSCertFile=/home/zabbix/zabbix_proxy.crt
TLSKeyFile=/home/zabbix/zabbix_proxy.key
```

5. Configure encryption for this proxy in Zabbix frontend:

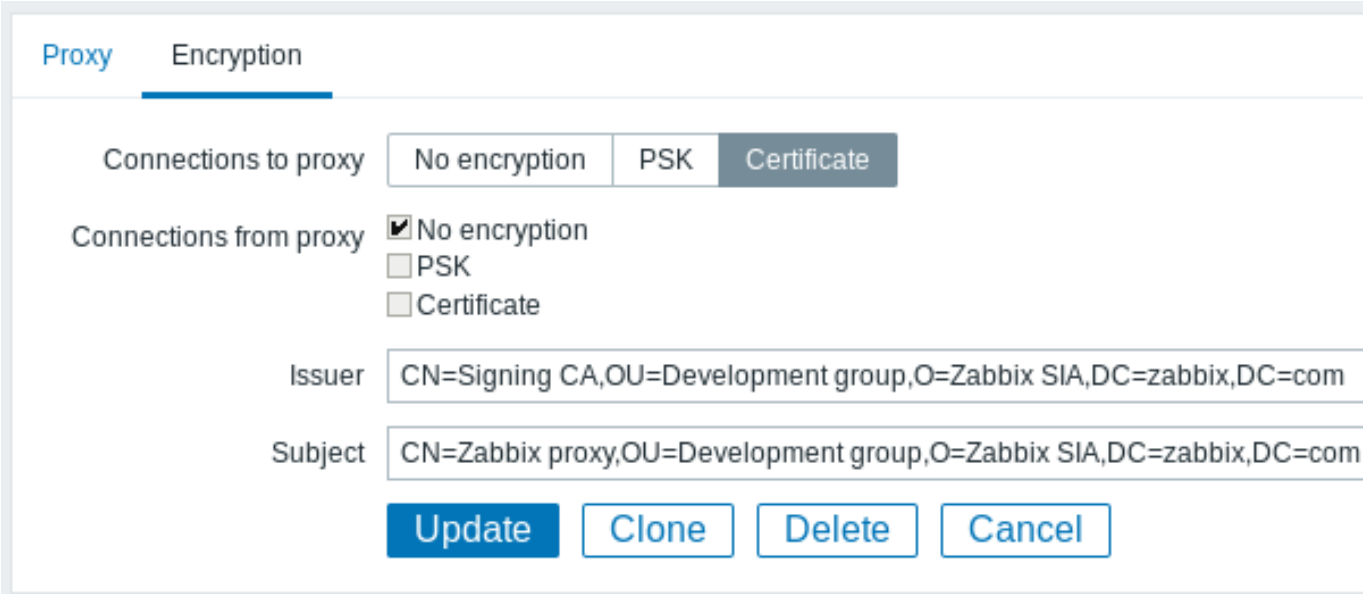
- Go to: *Administration* → *Proxies*
- Select proxy and click on **Encryption** tab

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



For passive proxy



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:

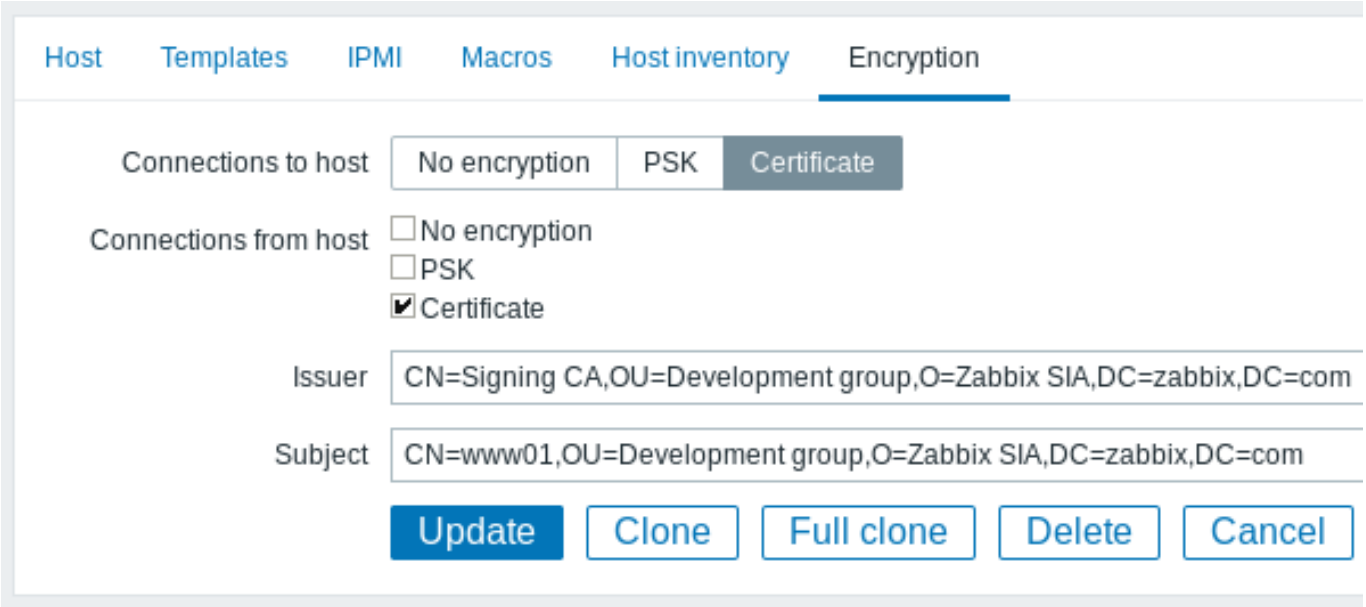
```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSCertFile=/home/zabbix/zabbix_agentd.crt
TLSKeyFile=/home/zabbix/zabbix_agentd.key
```

(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.



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:
    1. escape characters '"' (U+0022), '+' U+002B, ',' U+002C, ';' U+003B, '<' U+003C, '>' U+003E, '\' U+005C anywhere in string.
    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).
8. Requirements of RFC 4517 Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules and RFC 4518 Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation are not supported due to amount of work required.

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 low level (CN), proceeds to 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:

```
$ 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

$ openssl x509 -noout -text -in /home/zabbix/zabbix_proxy.crt
Certificate:
        ...
        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
```

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!

Attention:
To get proper Issuer and Subject strings usable in Zabbix invoke OpenSSL with special options `-nameopt esc_2253,esc_ctrl,utf8,dump_nostr,dump_unknown,dump_der,sep_comma_plus,dn_rev,sname`:

```
$ openssl x509 -noout -issuer -subject \
        -nameopt esc_2253,esc_ctrl,utf8,dump_nostr,dump_unknown,dump_der,sep_comma_plus,dn_rev,sname \
        -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
...
treZeUPjb7LSmZ3K2hpbZN7SoOZcAoHQ3GWd9npuctg=
-----END X509 CRL-----
-----BEGIN X509 CRL-----
MIIB+TCB4gIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLGQBGRYDY29t
...
CAEebS2CND3ShBedZ8YSil59O6JvaDP61lR5lNs=
-----END X509 CRL-----
```

CRL file is loaded only on Zabbix start. CRL update requires restart.

Attention:
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, "e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327b

Size limits

There are size limits for PSK identity and value in Zabbix, in some cases a crypto library can have lower limit:

Component	PSK identity max size	PSK value min size	PSK value max size
Zabbix	128 UTF-8 characters	128-bit (16-byte PSK, entered as 32 hexadecimal digits)	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
GnuTLS	128 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
OpenSSL 1.0.x, 1.1.0	127 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
OpenSSL 1.1.1	127 bytes (may include UTF-8 characters)	-	512-bit (64-byte PSK, entered as 128 hexadecimal digits)
OpenSSL 1.1.1a and later	127 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)

Attention:
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.

Attention:
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*:

```
$ openssl rand -hex 32
af8ced32dfe8714e548694e2d29e1a14ba6fa13f216cb35c19d0feb1084b0429
```

- with *GnuTLS*:

```
$ psktool -u psk_identity -p database.psk -s 32
Generating a random key for user 'psk_identity'
Key stored to database.psk

$ cat database.psk
psk_identity:9b8eafedfaae00cece62e85d5f4792c7d9c9bcc851b23216a1d300311cc4f7cb
```

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:

```
1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952
```

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:

```
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:

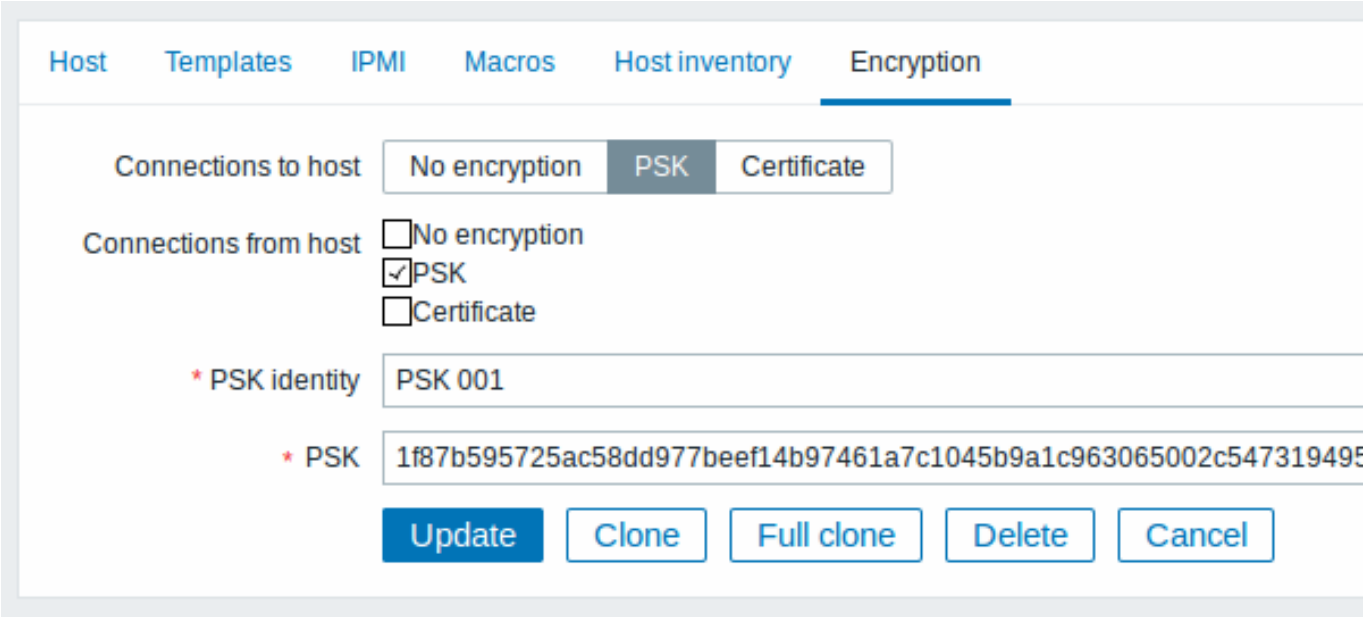
```
$ zabbix_get -s 127.0.0.1 -k "system.cpu.load[all,avg1]" --tls-connect=psk \
            --tls-psk-identity="PSK 001" --tls-psk-file=/home/zabbix/zabbix_agentd.psk
```

(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 "e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d083 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 `TLSAccept=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 pee
```

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 succ
...send value error: TCP successful, cannot establish TLS to [[192.168.1.2]:10051]: SSL_connect() I/O erro
```

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_ERR
```

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
    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:padd
```

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\ex
```

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.

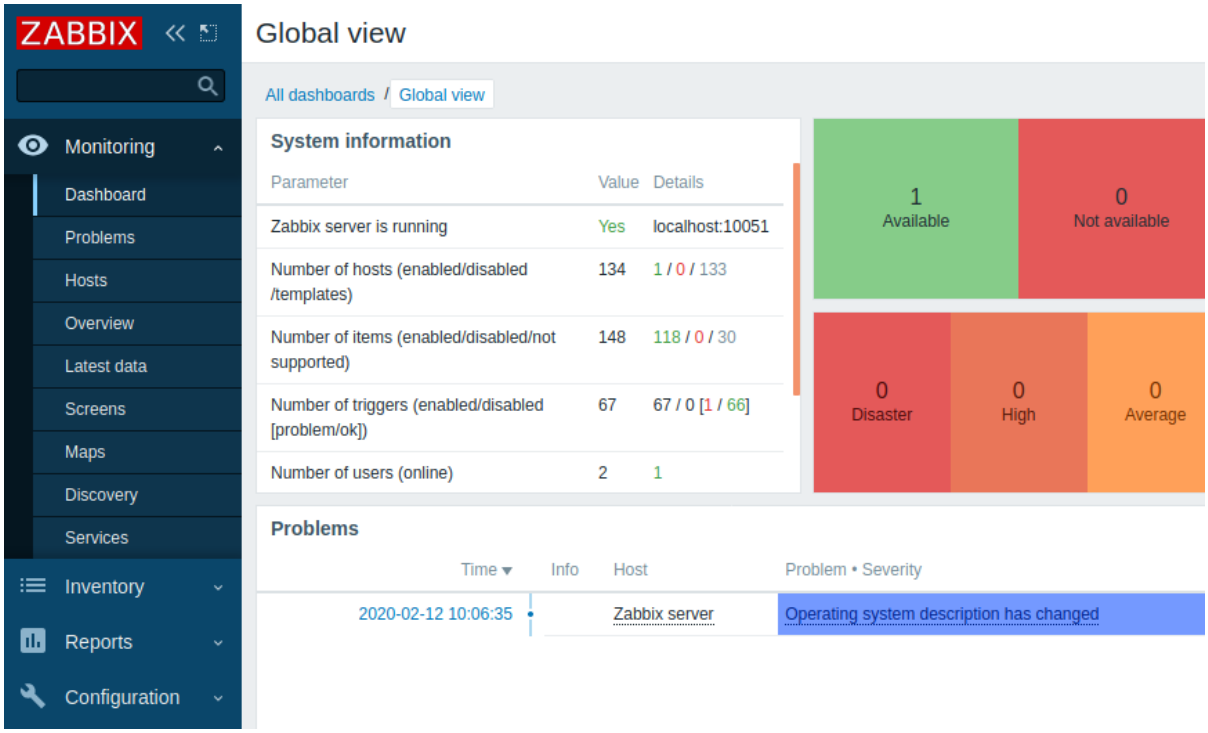
> **Note:**
> Trying to access two Zabbix frontend installations on the same host, on different ports, simultaneously will fail. Logging into the second one will terminate the session on the first one - unless the default frontend session name is adjusted for the second frontend in frontend definitions (see ZBX_SESSION_NAME).

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.

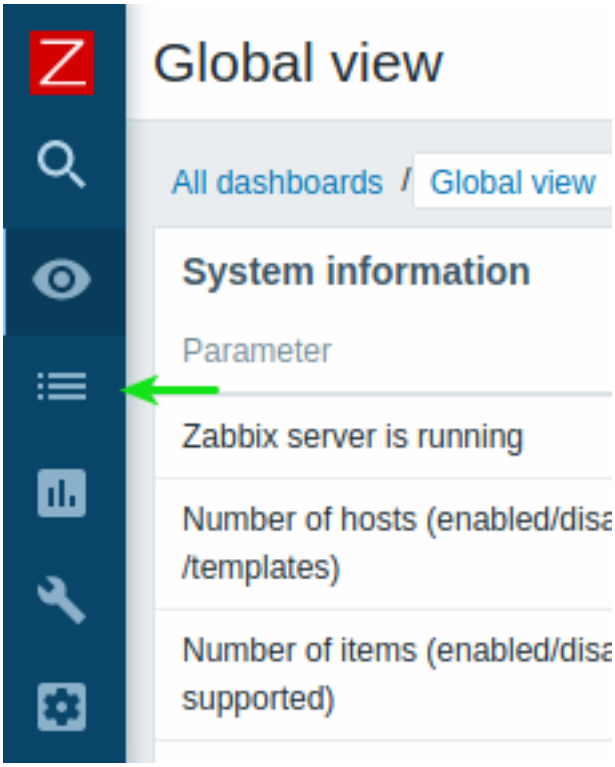


Working with the menu

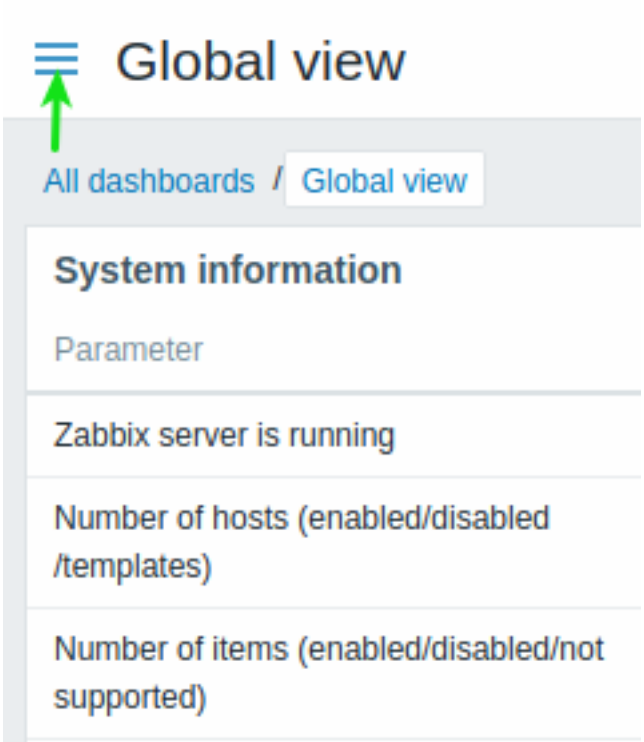
A global search box is located below the Zabbix logo.

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 with only the icons visible.



Hidden menu.

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. Note that you will be taken back to normal mode (not fullscreen mode).

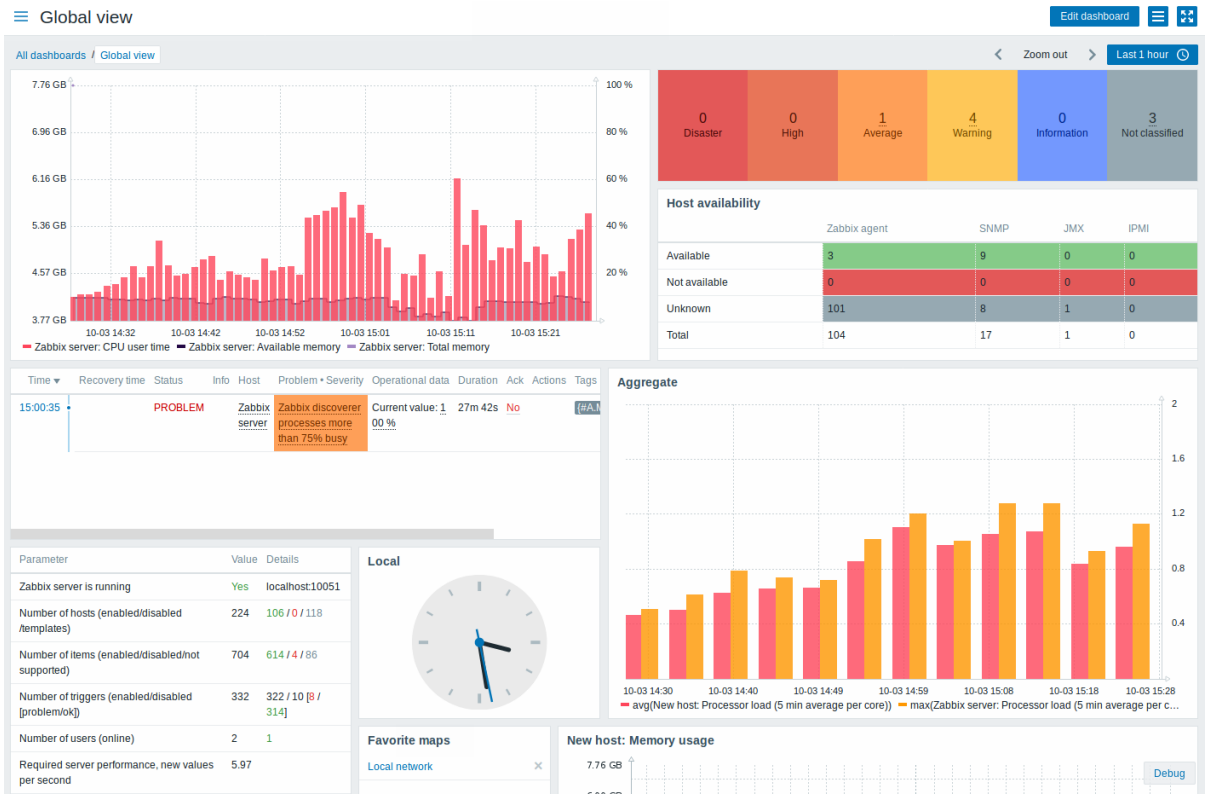
1 Dashboard

Overview

The *Monitoring* → *Dashboard* section is designed to display summaries of all the important information.

A dashboard 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.

Access to hosts in the widgets depends on host permissions.



Widgets are added and edited in the dashboard editing mode. Widgets are viewed in the dashboard viewing mode.

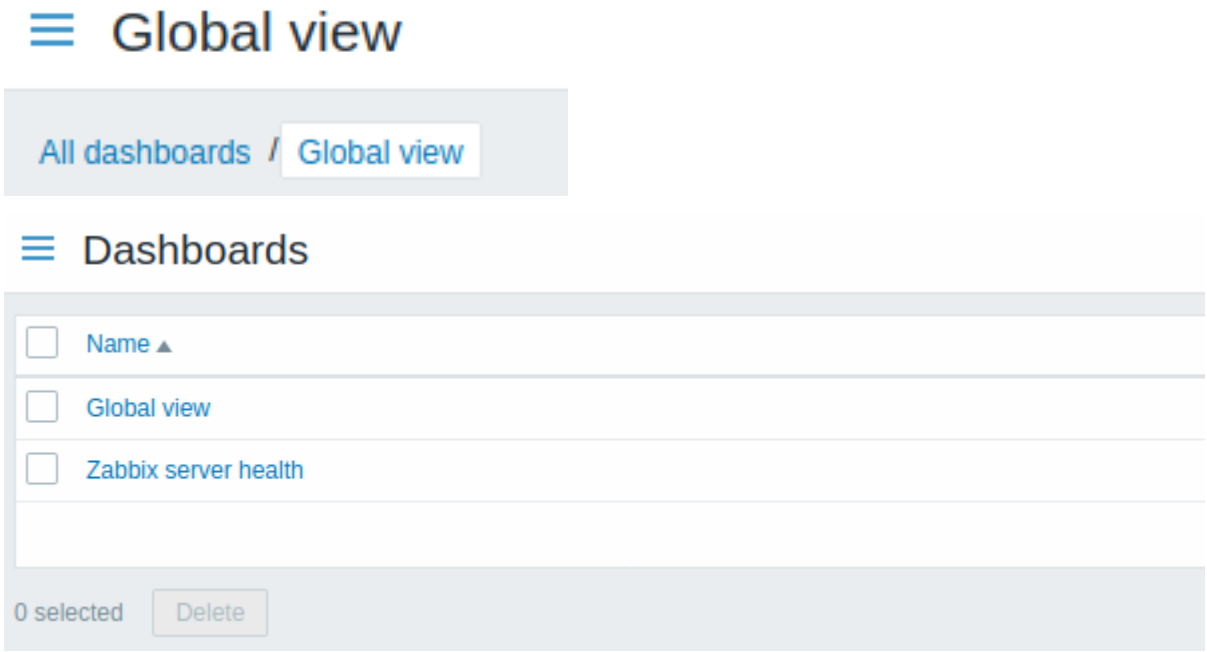
While in a single dashboard you can group widgets from various sources for a quick overview, it is also possible to create several dashboards containing different sets of overviews and switch between them.

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 to expand and collapse 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 access all configured dashboards, click on the *All dashboards* link just below the section title.

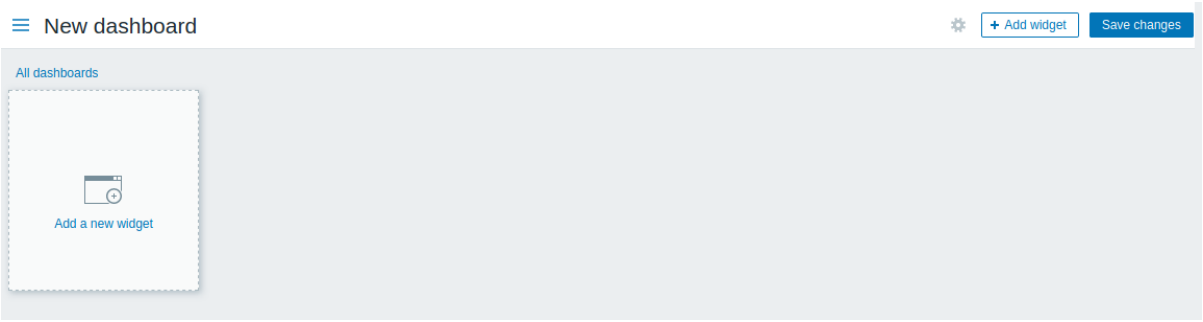


To view a single dashboard, click on its name in the list of all dashboards.

To delete one or several dashboards, mark the checkboxes of the respective dashboards and click on *Delete* below the list.

Creating a dashboard

When viewing all dashboards, you can click on the *Create dashboard* button to create a new dashboard:



Initially the dashboard is empty. To populate the dashboard, you can add widgets.

Click on the *Save changes* button to save the dashboard. If you click on *Cancel*, the dashboard will not be created.

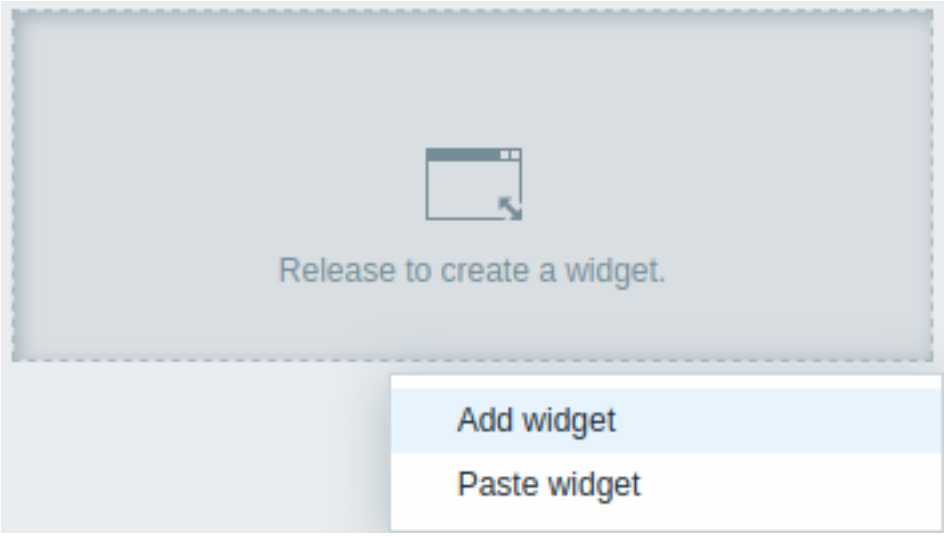
Adding widgets

To add a widget to a dashboard:

- Click on the *Add widget* button in dashboard editing mode. The widget will be created in its default size and placed after the existing widgets (if any);

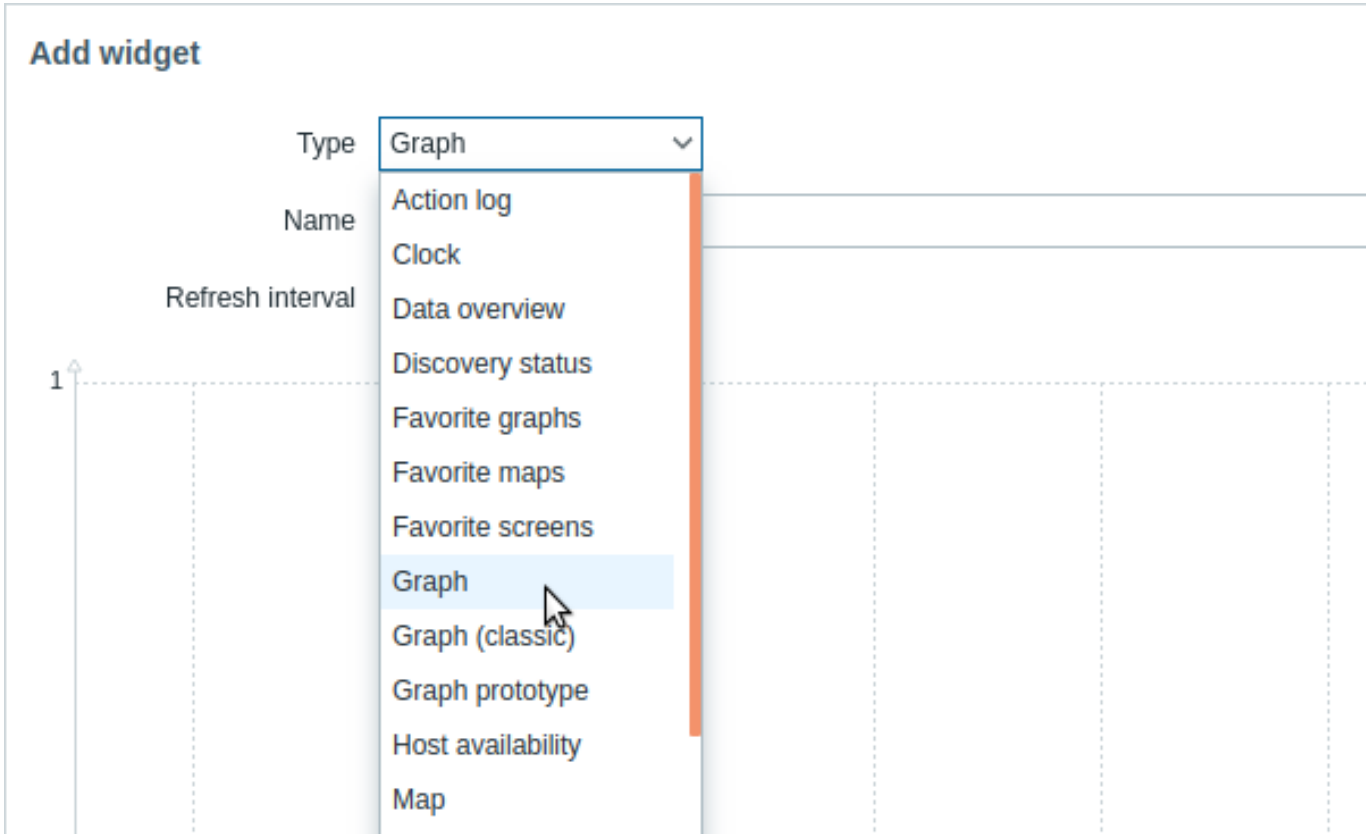
Or

- Move your mouse to the desired empty spot for the new widget. Notice how a placeholder appears, on mouseover, 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 take up all the available space if its default size is bigger. Alternatively you may click and drag the placeholder to the desired widget size, then release. (When releasing you may need to click on *Add widget* if the *Paste widget* from a copied widget option is also available.)



Then, in the widget configuration form:

- Select the *Type* of widget
- Enter widget parameters
- Click on *Add*



The following widgets can be added to a dashboard:

- Action log
- Clock
- Data overview
- Discovery status
- Favorite graphs
- Favorite maps
- Favorite screens
- Graph
- Graph (classic)
- Graph prototype
- Host availability
- Problem hosts
- Map
- Map navigation tree
- Plain text
- Problems
- System information
- Problems by severity
- Trigger overview
- URL
- Web monitoring

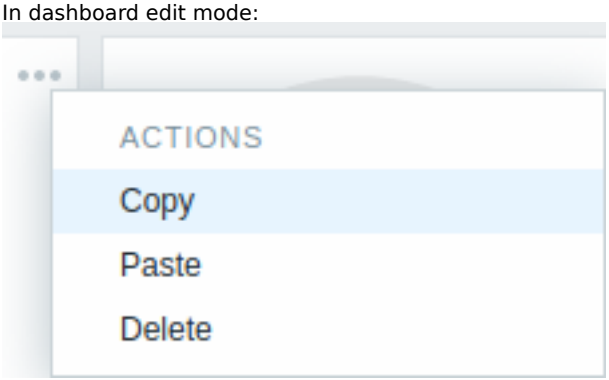
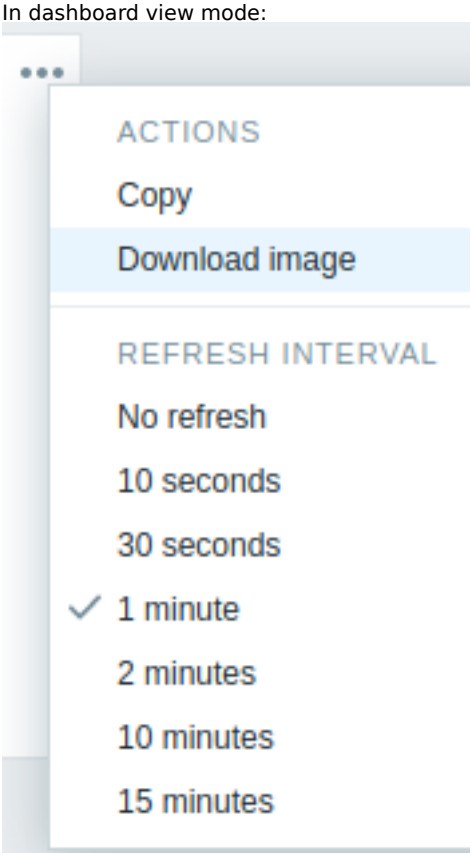
In dashboard editing mode widgets can be resized and moved around the dashboard by clicking on the widget title bar and dragging it to a new location. Also, you can click on the following buttons in the top-right corner of the widget to:

- - edit a widget;
- - access the widget menu

Click on *Save changes* for the dashboard to make any changes to the widgets permanent.

Widget menu

The widget menu contains different options based on whether the dashboard is in the edit or view mode:

Widget menu	Options
In dashboard edit mode: 	*Copy* - copy the widget *Paste* - paste a copied widget over this widget This option is grayed out if no widget has been copied. *Delete* - delete the widget
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

Copying/pasting widgets

Dashboard widgets can be copied and pasted. 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. Then the copied widget can be used to create a new widget with the same properties:

- using the *Paste widget* button when editing the dashboard
- using the *Paste widget* option when adding a new widget by selecting some area in the dashboard (a widget must 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.

Dynamic widgets

When configuring some of the widgets:

- Classic graph
- Graph prototype
- 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 to select 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.

Viewing and editing a dashboard

When viewing a single dashboard, the following options are available:

Edit dashboard		Switch to the dashboard editing mode.
		Open the action menu (see description of the available actions below).
	Sharing	Edit sharing preferences for the dashboard. 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 to other users and user groups. For details on configuring sharing, see the map configuration section.
	Create new	Create a new dashboard. First you are prompted to enter general properties of the new dashboard - owner and name. Then, the new dashboard opens in editing mode and you can add widgets.
	Clone	Create a new dashboard by copying properties of the existing one. First you are prompted to enter general properties of the new dashboard - owner and name. Then, the new dashboard opens in editing mode with all the widgets of the original dashboard.
	Delete	Delete the dashboard.
		Display only page content (kiosk mode). Kiosk mode can also be accessed with the following URL parameters: `/zabbix.php?action=dashboard.view&kiosk=` To exit to normal mode: `/zabbix.php?action=dashboard.view&kiosk=`

Editing mode is opened:

- when a new dashboard is being created
- when you click on the edit button of a widget
- when you click the *Edit dashboard* button for an existing dashboard

In the dashboard editing mode the following options are available:

	Edit general dashboard properties - name and owner.
	Add a new widget.
	Paste a copied widget. This button is grayed out if no widget has been copied.
	Save dashboard changes.
	Cancel dashboard changes.

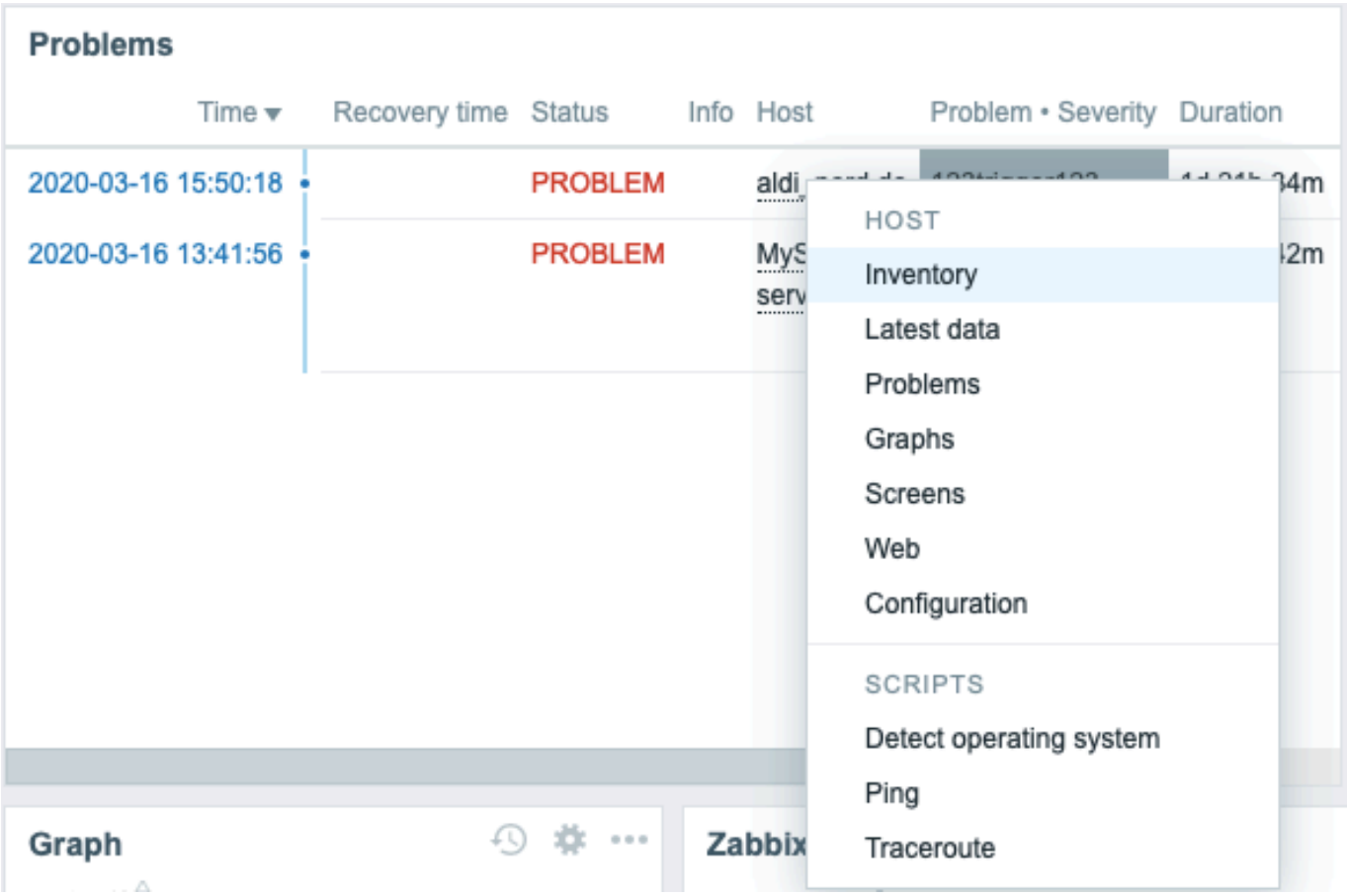
Permissions to dashboards

Permissions to dashboards for regular and Zabbix Admin users 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 custom scripts, inventory, latest data, problems, graphs, screens, web scenarios and configuration for the host. Note that host configuration is available for Admin and Superadmin users only.



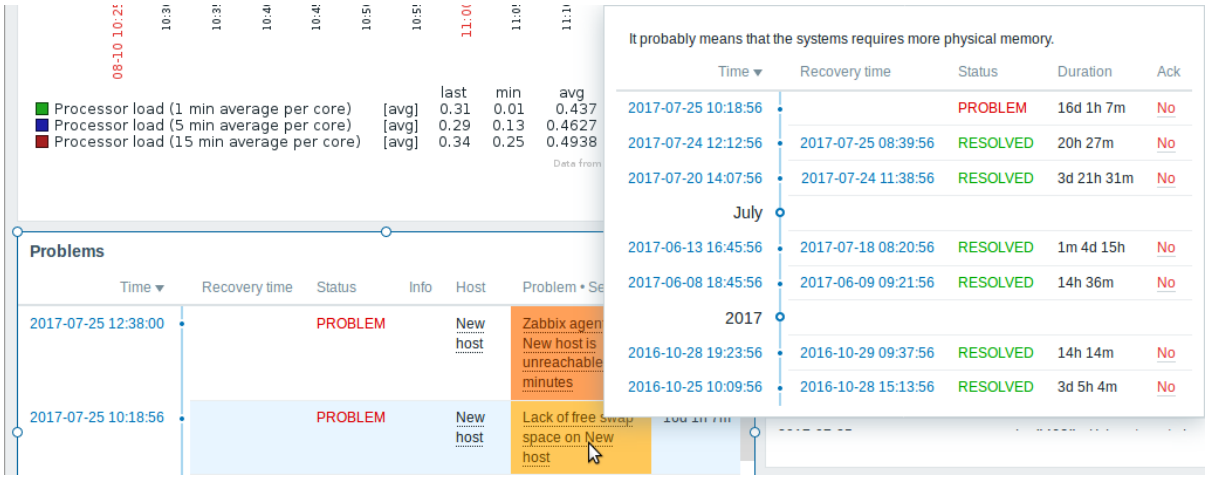
The host menu is accessible by clicking on a host in several other frontend sections:

- Monitoring → Problems
- Monitoring → Problems → Event details
- Monitoring → Hosts
- Monitoring → Hosts → Web Monitoring (since 5.0.4)
- Monitoring → Overview (on *Hosts: left*)
- Monitoring → Latest data

- Monitoring → Screens (in *Host issues* and *Host group issues* widgets)
- 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 name in the *Problem-Severity* column of the *Problems* widget. The popup disappears once you remove the mouse from the problem name.
- click on the problem name in the *Problem-Severity* column of the *Problems* widget. The popup disappears only if you click on the problem name again.

1 Dashboard widgets

Overview

This section lists available dashboard widgets and provides details for widget configuration.

The following parameters are common for every single widget:

Name	Enter a widget name.
Refresh interval	Configure default refresh interval. Default refresh intervals for widgets range from *No refresh* to *15 minutes* depending on the type of widget. For example: *No refresh* for URL widget, *1 minute* for action log widget, *15 minutes* for clock widget.
Show header	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.

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:

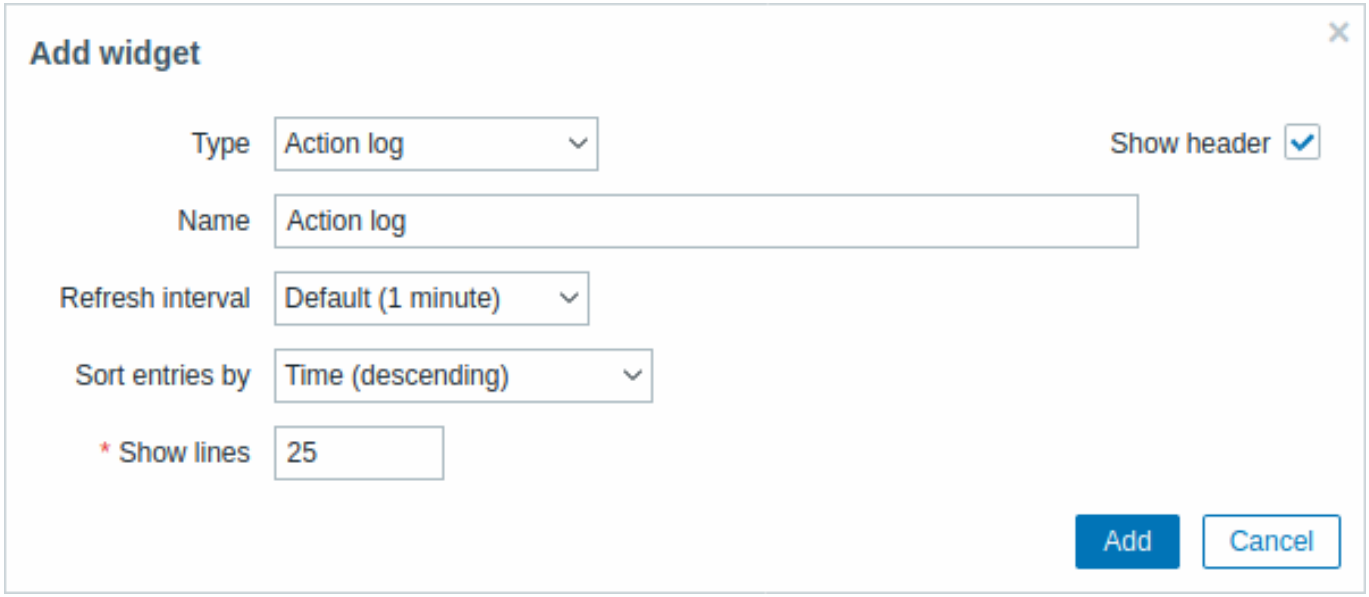
- To set a default value for all the corresponding users switch to editing mode (click the *Edit dashboard* button, find the right widget, click the *Edit* button opening the editing form of a widget) and choose the required refresh interval from the dropdown list.
- Setting a unique refresh interval 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.

Action log

In the action log widget you can display details of action operations (notifications, remote commands). It replicates information from *Reports* → *Action log*.

To configure, select *Action log* as type:



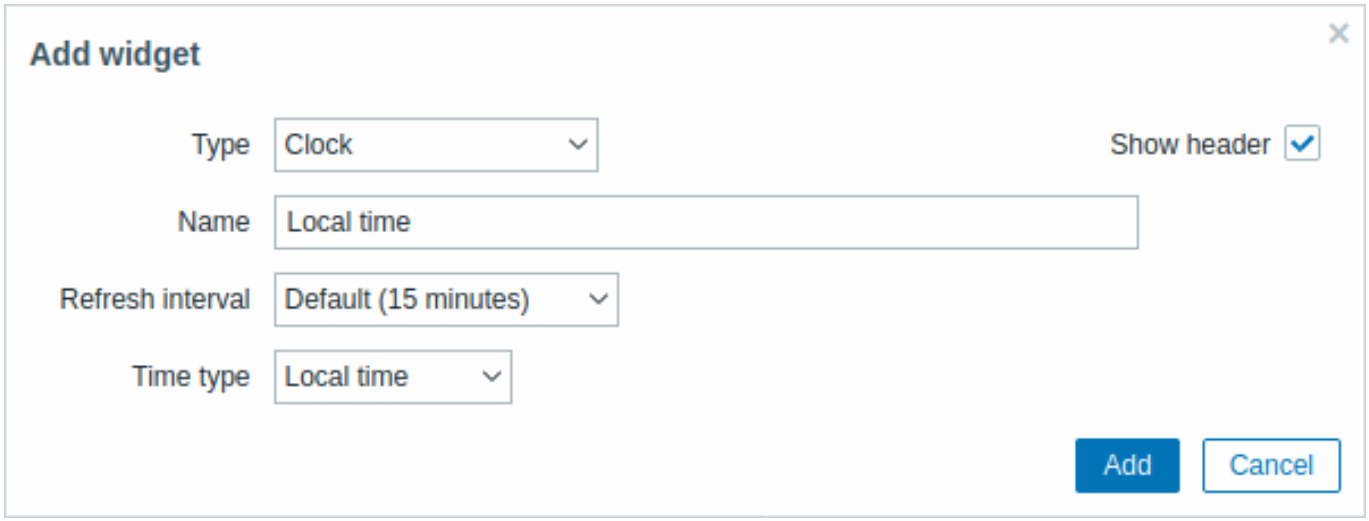
You may set the following specific options:

Sort entries by	Sort entries by: **Time** (descending or ascending) **Type** (descending or ascending) **Status** (descending or ascending) **Recipient** (descending or ascending).
Show lines	Set how many action log lines will be displayed in the widget.

Clock

In the clock widget you may display local, server or specified host time.

To configure, select *Clock* as type:



You may set the following specific options:

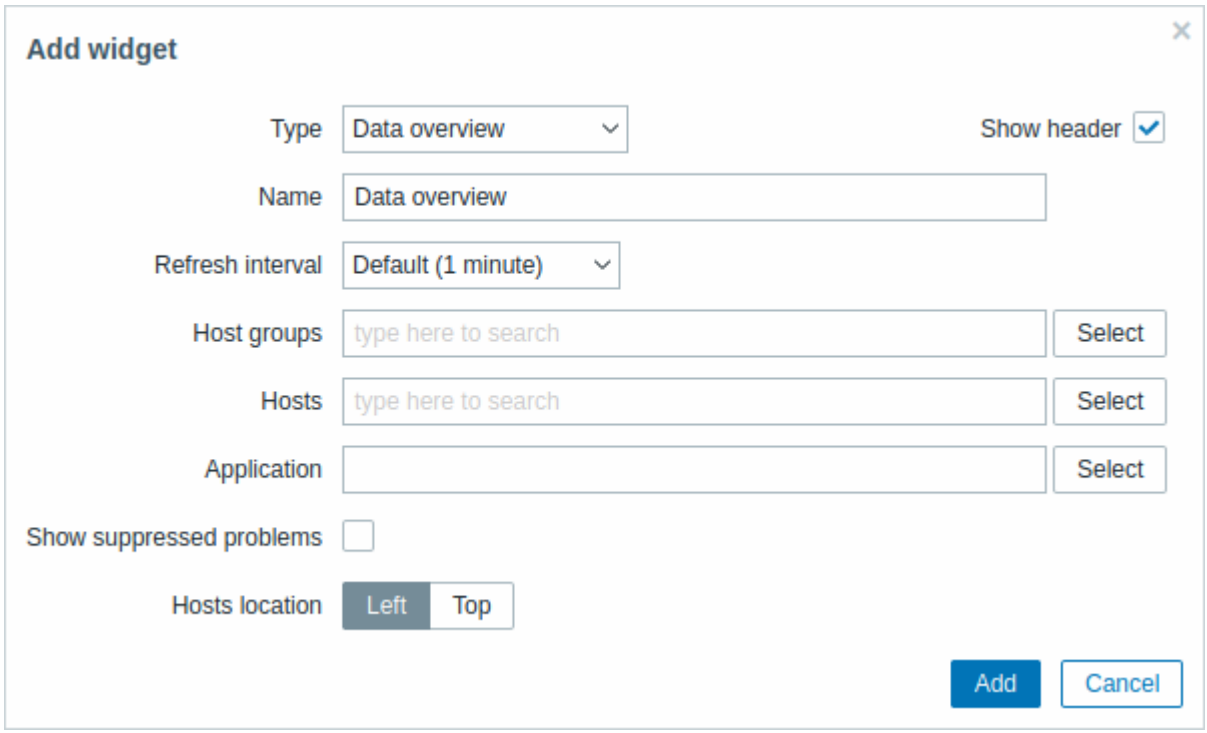
Time type	Select local, server or specified host time.
Item	Select the item for displaying time. To display host time, use the `system.localtime[local]` item. This item must exist on the host. This field is available only when *Host time* is selected.

Data overview

In the data overview widget you can display the latest data for a group of hosts. It replicates information from *Monitoring* → *Overview* (when viewing *Data overview*).

Note that there is a hard-coded limit of 50 records displayed. There is no pagination. If more records exist, a message is displayed at the bottom of the table, asking to provide more specific filtering criteria. Note that this limit is applied first, before any further filtering of data by parameter.

To configure, select *Data overview* as type:

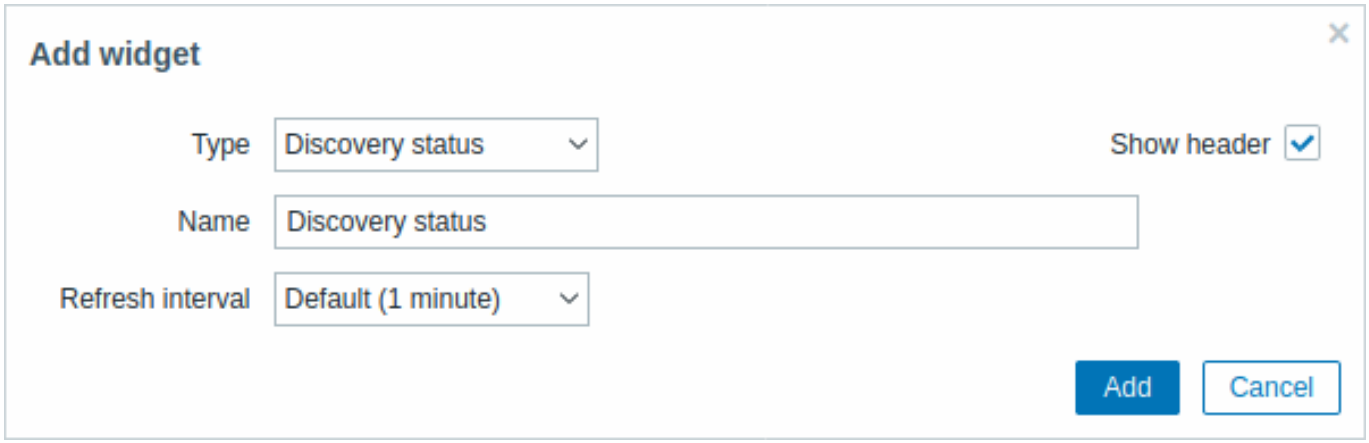


You may set the following specific options:

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.
Application	Enter application name.
Show suppressed problems	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.
Hosts location	Select host location - left or top.

Discovery status

This widget displays a status summary of the active network discovery rules.



Favorite graphs

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.

Favorite maps

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.

Favorite screens

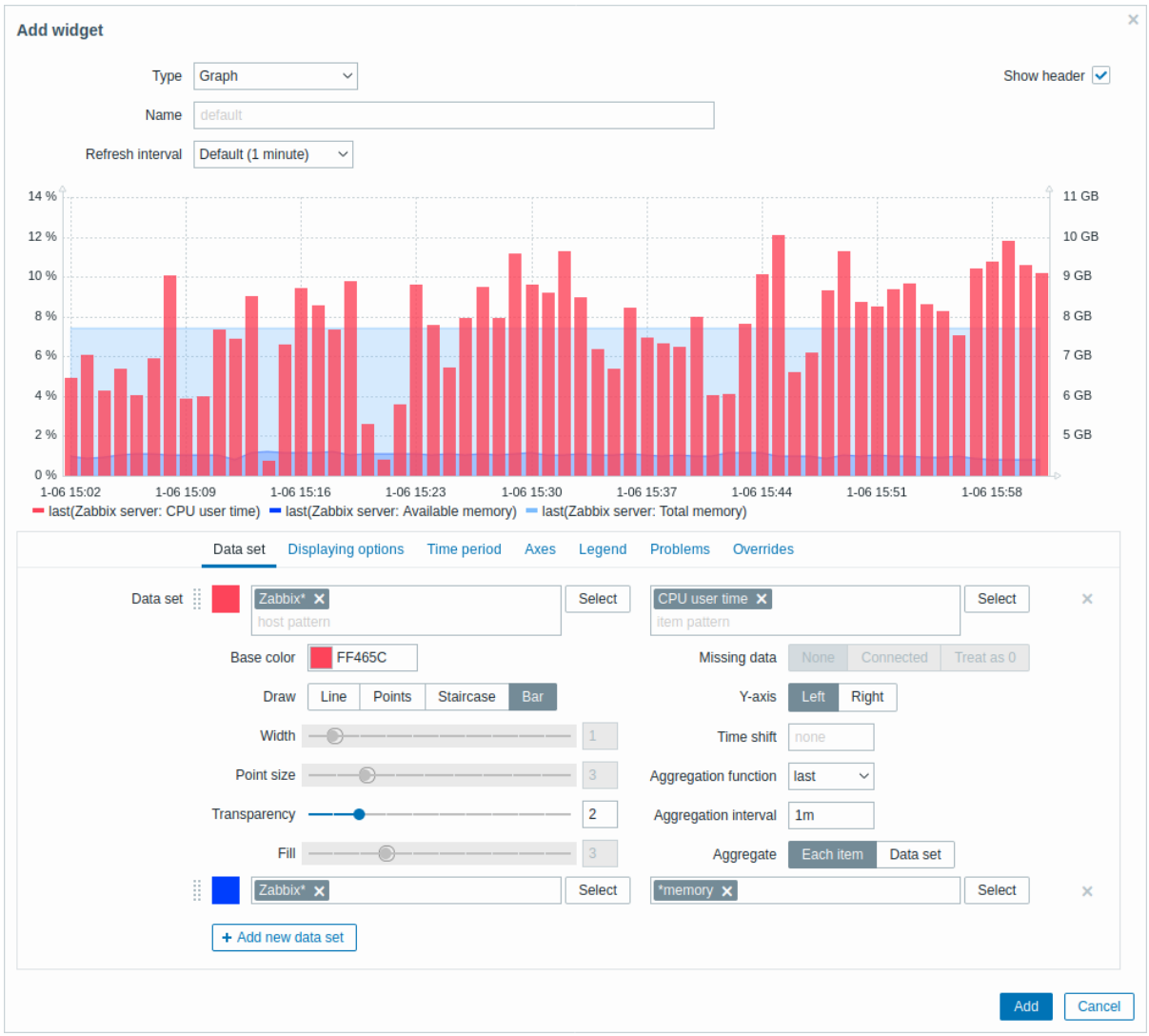
This widget contains shortcuts to the most needed screens and slide shows, sorted alphabetically.

The list of shortcuts is populated when you view a screen/slide show and then click on its *Add to favorites* button.

Graph

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).

To configure, select *Graph* as type:



The **Data set** tab allows to add data sets and define their visual representation:

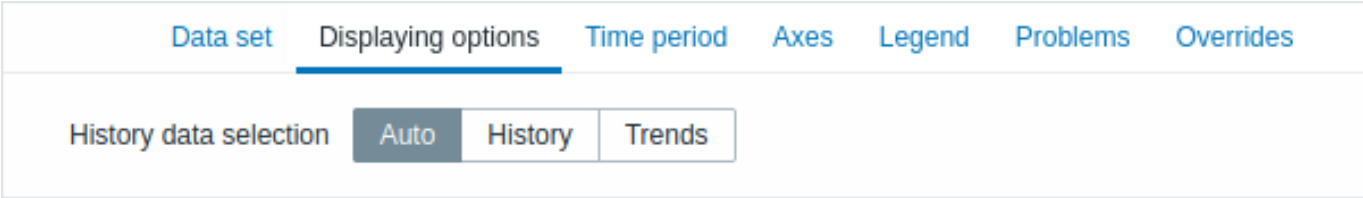
Data set		Select hosts and items to display on the graph. Alternatively you may enter host and item patterns. Wildcard patterns may be used (for example, * will return results that match zero or more characters). To specify a wildcard pattern, just enter the string manually and press *Enter*. While you are typing, note how all matching hosts are displayed in the dropdown. Up to 50 items may be displayed in the graph. Host pattern and item pattern fields are mandatory. *Note* that item names containing deprecated positional macros ($1-$9) are not resolved in this field. Therefore it will not be possible to add any of the items named like `CPU $2 time` individually to the graph using its resolved name (like `CPU user time`); while using its unresolved name `CPU $2 time` will add all corresponding items to the graph (e.g. `CPU user time`, `CPU system time`, `CPU idle time`, etc.). 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).
	Base color	Adjust base color, either from the color picker or manually. Base color is used to calculate different colors for each item of the data set. Base color input field is mandatory.
	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 line/staircase graph it is drawn as point regardless of draw type. The point size is calculated from line width, but it cannot be smaller than 3 pixels, even if line width is less.
	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 Not applicable for the *Points* and *Bar* draw type.
	Y-axis	Select the side of the graph where 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. This option is supported since Zabbix 4.4.
	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. This option is supported since Zabbix 4.4.
	Aggregate	Specify whether to aggregate: **Each item** - each item in the dataset will be aggregated and displayed separately. **Data set** - all dataset items will be aggregated and displayed as one value. This option is supported since Zabbix 4.4.

Existing data sets are displayed in a list. You may:

- 

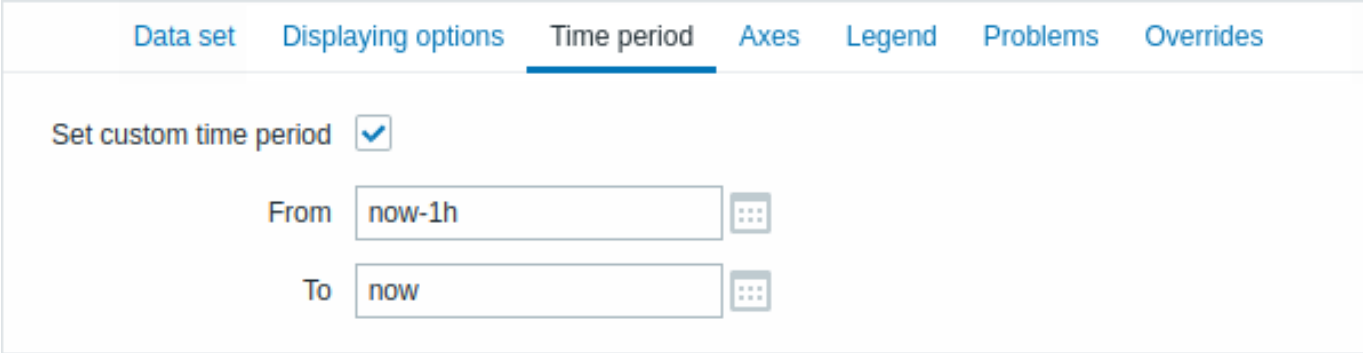
- click on this button to add a new data set
- - click on the color icon to expand/collapse data set details
-  - click on the move icon and drag a data set to a new place in the list

The **Displaying options** tab allows to define history data selection:



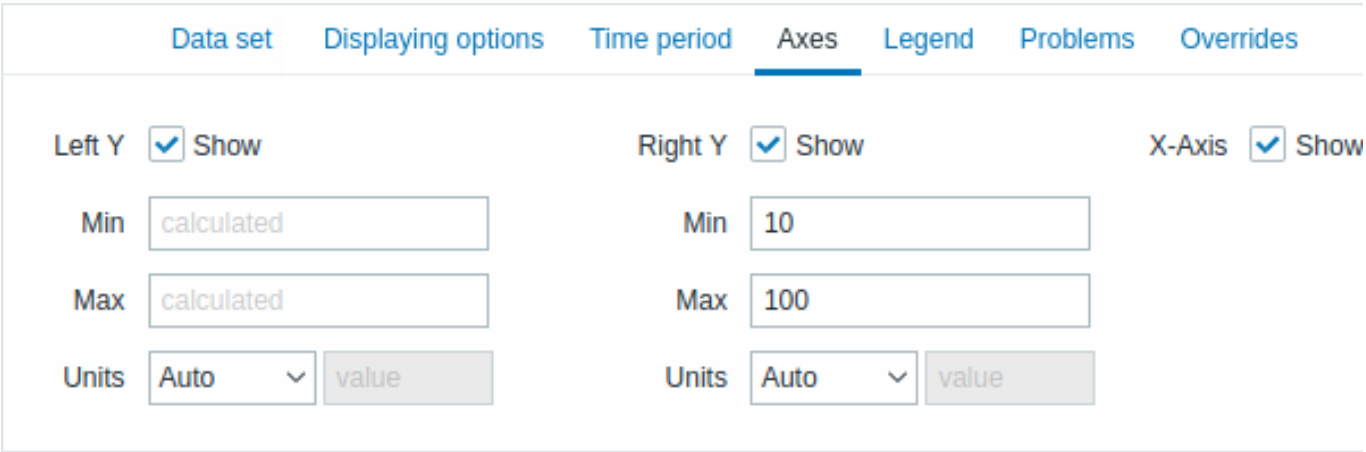
History data selection	Set the source of graph data: **Auto** - data are sourced according to the classic graph algorithm (default) **History** - data from history **Trends** - data from trends

The **Time period** tab allows to set a custom time period:



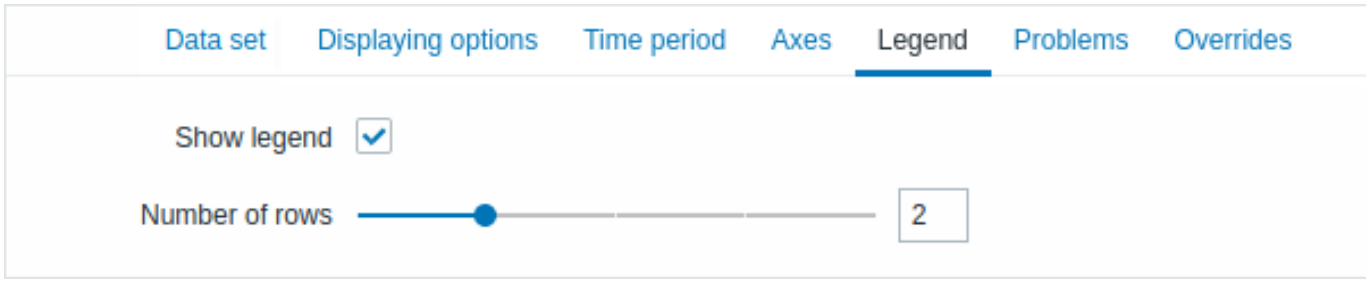
Set custom time period	Mark this checkbox to set custom time period for the graph (unmarked by default).
From	Set the start time of the custom time period for the graph.
To	Set the end time of the custom time period for the graph.

The **Axes** tab allows to customize how axes are displayed:



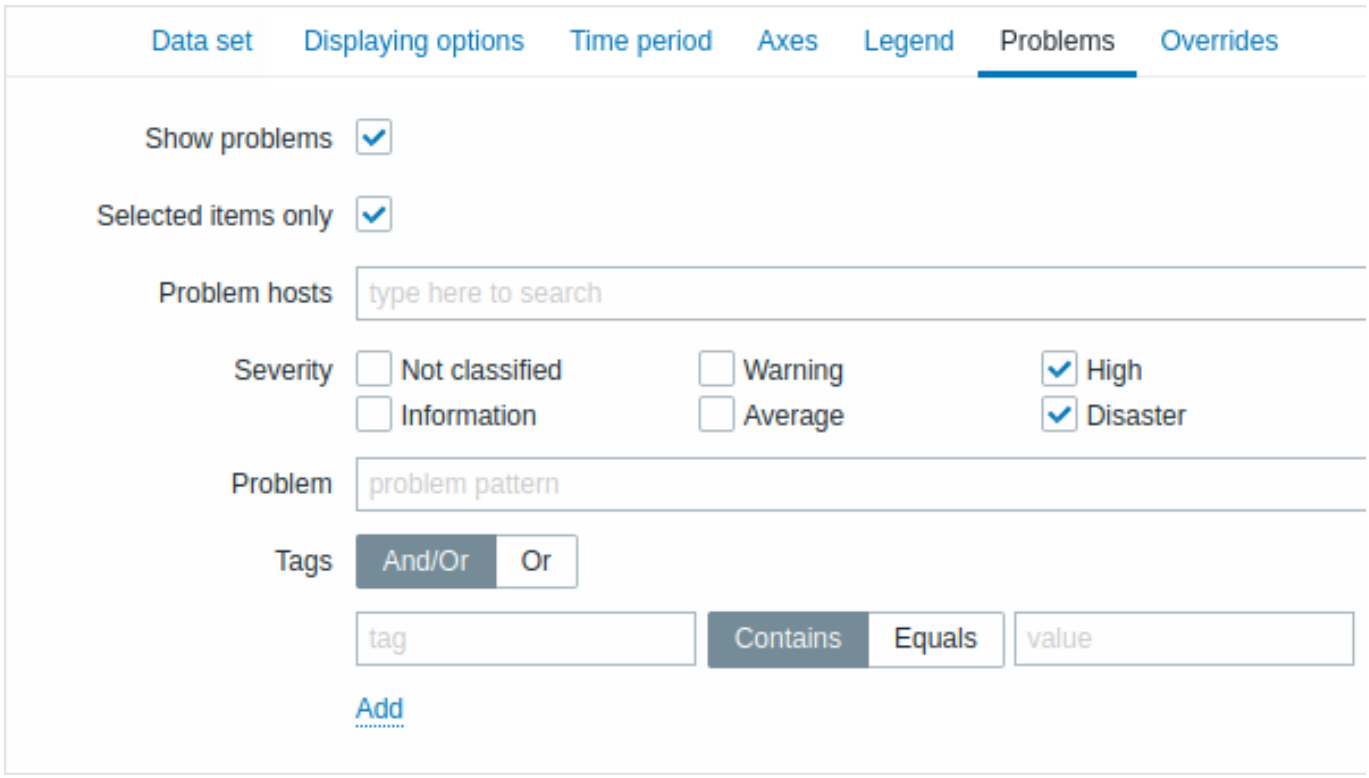
Left Y	Mark this checkbox to make left Y axis visible. The checkbox may be disabled if unselected either in *Data set* or in *Overrides* tab.
Right Y	Mark this checkbox to make right Y axis visible. The checkbox may be disabled if unselected either in *Data set* or in *Overrides* tab.
X-Axis	Unmark this 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 *Auto* option is chosen axis values are displayed using units of the first item of corresponding axis. *Static* option allows you to assign corresponding axis' custom name. If *Static* option is chosen and *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:



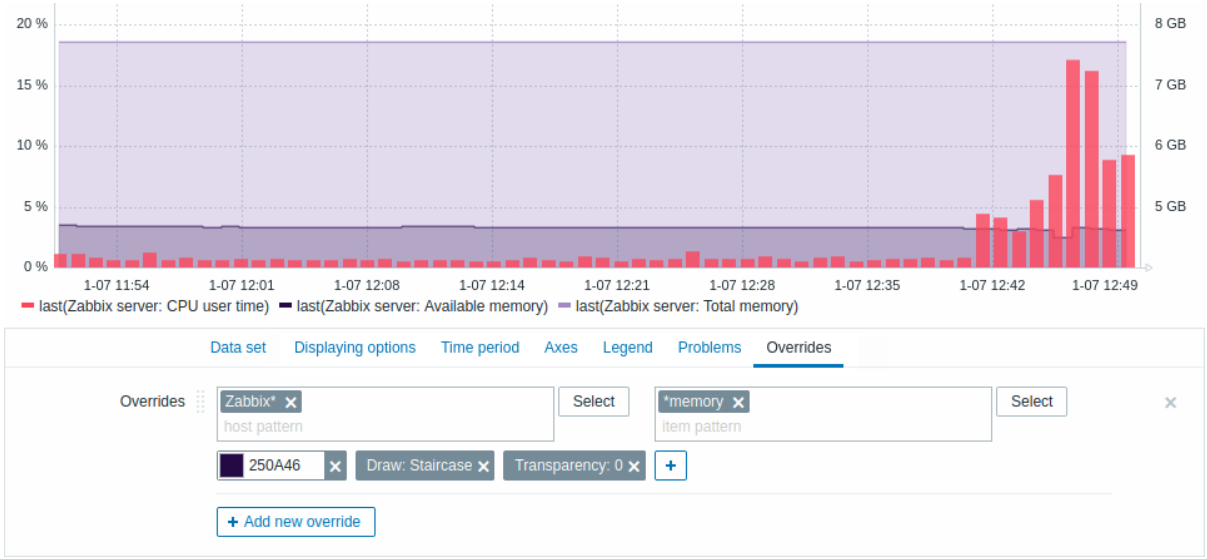
Show legend	Unmark this checkbox to hide the legend on the graph (marked by default).
Number of rows	Set the number of rows to be displayed on the graph.

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 tag name and value to limit the number of problems displayed on the graph. To add more tag names and values, click on *Add*. There are two calculation types for several conditions: **And/Or** - all conditions must be met, conditions having same tag name will be grouped by Or condition **Or** - enough if one condition is met 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).

The **Overrides** tab allows to add custom overrides for data sets:

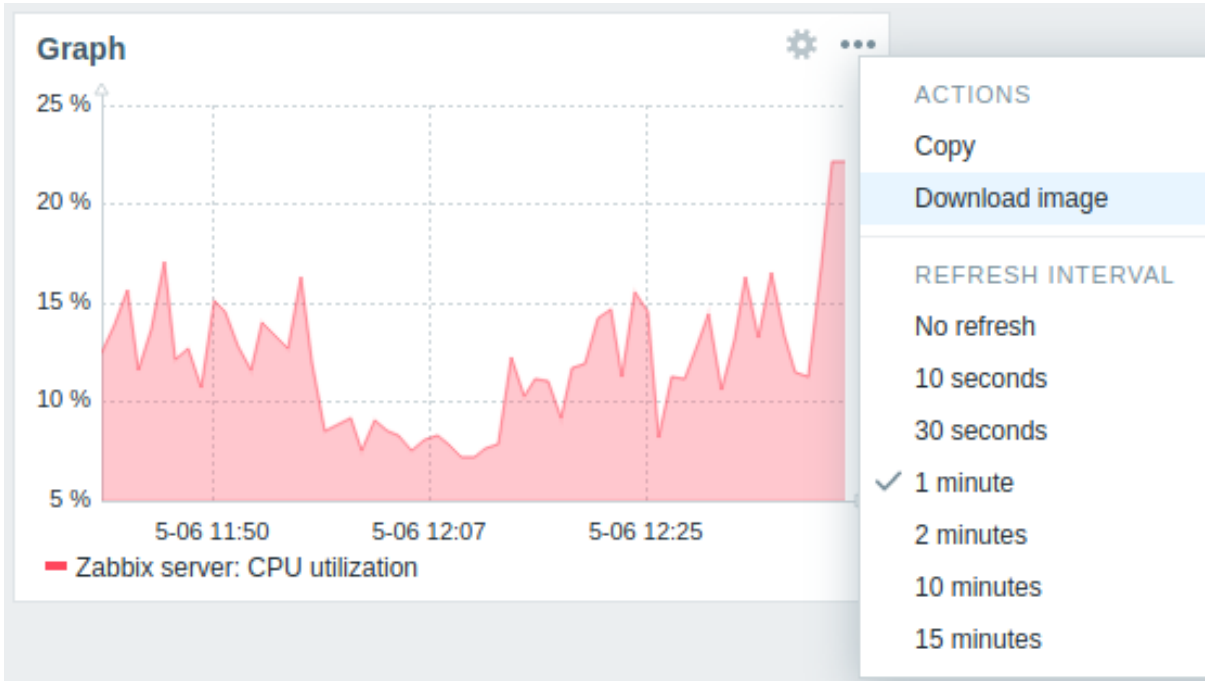


Overrides are useful when several items are selected for a data set using the * wildcard and you want to change how the items are displayed by default (e.g. default base color or any other property).

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:

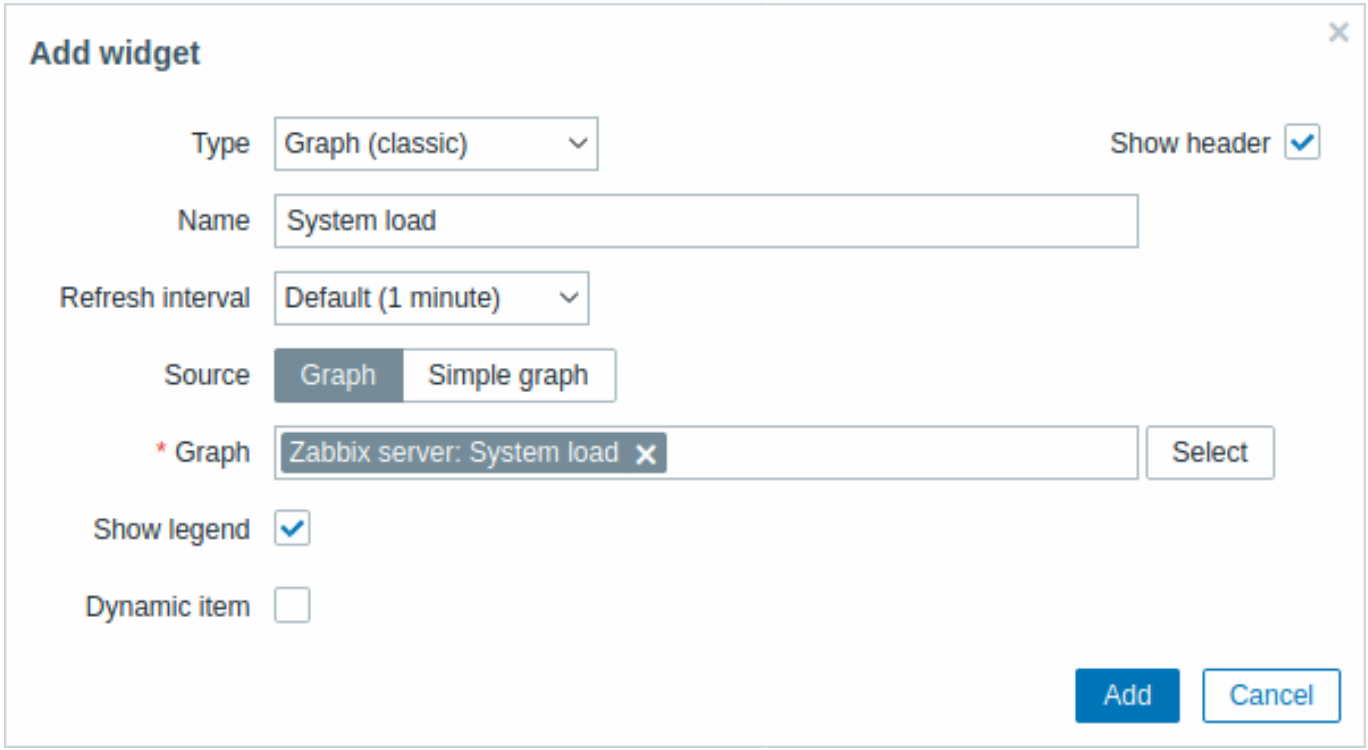


A screenshot of the widget will be saved to the Downloads folder.

Graph (classic)

In the classic graph widget you can display a single custom graph or simple graph.

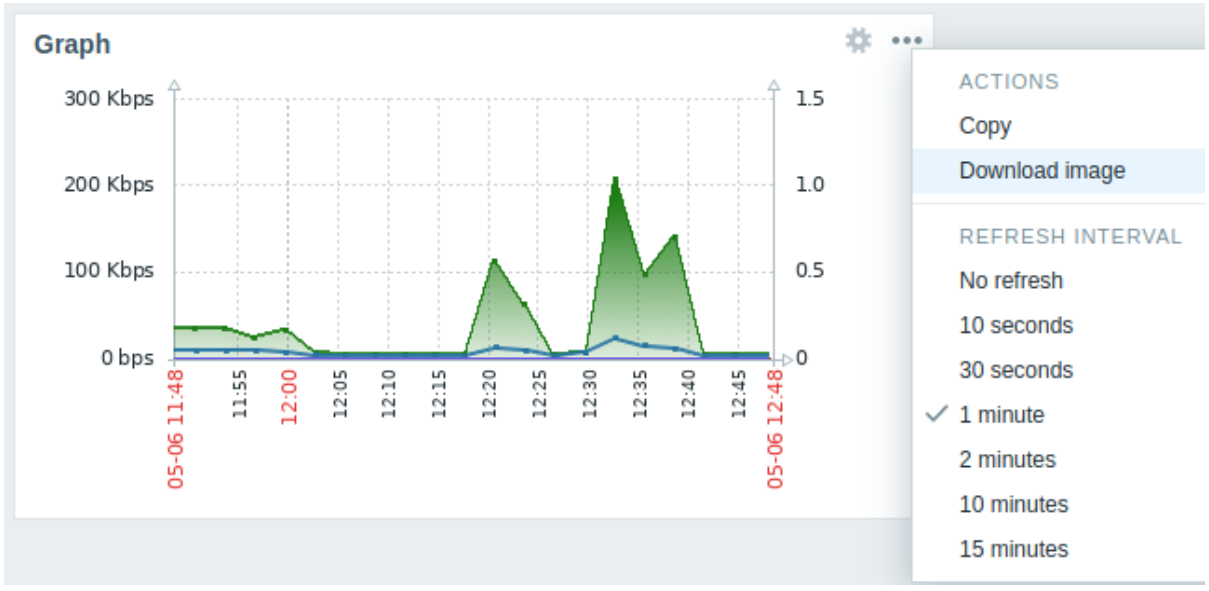
To configure, select *Graph* as type:



You may set the following specific options:

Source	Select graph type: **Graph** - custom graph **Simple graph** - simple graph
Graph	Select the custom graph to display. This option is available if 'Graph' is selected as *Source*.
Item	Select the item to display in a simple graph. This option is available if 'Simple graph' is selected as *Source*.
Show legend	Unmark this checkbox to hide the legend on the graph (marked by default).
Dynamic item	Set graph to display different data depending on the selected host.

Information displayed by the classic graph widget can be downloaded as .png image using the widget menu:

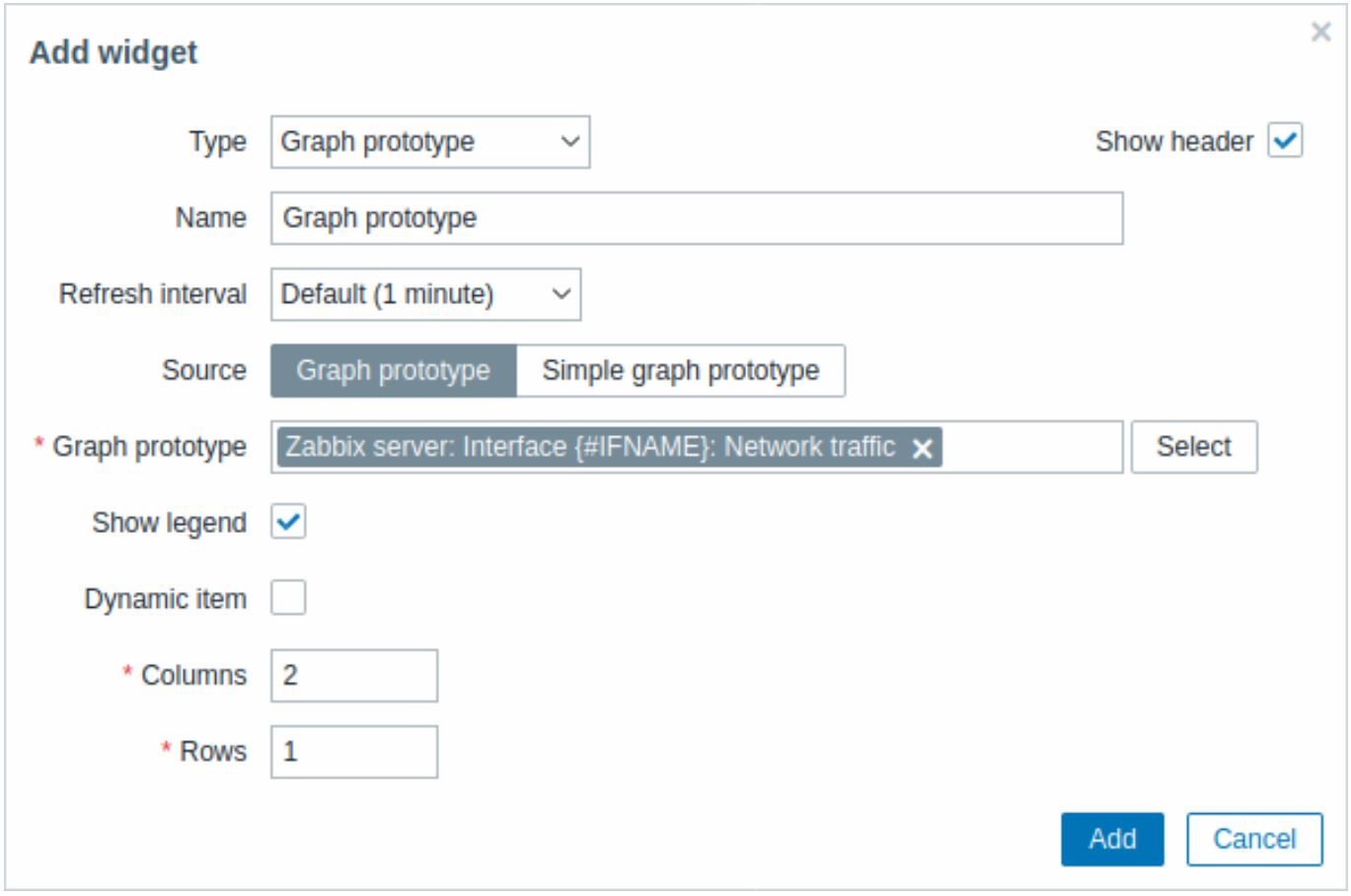


A screenshot of the widget will be saved to the Downloads folder.

Graph prototype

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.

To configure, select *Graph prototype* as widget type:



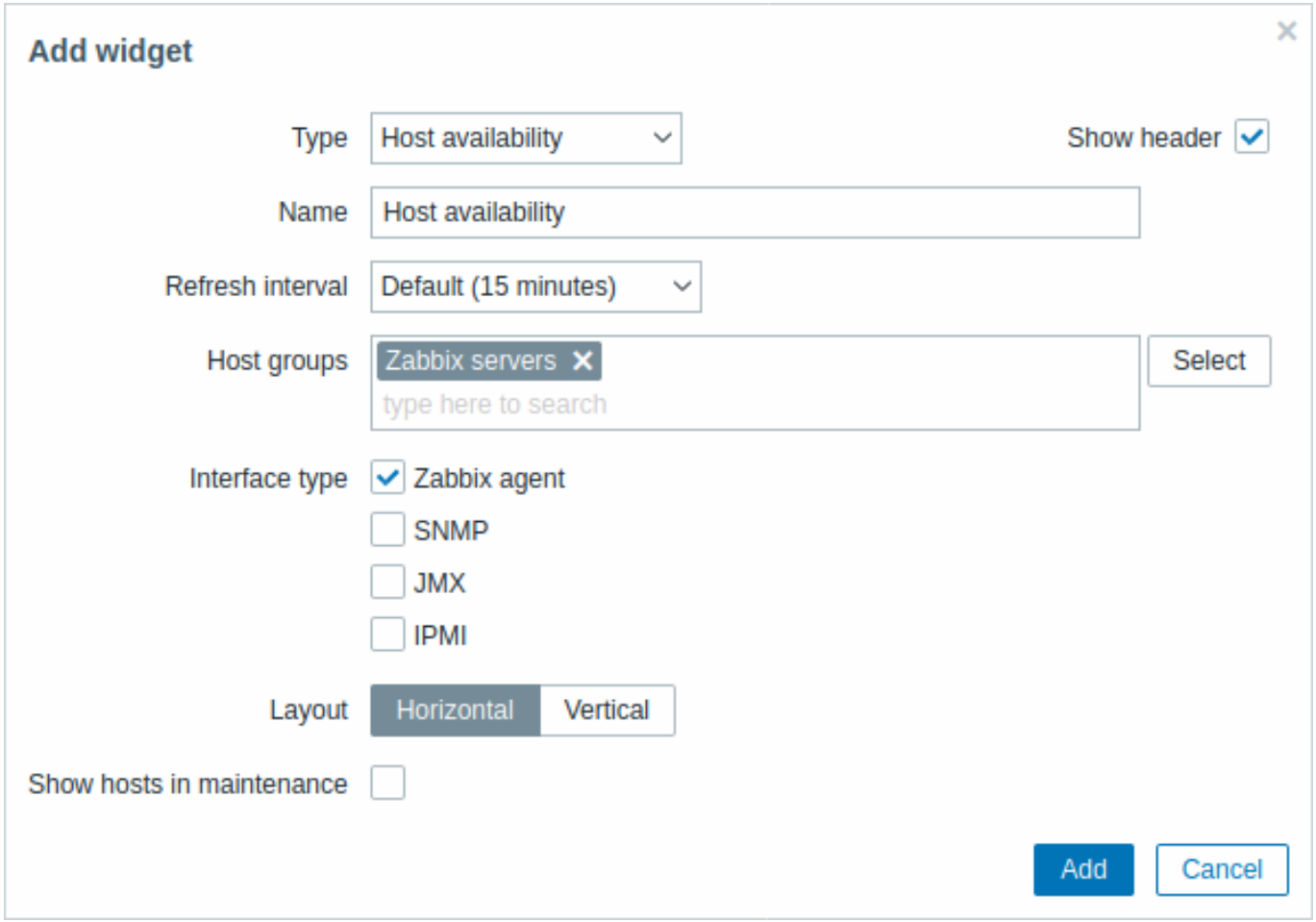
You may set the following specific options:

Source	Select source: either a **Graph prototype** or a **Simple graph prototype**.
Graph prototype	Select a graph prototype to display discovered graphs of the graph prototype. This option is available if 'Graph prototype' is selected as Source.
Item prototype	Select an item prototype to display simple graphs based on discovered items of an item prototype. This option is available if 'Simple graph prototype' is selected as Source.
Show legend	Mark this checkbox to show the legend on the graphs (marked by default).
Dynamic item	Set graphs to display different data depending on the selected host.
Columns	Enter number of columns of graphs to display within a graph prototype widget.
Rows	Enter number of rows of graphs to display within a graph prototype widget.

Host availability

In the host availability widget you can display high-level statistics about host availability.

To configure, select *Host availability* as type:



You may set the following specific options:

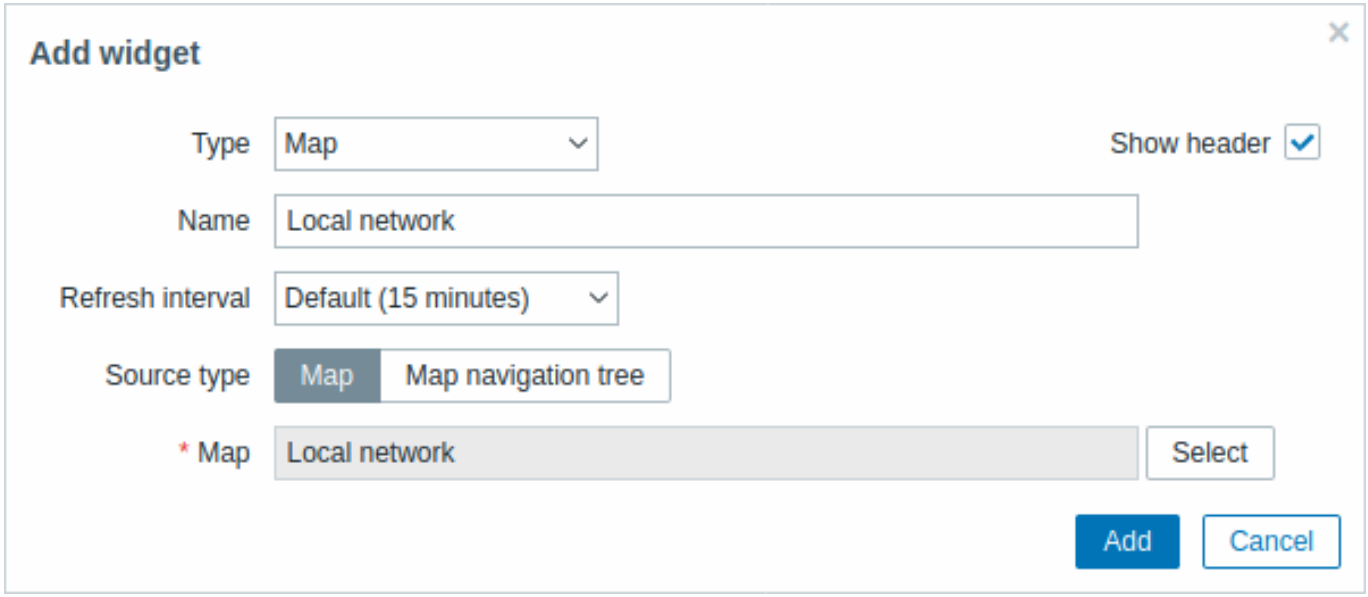
Host groups	Select host group(s). This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Scroll down to select. Click on 'x' to remove the selected.
Interface type	Select which host interfaces you want to see availability data for. Availability of all interfaces is displayed by default, if nothing is selected.
Layout	Select vertical or horizontal display.
Show hosts in maintenance	Include hosts that are in maintenance in the statistics.

Map

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).

To configure, select *Map* as type:



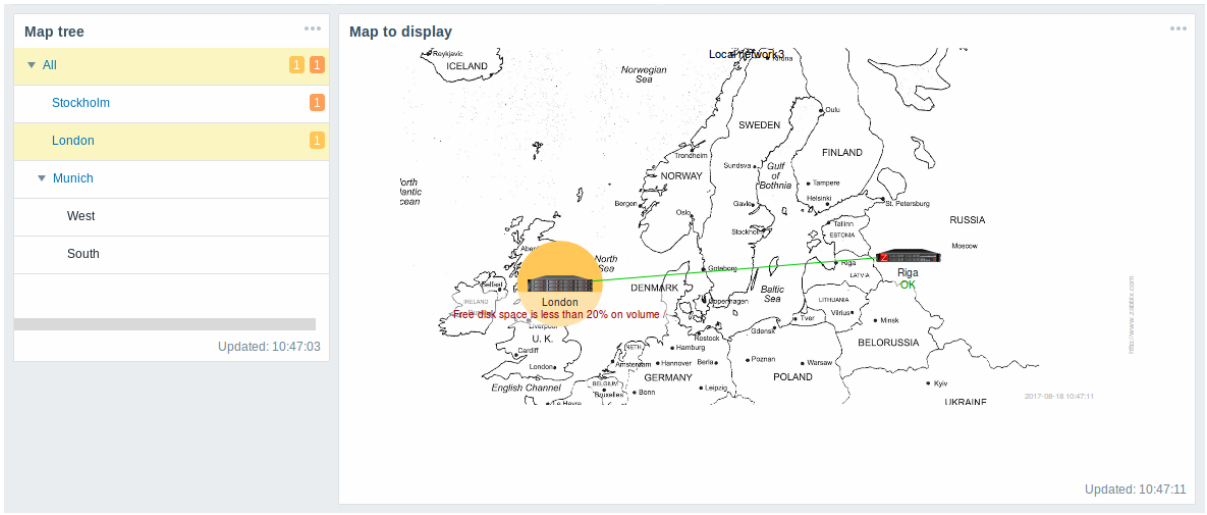
You may set the following specific options:

Source type	Select to display: **Map** - network map **Map navigation tree** - one of the maps in the selected map navigation tree
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*.

Map navigation tree

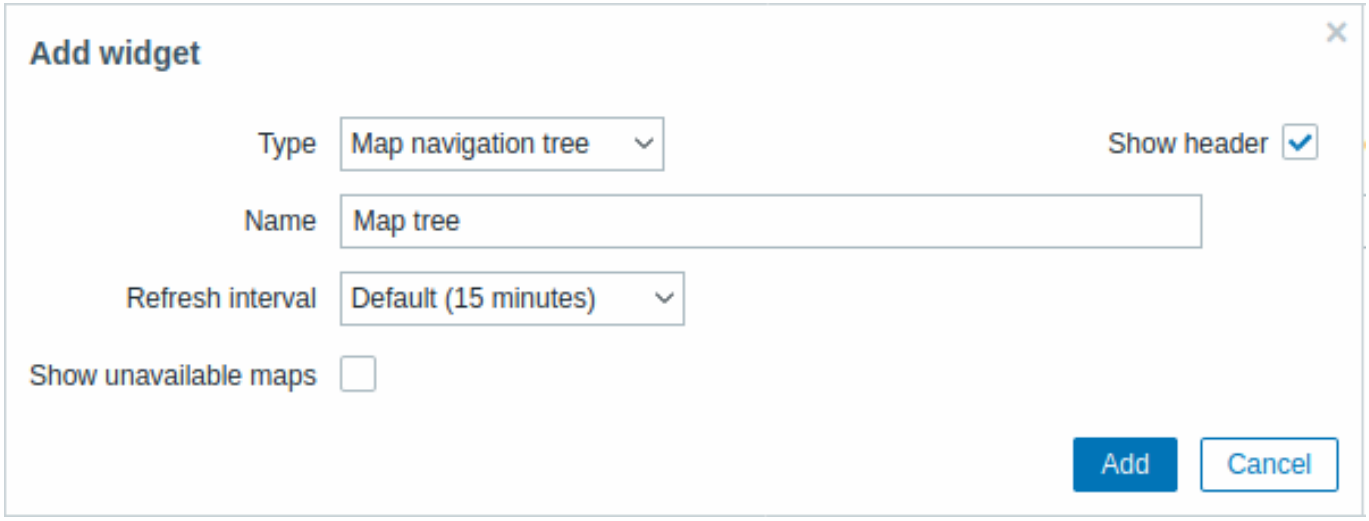
This widget allows to build a hierarchy of existing maps while also displaying problem statistics with each included map and map group.

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 submaps and its own problems.

To configure the navigation tree widget, select *Map navigation tree* as type:



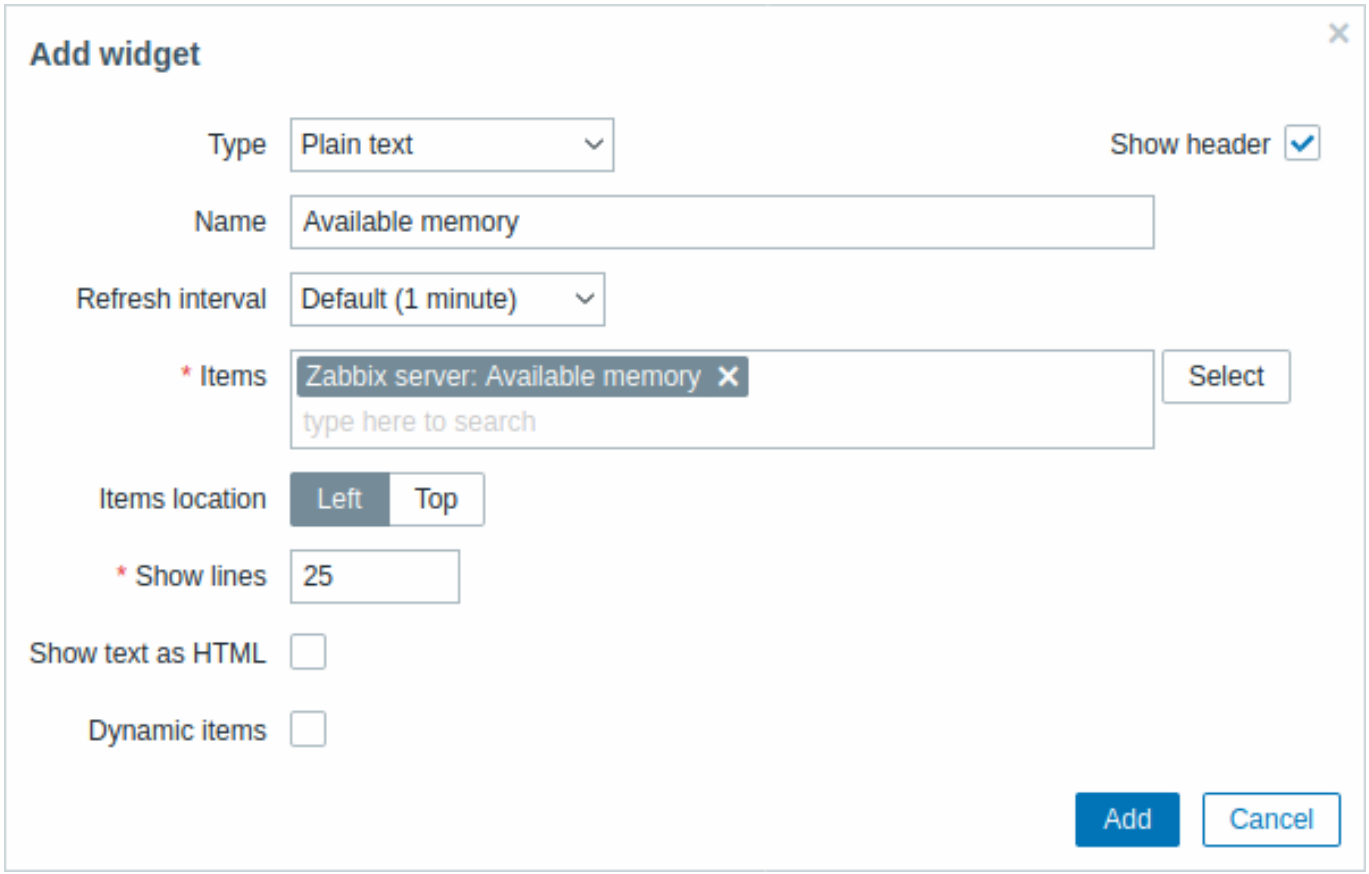
You may set the following specific options:

Show unavailable maps	Mark this checkbox to display maps that the user does not have read permission to. Unavailable maps in the navigation tree will be displayed with a grayed out icon. Note that if this checkbbox is marked, available submaps are displayed even if the parent level map is unavailbale. If unmarked, available submaps to an unavailable parent map will not be displayed at all. Problem count is calculated based on available maps and available map elements.

Plain text

In the plain text widget you can display latest item data in plain text.

To configure, select *Plain text* as type:



You may set the following specific options:

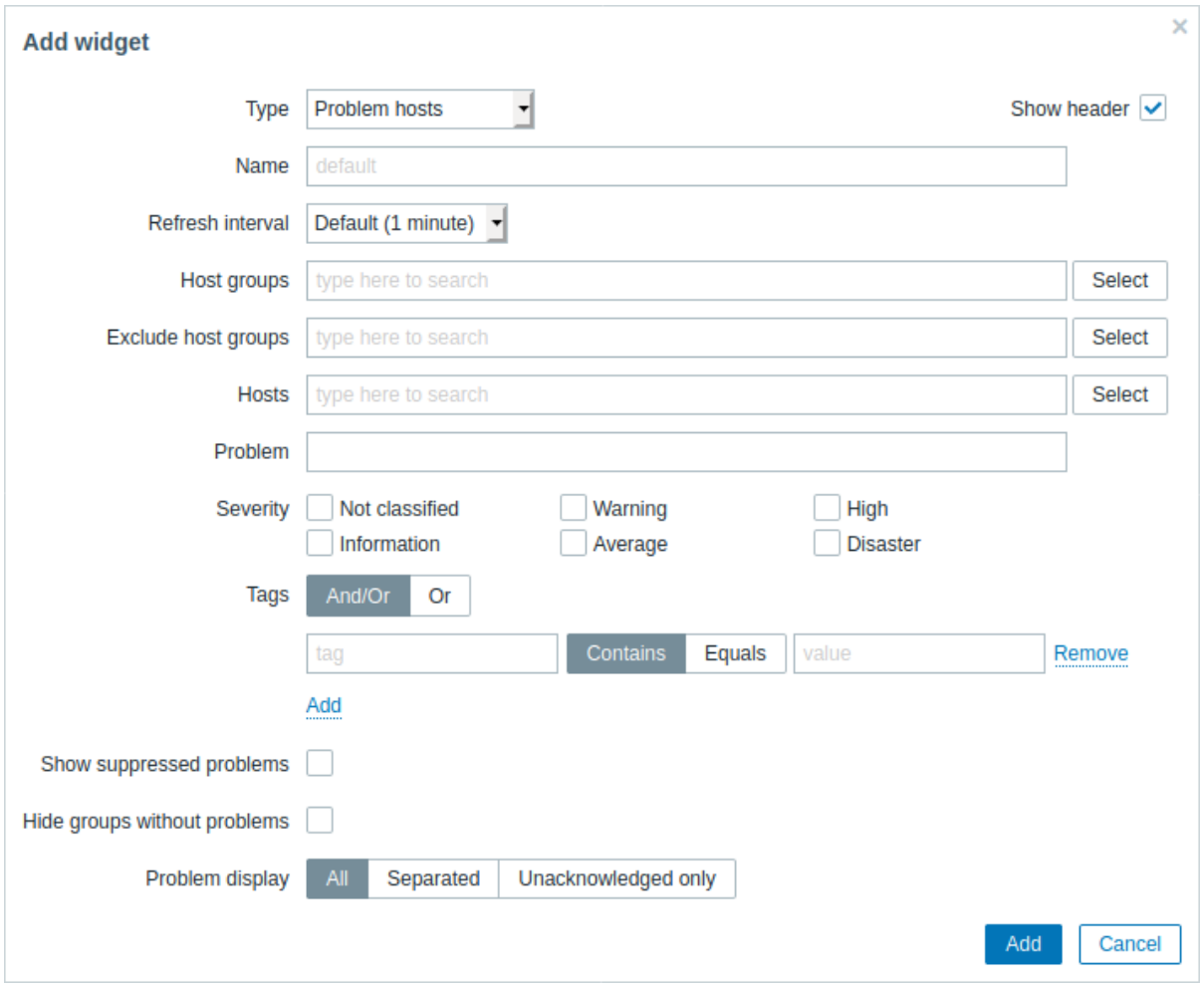
Items	Select the items.
Items location	Choose the location of selected items to be displayed in the widget.

Show lines	Set how many latest data lines will be displayed in the widget.
Show text as HTML	Set to display text as HTML.
Dynamic item	Set to display different data depending on the selected host.

Problem hosts

In the host information widget you can display high-level information about host availability.

To configure, select *Problem hosts* as type:



You may set the following specific options:

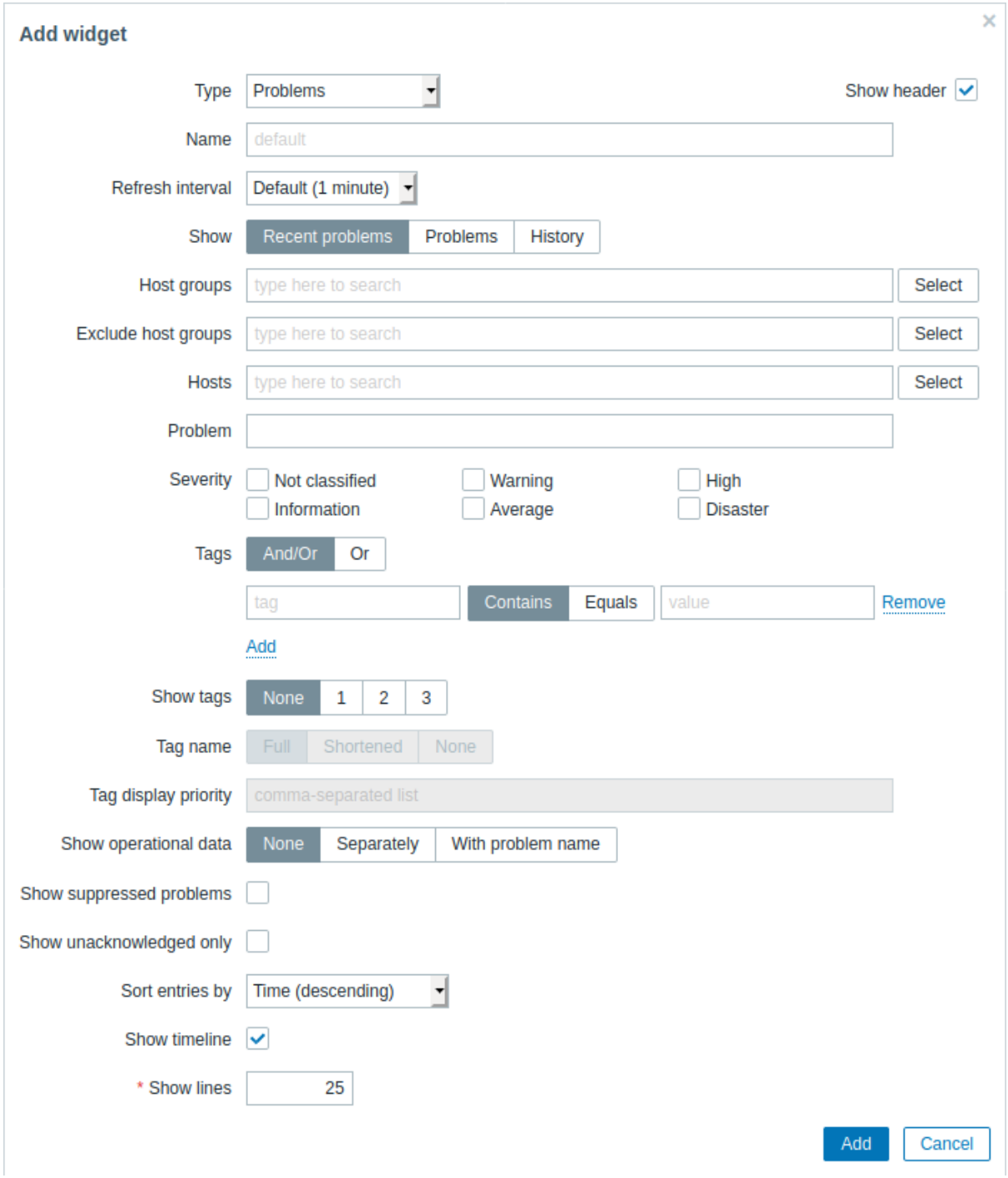
Host groups	Enter host groups to display in the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will be displayed in the widget. If no host groups are entered, all host groups will be displayed.
Exclude host groups	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.
Hosts	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.
Problem	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.
Severity	Mark the problem severities to be displayed in the widget.

Tags	Specify tag name and value to limit the number of problems displayed on the graph. To add more tag names and values, click on *Add*. There are two calculation types for several conditions: **And/Or** - all conditions must be met, conditions having same tag name will be grouped by Or condition **Or** - enough if one condition is met 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).
Show suppressed problems	Mark the checkbox to display problems which 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.

Problems

In this widget you can display current problems. The information in this widget is similar to *Monitoring* → *Problems*.

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.

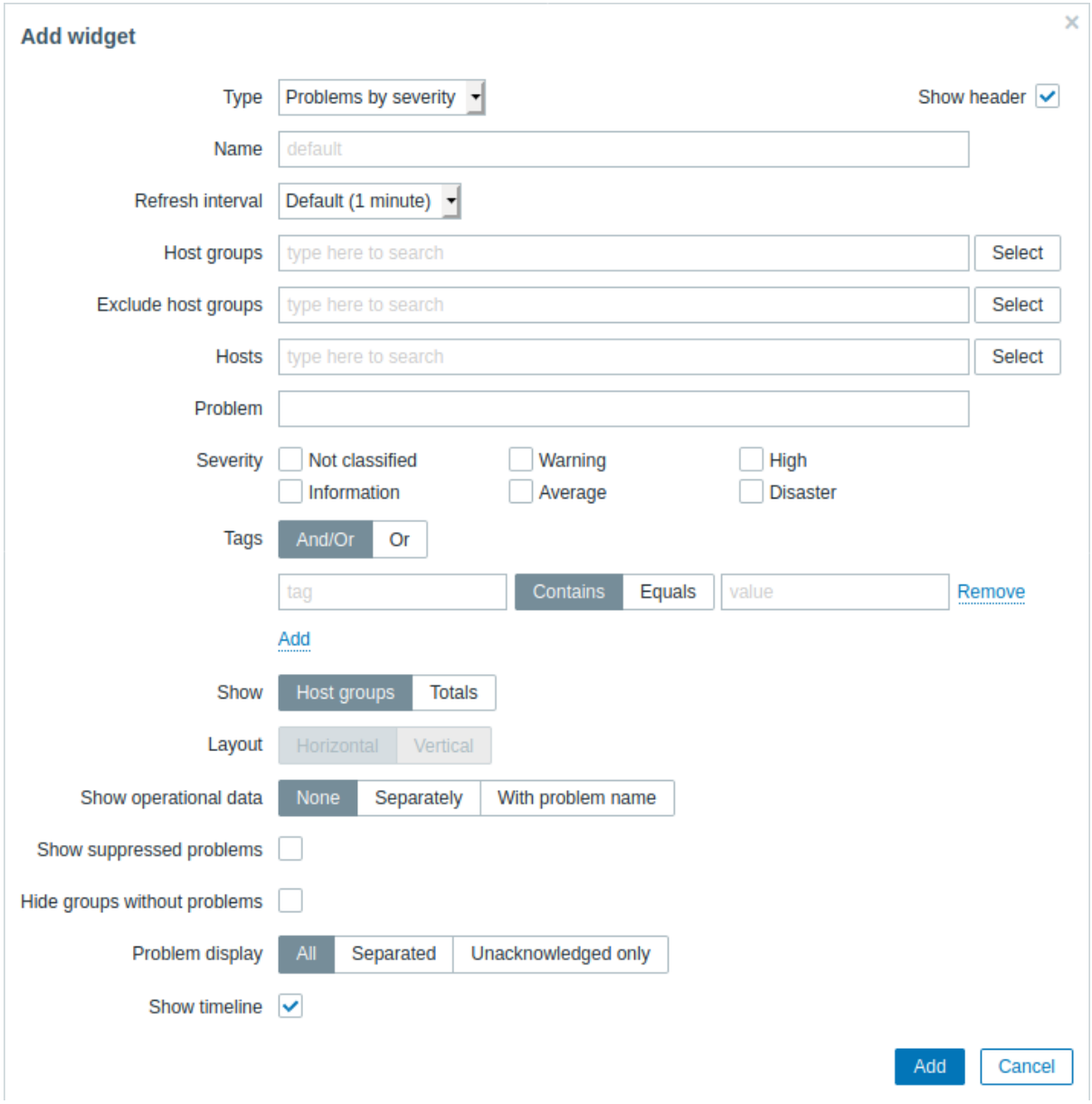
Show	Filter by problem status: **Recent problems** - unresolved and recently resolved problems are displayed (default) **Problems** - unresolved problems are displayed **History** - history of all events is displayed
Host groups	Enter host groups to display problems of in the widget. This field is auto-complete 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.

Exclude host groups	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.
Hosts	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.
Problem	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.
Severity	Mark the problem severities to be displayed in the widget.
Tags	Specify event tag name and value to limit the number of problems displayed. To add more event tag names and values, click on *Add*. There are two calculation types for several conditions: **And/Or** - all conditions must be met, conditions having same tag name will be grouped by Or condition **Or** - enough if one condition is met 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) 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 which would otherwise be suppressed (not shown) because of host maintenance.
Show unacknowledged only	Mark the checkbox to display unacknowledged problems only.
Sort entries by	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.

Problems by severity

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.

To configure, select *Problems by severity* as type:



You may set the following specific options:

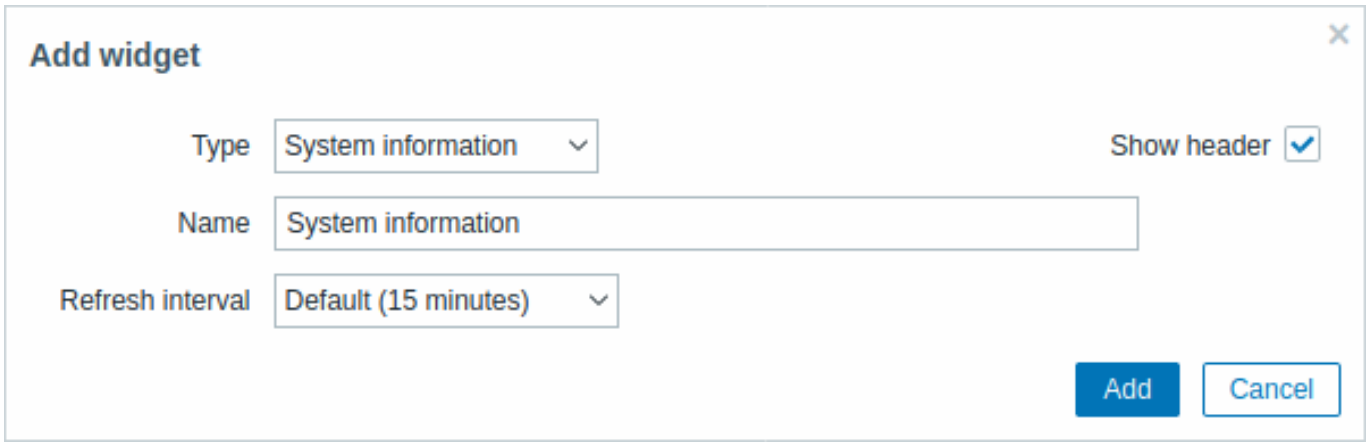
Host groups	Enter host groups to display in the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will be displayed in the widget. If no host groups are entered, all host groups will be displayed.
Exclude host groups	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.
Hosts	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.
Problem	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.
Severity	Mark the problem severities to be displayed in the widget.

Tags	Specify tag name and value to limit the number of problems displayed on the graph. To add more tag names and values, click on *Add*. There are two calculation types for several conditions: **And/Or** - all conditions must be met, conditions having same tag name will be grouped by Or condition **Or** - enough if one condition is met 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).
Show	Select the show option: **Host groups** - display problems per host group **Totals** - display a problem total for all selected host groups in colored blocks corresponding to the problem severity.
Layout	Select the layout option: **Horizontal** - colored blocks of totals will be displayed horizontally **Vertical** - colored blocks of totals will be displayed vertically This field is available for editing if 'Totals' is selected as the *Show* option.
Show suppressed problems	Mark the checkbox to display problems which 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.
Show operational data	Mark the checkbox to display operational data (see description of *Operational data* in *Monitoring → Problems*).
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.
Show timeline	Mark the checkbox to display a visual timeline.

System information

In the System information widget you can display high-level Zabbix and Zabbix server information.

To configure, select *System information* as type:

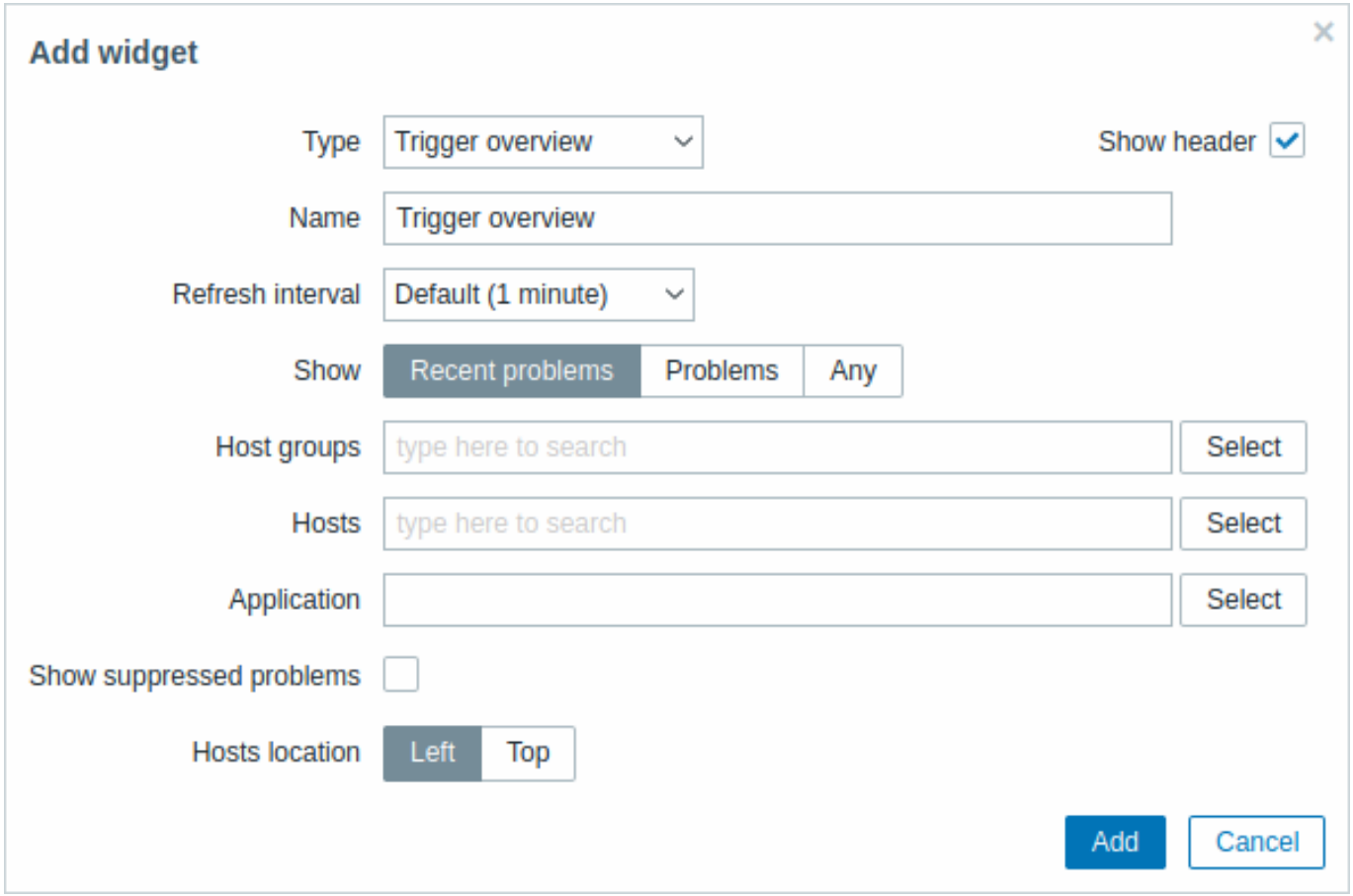


Trigger overview

In the trigger overview widget you can display the trigger states for a group of hosts. It replicates information from *Monitoring → Overview* (when viewing *Trigger overview*).

Note that there is a hard-coded limit of 50 records displayed. There is no pagination. If more records exist, a message is displayed at the bottom of the table, asking to provide more specific filtering criteria. Note that this limit is applied first, before any further filtering of data by parameter.

To configure, select *Trigger overview* as type:



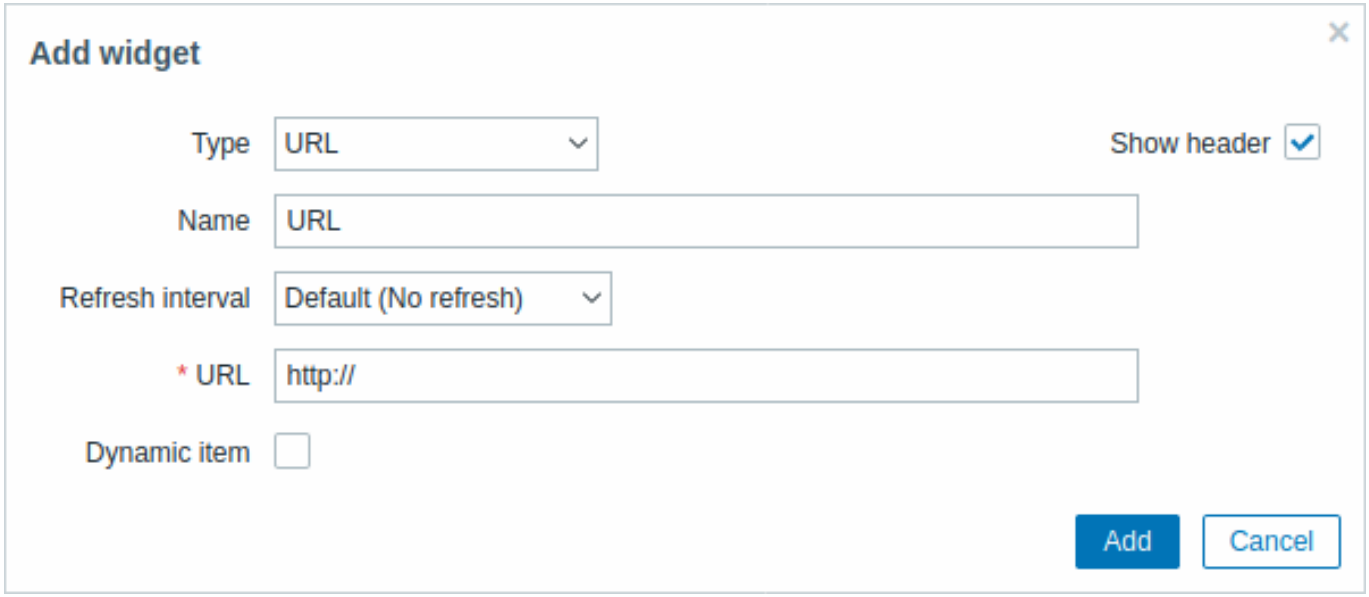
You may set the following specific options:

Show	Filter triggers by trigger state: **Recent problems** - *(default)* show triggers that recently have been or still are in a PROBLEM state (resolved and unresolved); **Problems** - show triggers that are in a PROBLEM state (unresolved); **Any** - show all triggers.
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.
Application	Enter the application name.
Show suppressed problems	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.
Hosts location	Select host location - left or top.

URL

This widget displays the content retrieved from the specified URL.

To configure, select *URL* as type:



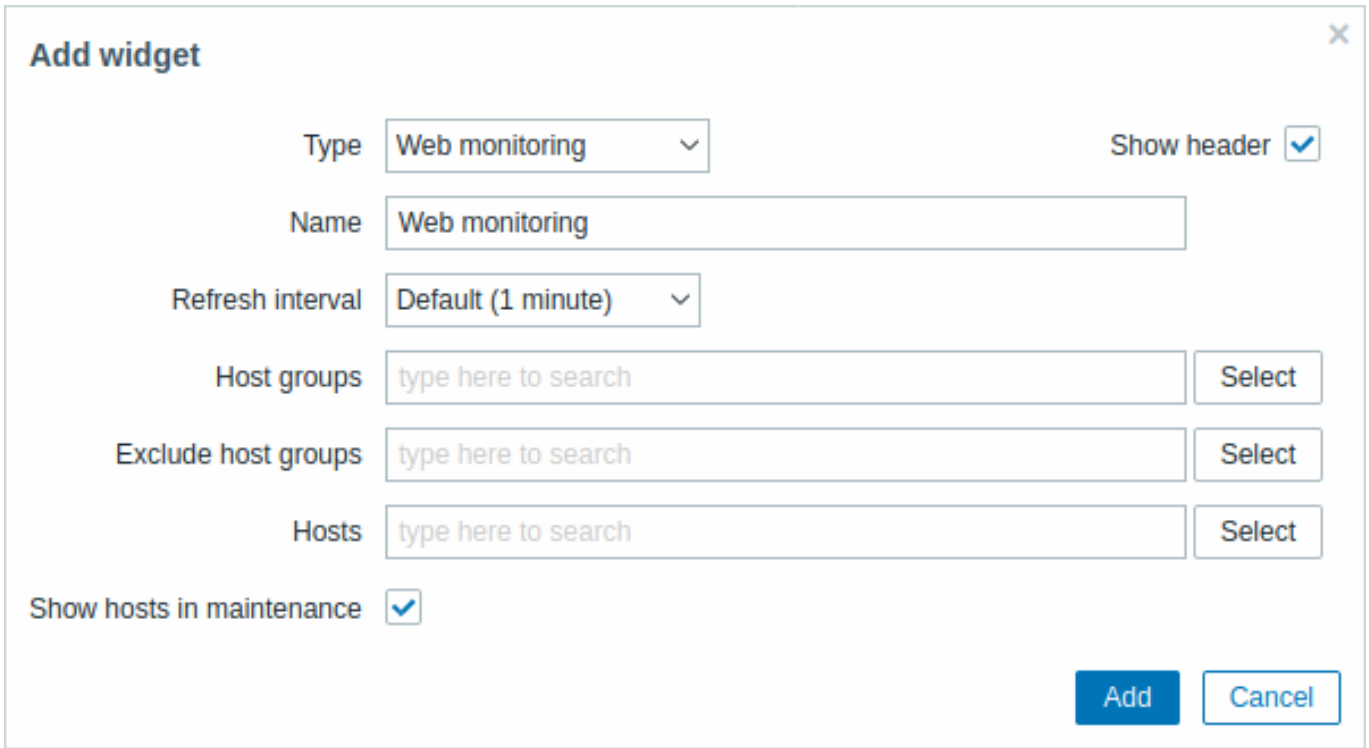
You may set the following specific options:

URL	Enter the URL to display. External URLs must start with `http://` or `https://`. Since Zabbix 4.4.8, internal URLs support relative paths (for example, `zabbix.php?action=report.status`). {HOST.*} macros are supported.
Dynamic item	Set to display different URL content depending on the selected host. This can work if {HOST.*} macros are used in the URL.

Attention:
Browsers might not load an HTTP page included in the widget, if Zabbix frontend is accessed over HTTPS.

Web monitoring

This widget displays a status summary of the active web monitoring scenarios.

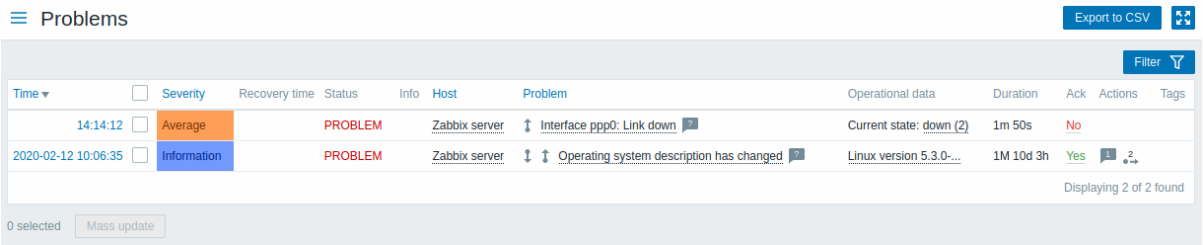


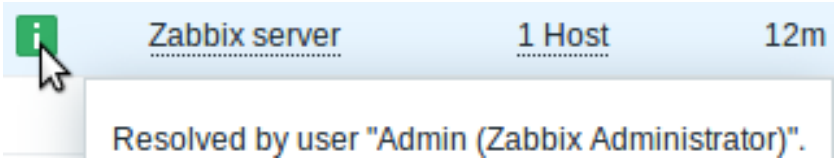
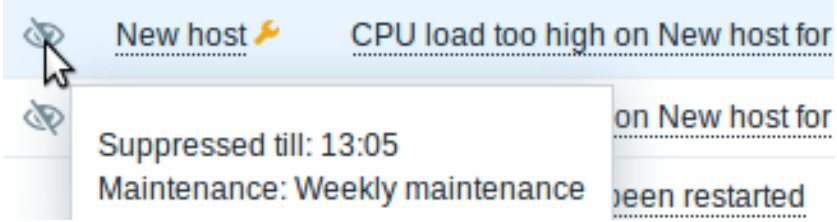
Note:
In cases when a user does not have permission to access certain widget elements, that element's name will appear as *Inaccessible* during the widget's configuration. This results in *Inaccessible Item, Inaccessible Host, Inaccessible Group, Inaccessible Map and Inaccessible Graph* appearing instead of the "real" name of the element.

2 Problems

Overview

In *Monitoring* → *Problems* you can see what problems you currently have. Problems are those triggers that are in the "Problem" state.



Column	Description
Time	Problem start time is displayed.
Severity	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.
Recovery time	Problem resolution time is displayed.
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*.
Info	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:  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: 
Host	Problem host is displayed.

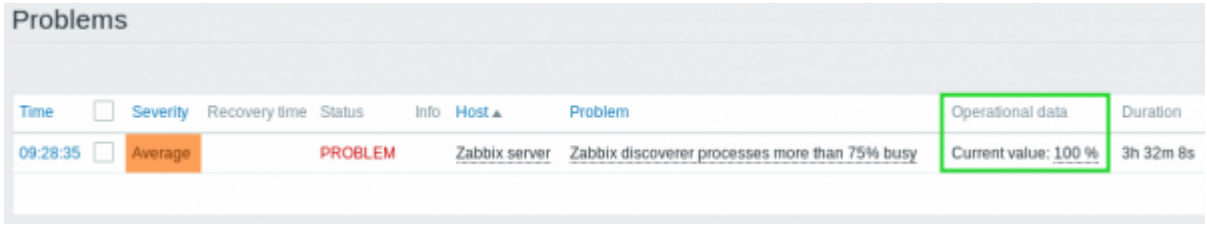
Column	Description
Problem	Problem name is displayed. Problem name is based on the name of the underlying problem trigger. Macros in the trigger name are resolved at the time of the problem happening and the resolved values do not update any more. *Note* that it is possible to append the problem name 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).
Operational data	Operational data are displayed containing latest item values. Operational data can be a combination of text and item value macros, if configured on a trigger level. If no operational data is configured on a trigger level, 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.
Duration	Problem duration is displayed. See also: Negative problem duration
Ack	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.
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 for the explanation on icons used in the popup for actions taken.
Tags	Event 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

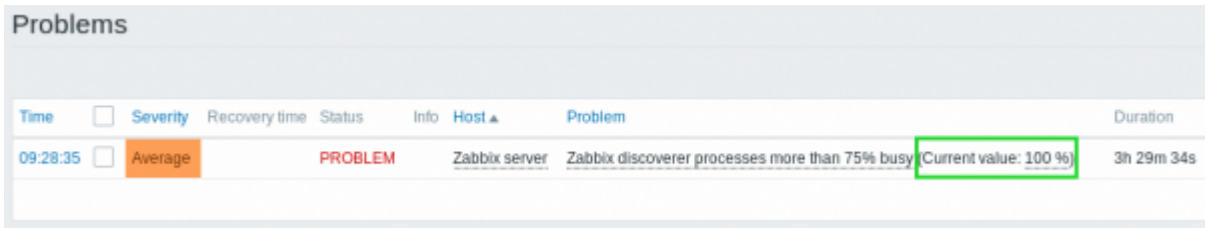
It is possible to display operational data for current problems, i.e. the latest item values as opposed to the item values at the time of the problem.

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 user or being closed by correlation).

Note that closing the problem manually does not produce a new value so the resolved value of {ITEM.LASTVALUE<1-9>} or {ITEM.VALUE<1-9>} will still show the value from the problem time.

{ITEM.LASTVALUE<1-9>} or {ITEM.VALUE<1-9>} will resolve to *UNKNOWN* if the latest history value has been collected more than the *Max history display period* time ago (see *Administration → General*).

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 item.nodata() 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 item.nodata() 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.

Note:
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

Button to the right offers the following option:

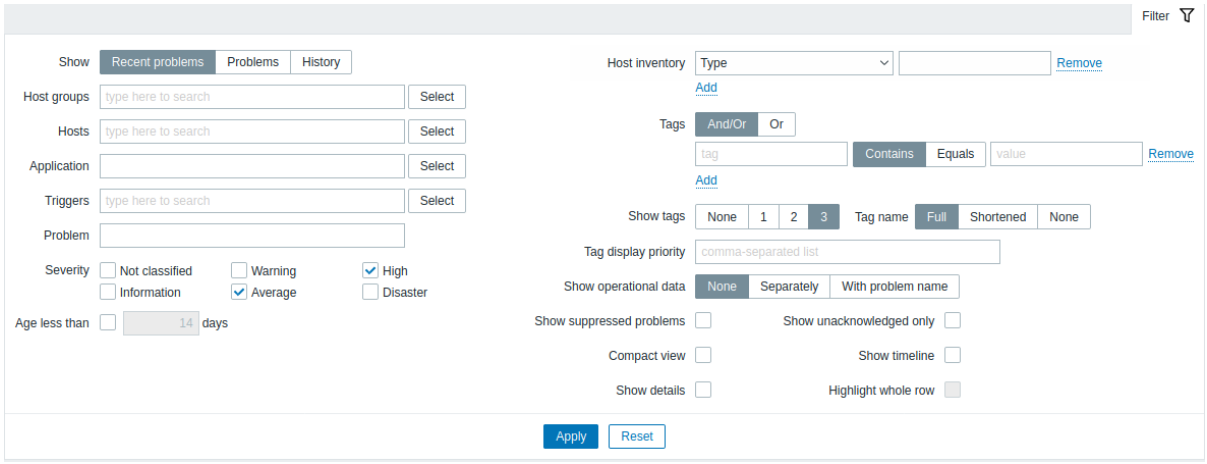
Export to CSV	Export content from all pages to a CSV file.

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.

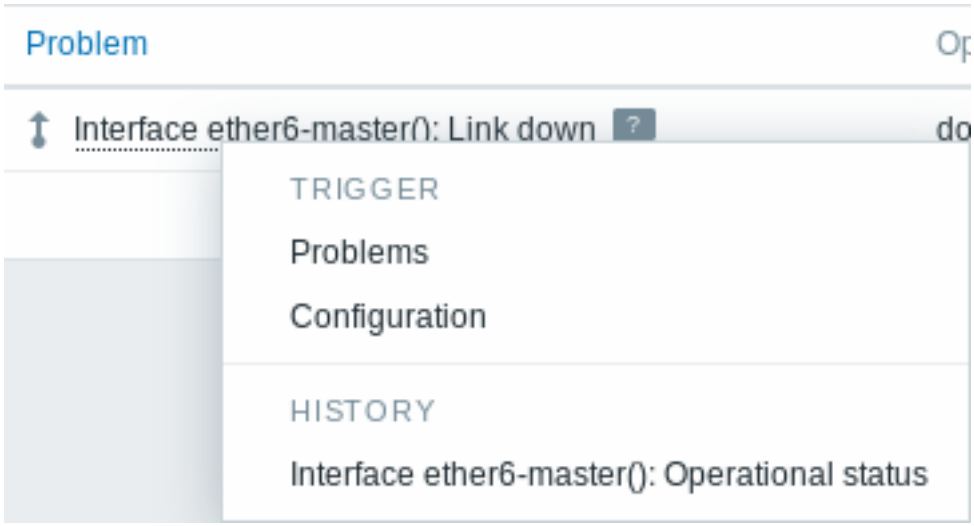


Parameter	Description
Show	Filter by problem status: **Recent problems** - unresolved and recently resolved problems are displayed (default) **Problems** - unresolved problems are displayed **History** - history of all events is displayed
Host groups	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups.
Hosts	Filter by one or more hosts.
Application	Filter by application name.
Triggers	Filter by one or more triggers.
Problem	Filter by problem name.
Severity	Filter by trigger (problem) severity.
Age less than	Filter by how old the problem is.
Host inventory	Filter by inventory type and value.
Tags	Filter by event tag name and value. Several conditions can be set. There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having same tag name will be grouped by Or condition **Or** - enough if one condition is met 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) 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

Parameter	Description
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 which would otherwise be suppressed (not shown) because of host maintenance.
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 the 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.

Event menu

Clicking on the problem name brings up the event menu:

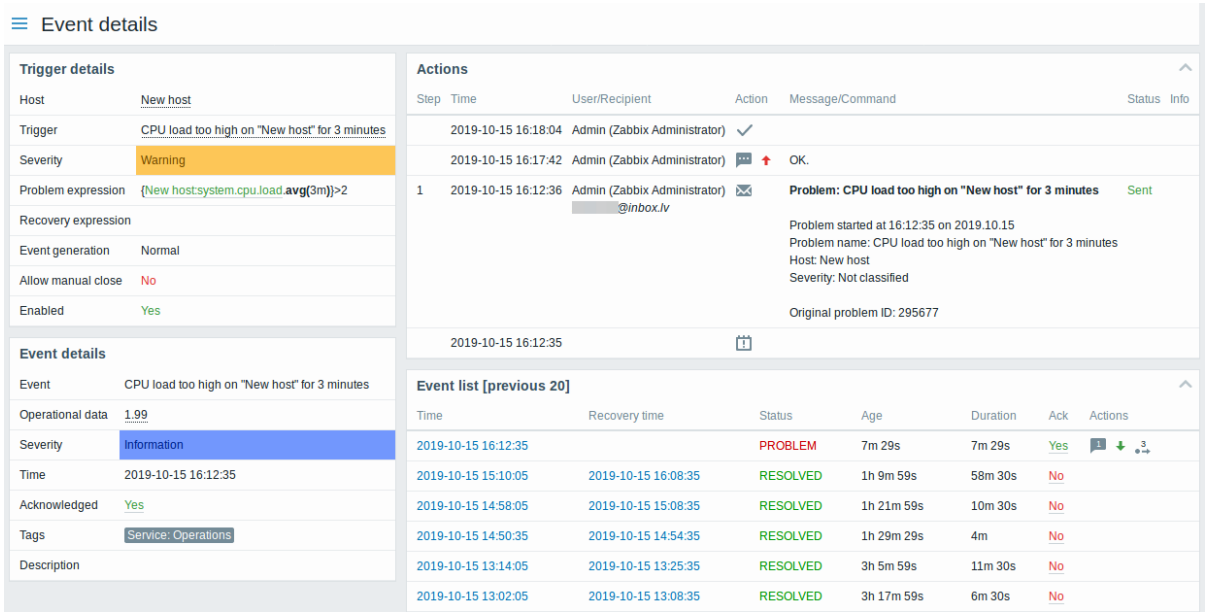


The event menu allows to:

- 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)

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.

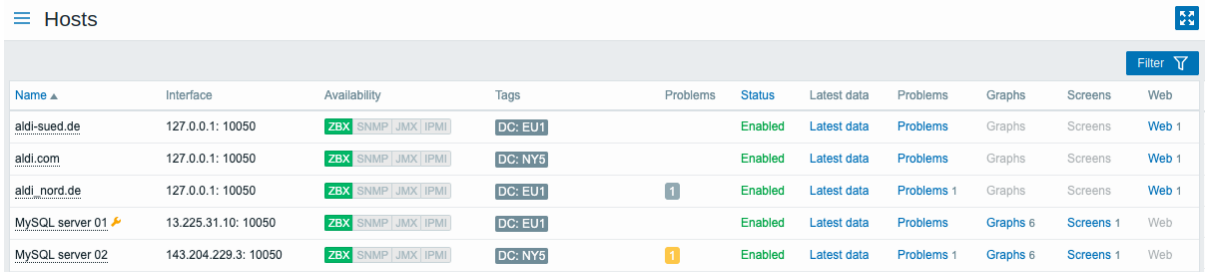
In the action list, the following icons are used to denote the activity type:

-  - problem event generated
-  - message has been sent
-  - problem event acknowledged
-  - problem event unacknowledged
-  - comment has been added
-  - 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)
-  - remote command has been executed
-  - problem event has recovered
-  - problem has been closed manually

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, screens, and web scenarios.



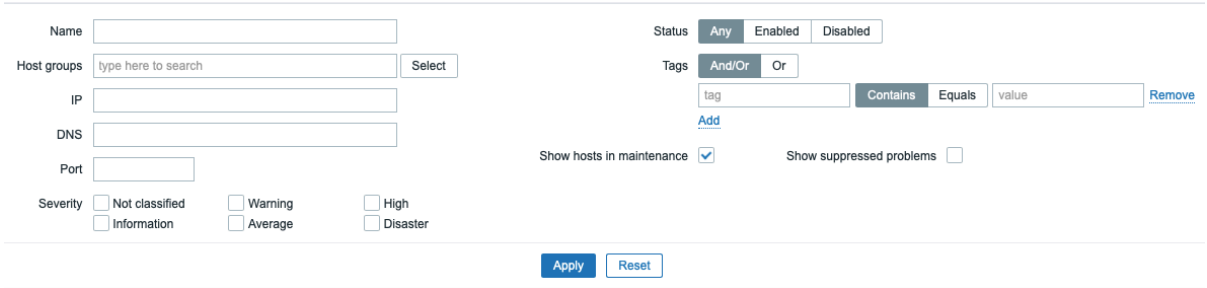
Column	Description
Name	Host's visible name. Clicking on the name brings up the host menu. An orange wrench icon after the name indicates that this host is in maintenance. Clicking on the header row in this column will sort hosts by name in ascending (default) or descending order.
Interface	The main interface of the host is displayed.
Availability	Availability of the host is displayed. Four icons each represent a supported interface (Zabbix agent, SNMP, IPMI, JMX). The current status of the interface is displayed by the respective color: **Green** - available **Red** - not available (upon mouseover, details of why the interface cannot be reached are displayed) **Gray** - unknown or not configured Note that active Zabbix agent items do not affect host availability.
Tags	Tags of the host and all linked templates, with macros unresolved.
Problems	Square icons that show current open problems. Icon color indicates problem severity. The number on an icon means the number of problems for the given severity. Use the filter to select whether suppressed problems should be included (not included by default).
Status	Host status is displayed - *Enabled* or *Disabled*. Clicking on the header row in this column will sort hosts by status in ascending or descending order.
Latest data	Clicking on the link will open *Monitoring* **-** *Latest data* page with all the latest data collected from the host.
Problems	Clicking on the link will open *Monitoring* **-** *Problems* section filtered to show only information for the given host.
Graphs	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.
Screens	Clicking on the link will display screens configured for the host. The number of screens is displayed in gray. If a host has no screens, the link is disabled (gray text) and no number is displayed.
Web	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.

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 hosts you are interested in. 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.



Notes:

- *Host groups* field supports multiple entries.
- Specifying a parent host group implicitly selects all nested host groups.
- By default problems of all severities are displayed, if not suppressed.
- By default hosts in maintenance are displayed, but suppressed problems on these hosts are not shown.
- When *Severity* filter is used and *Show suppressed problems* checkbox is unchecked, hosts with suppressed problems of the specified severity will not be displayed.
- Hosts can be filtered by host-level tags as well as tags from all linked templates, including parent templates.

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, however, no more than 20 graphs can be displayed at one time.

The *View as* option allows to view the data graphically or as values. Graphs for disabled hosts are also accessible.

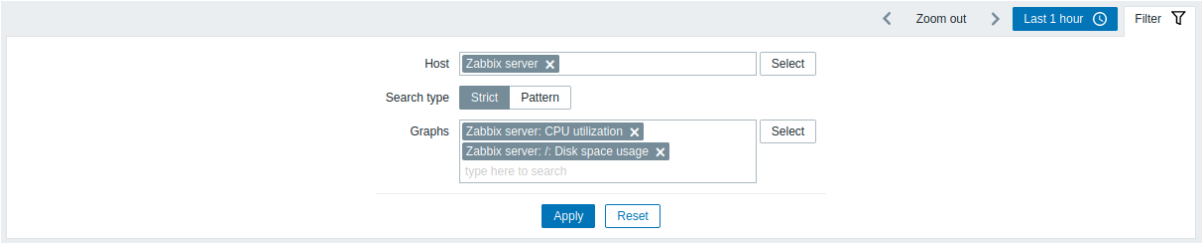
Time period selector

Take note of the time period selector above the graph. It allows to select 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 one host at a time (host is mandatory), and then specify host graphs either by selecting from the list or by searching by the graph name pattern.



Buttons

Buttons to the right offer the following options:

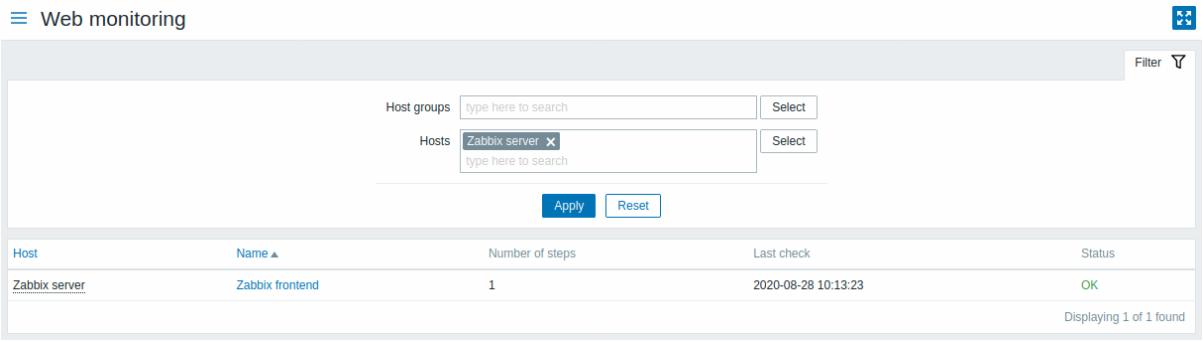
	Add graph to the favorites widget in the Dashboard.
	The graph is in the favorites widget in the Dashboard. Click to remove graph from the favorites widget.

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.



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.

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.

Only values that fall within the last 24 hours are displayed by default. This limit has been introduced with the aim of improving initial loading times for large pages of web monitoring. It is also possible to change this limitation by changing the value of ZBX_HISTORY_PERIOD constant in *include/defines.inc.php*.

The scenario name is link to more detailed statistics about it:

Buttons

View mode buttons being common for all sections are described on the Monitoring page.

4 Overview

Overview

The *Monitoring → Overview* section may display either:

- *Trigger overview* - an overview of trigger states
- *Data overview* - a comparison of data for various hosts at once

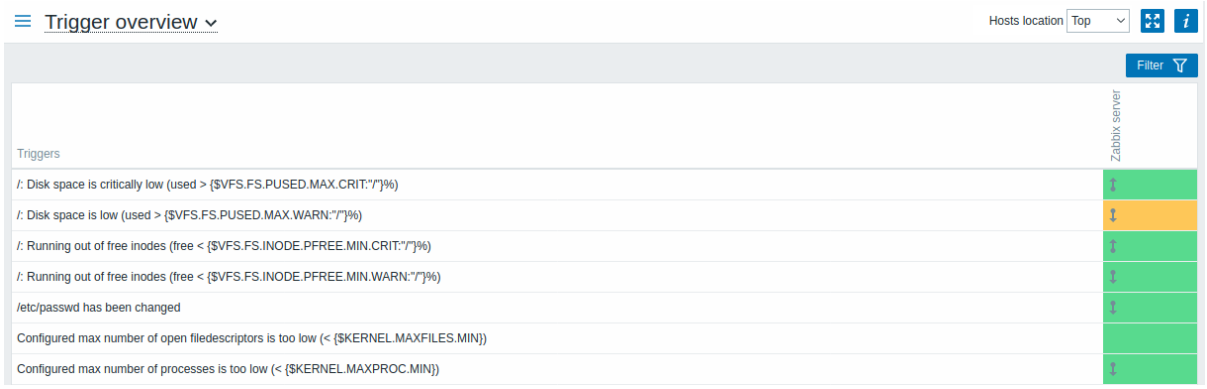
The following additional display options are available:

- select horizontal or vertical display of information in the *Hosts location* dropdown

Note that there is a hard-coded limit of 50 records displayed. There is no pagination. If more records exist, a message is displayed at the bottom of the table, asking to provide more specific filtering criteria. Note that this limit is applied first, before any further filtering of data.

Overview of triggers

In the next screenshot *Trigger overview* is selected. As a result, the trigger states of a local host 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):

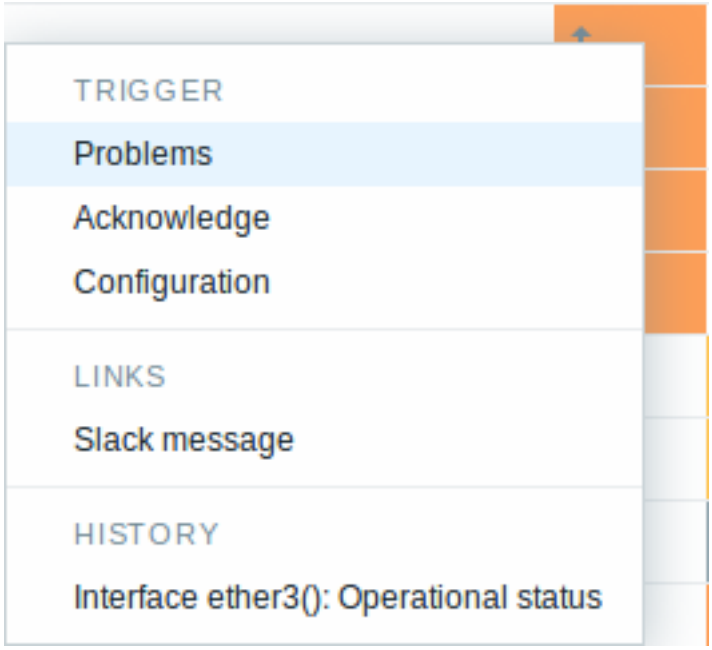


Note that recent trigger changes (within the last 2 minutes) will be displayed as blinking blocks.

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.



Buttons

Button to the right offers the following option:

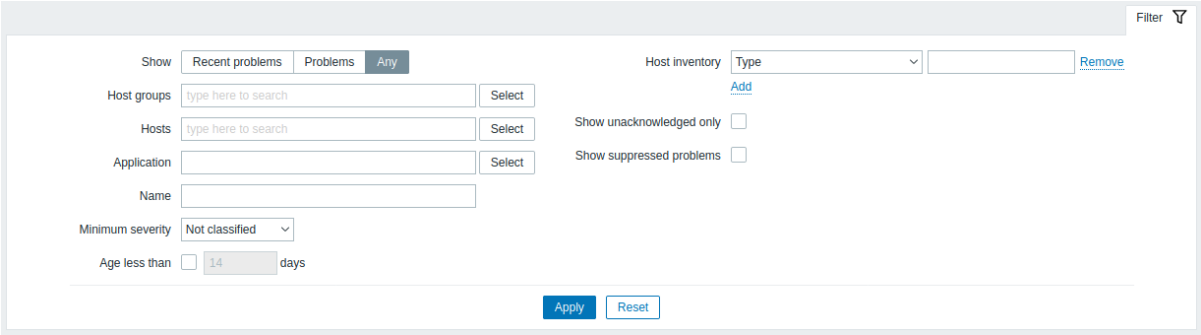
	Additional information on the page content is displayed if you roll the mouse over this button.

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.

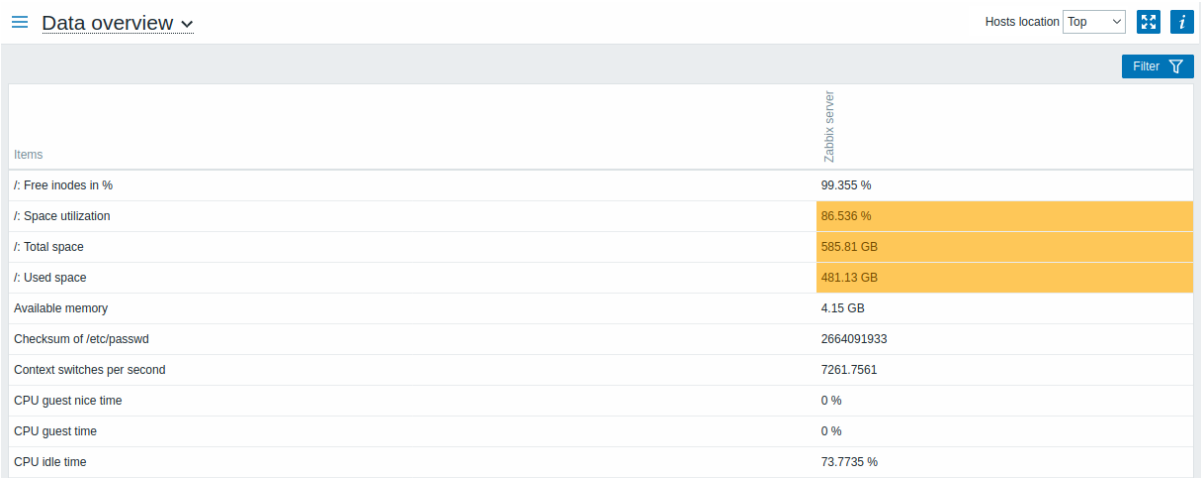
The filter is located above the table.



Parameter	Description
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	Filter by host group.
Hosts	Filter by host.
Application	Filter by application.
Name	Filter by problem name.
Minimum severity	Filter by minimum problem severity.
Age (less than)	Mark the checkbox to filter by problem age.
Host inventory	Filter by inventory type and value.
Show unacknowledged only	Mark the checkbox to only display problems which are unacknowledged.
Show suppressed problems	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.

Overview of data

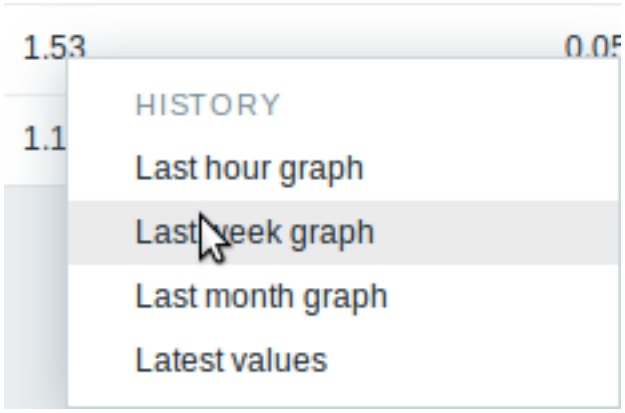
In the next screenshot *Data overview* is selected. As a result, item data of a local host is displayed.



The color of problem items is based on the problem severity color, which can be adjusted in the problem update screen.

Only values that fall within the last 24 hours are displayed by default. This limit has been introduced with the aim of improving initial loading times for large pages of latest data. It is also possible to change this limitation by changing the value of ZBX_HISTORY_PERIOD constant in *include/defines.inc.php*.

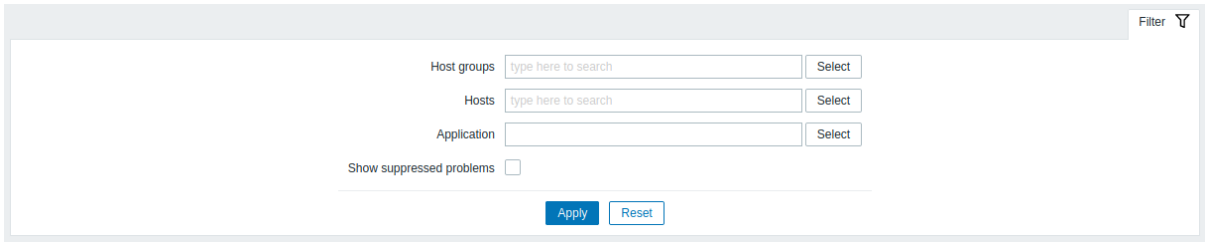
Clicking on a piece of data offers links to some predefined graphs or latest values.



Using filter

You can use the filter to display only the data you are interested in. For better search performance, data is searched with macros unresolved.

The filter is located above the table.

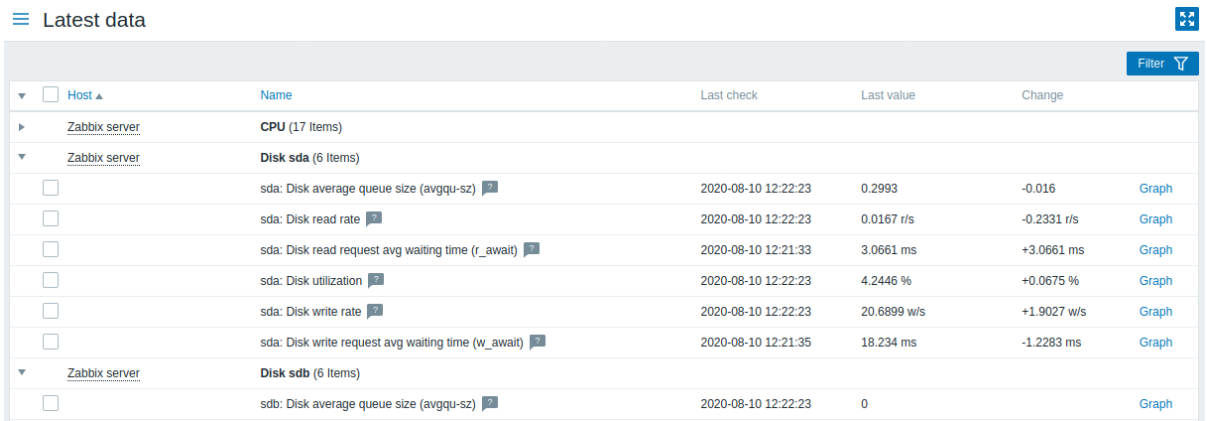


Parameter	Description
Host groups	Filter by host group.
Hosts	Filter by host.
Application	Filter by application.
Show suppressed problems	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.

5 Latest data

Overview

The section in *Monitoring* → *Latest data* can be used to view latest values gathered by items as well as to access various graphs for the items.



In the list displayed, click on ▶ before a host and the relevant application to reveal latest values of that host and application. You can expand all hosts and all applications, thus revealing all items by clicking on ▶ in the header row. (Note that expanding/collapsing applications is not available in Zabbix 5.0.2.)

Items are grouped by host and application. Items are displayed with their name, last check time, last value, change amount and a link to a simple graph/history of item values.

An icon with a question mark is displayed next to the item name for all items that have a description. If you position the mouse cursor on this icon, the item description is displayed as a tooltip.

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*.

Only values that fall within the last 24 hours are displayed by default. This limit has been introduced with the aim of improving initial loading times for large pages of latest data. It is also possible to change this limitation by changing the value of ZBX_HISTORY_PERIOD constant in *include/defines.inc.php*.

Attention:
For items with 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.

Buttons

View mode buttons being common for all sections are described on the Monitoring page.

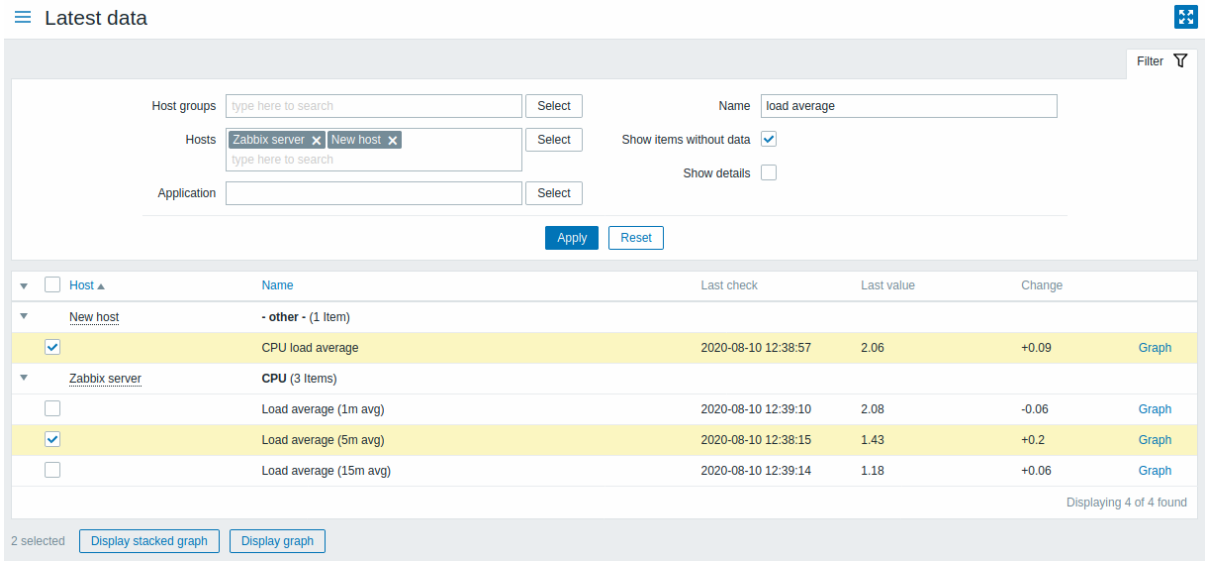
Using filter

You can use the filter to display only the items you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* link is located above the table to the right. You can use it to filter items by host group, host, application, a string in the item name; you can also select to display items that have no data gathered.

Specifying a parent host group implicitly selects all nested host groups.

Show details allows to extend displayable information on the items. Such details as refresh interval, history and trends settings, item type and item errors (fine/unsupported) are displayed. A link to item configuration is also available.



By default, items without data are shown but details are not displayed. For better page performance, since Zabbix 5.0.3, the *Show items without data* option is checked and disabled if no host is selected in the filter.

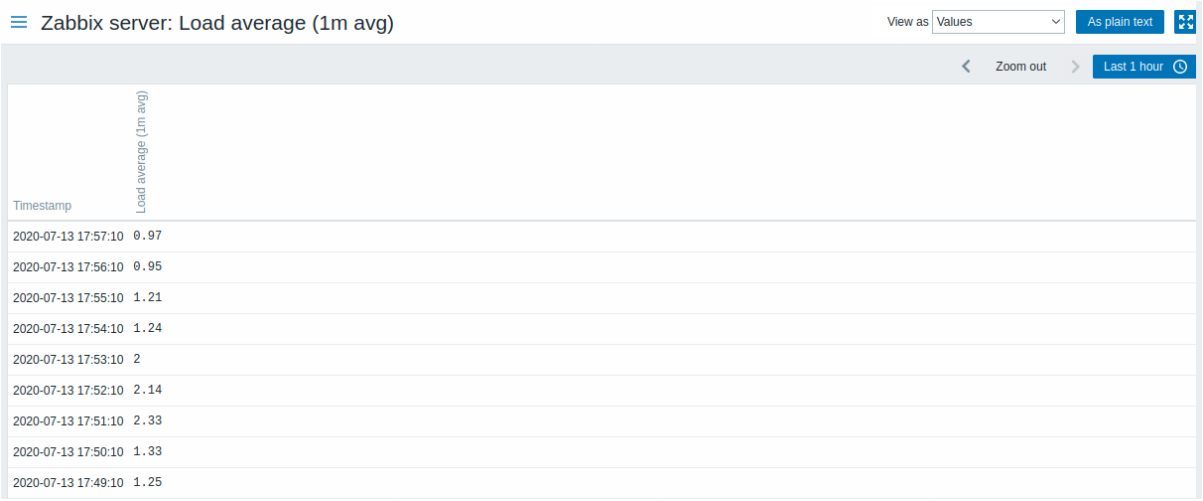
Ad-hoc graphs for comparing items

You may use the checkbox in the first column to select several items and then compare their data in a simple or stacked ad-hoc graph. To do that, select items of interest, then click on the required graph button below the table.

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.

Note:
The total amount of values displayed is defined by the value of *Limit for search and filter results* parameter, set in Administration → General.

6 Screens

Overview

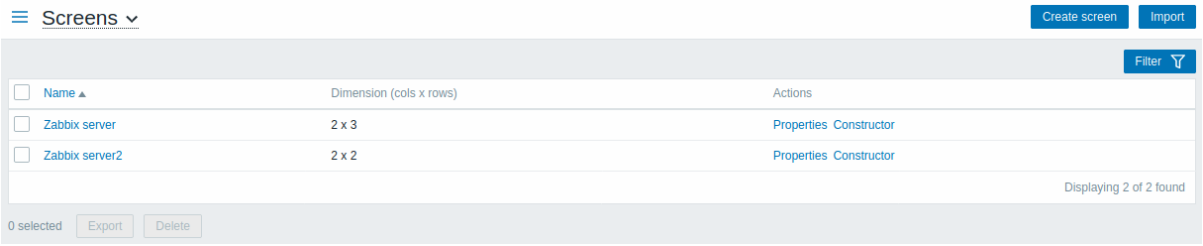
In the *Monitoring* → *Screens* section you can configure, manage and view Zabbix global screens and slide shows.

When you open this section, you will either see the last screen/slide show you accessed or a listing of all entities you have access to. Screen/slide show listing can be filtered by name.

All screens/slide shows can be either public or private. The public ones are available to all users, while private ones are accessible only to their owner and the users the entity is shared with.

Use the dropdown in the title bar to switch between screens and slide shows.

Screen listing



Displayed data:

Column	Description
Name	Name of the screen. Click on the name to view the screen.
Dimensions	The number of columns and rows of the screen.
Actions	Two actions are available: **Properties** - edit general screen properties (name and dimensions) **Constructor** - access the grid of screen elements for editing

To create a new screen, click on the *Create screen* button in the top right-hand corner. To import a screen from an XML file, click on the *Import* button in the top right-hand corner. The user who imports the screen will be set as its owner.

Mass editing options

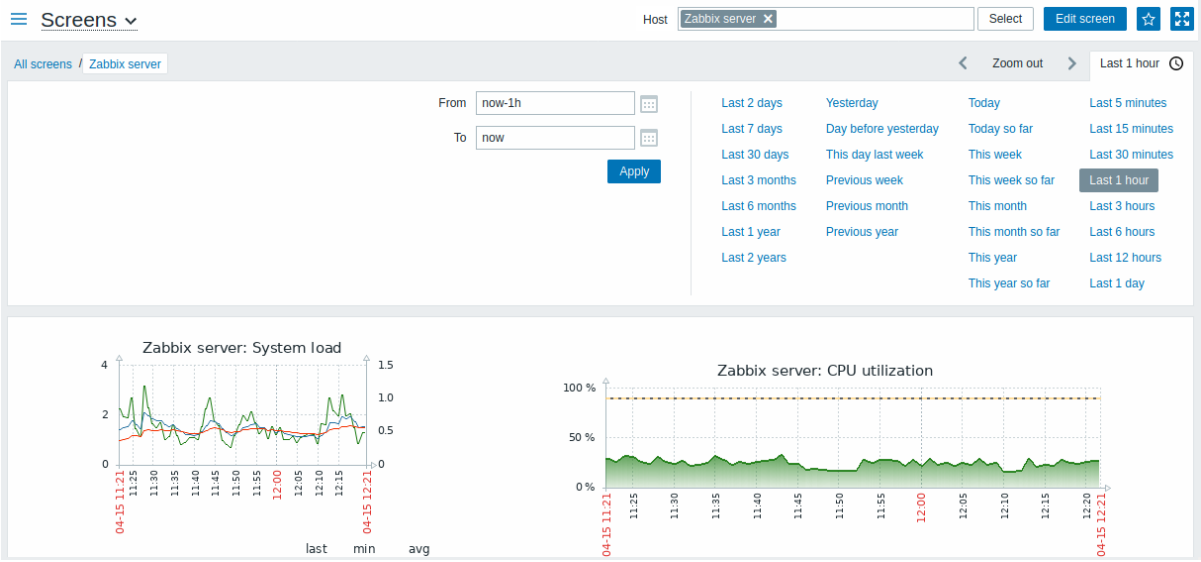
Buttons below the list offer some mass-editing options:

- *Export* - export the screens to an XML file
- *Delete* - delete the screens

To use these options, mark the checkboxes before the respective screens, then click on the required button.

Viewing screens

To view a screen, click on its name in the list of all screens.



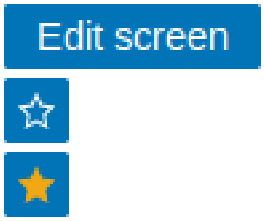
A field for selecting the host is available when there are screen elements with dynamic, host-dependent content.

Time period selector

Take note of the time period selector above the screen. It allows to select often required periods with one mouse click, affecting the data displayed in graphs etc. See also: Time period selector

Buttons

Buttons to the right offer the following options:

	Go to the screen constructor to edit the screen. Add screen to the favorites widget in the Dashboard. The screen is in the favorites widget in the Dashboard. Click to remove screen from the favorites widget.

View mode buttons being common for all sections are described on the Monitoring page.

Slide show listing

Use the dropdown in the title bar to switch from screens to slide shows.

Displayed data:

Column	Description
Name	Name of the slide show. Click on the name to view the slide show.
Delay	The default duration of showing one slide is displayed.
Number of slides	The number of slides in the slide show is displayed.
Actions	One action is available: **Properties** - edit slide show properties

To create a new slide show, click on the *Create slide show* button in the top right-hand corner.

Mass editing options

A button below the list offers one mass-editing option:

- *Delete* - delete the slide shows

To use this option, mark the checkboxes before the respective slide shows and click on *Delete*.

Viewing slide shows

To view a slide show, click on its name in the list of all slide shows.

A field for selecting the host is available when there are screen elements with dynamic, host-dependent content.

Buttons

Buttons to the right offer the following options:

	Go to the slide show properties. Add slide show to the favorites widget in the Dashboard. The slide show is in the favorites widget in the Dashboard. Click to remove slide show from the favorites widget. Slow down or speed up a slide show.

View mode buttons being common for all sections are described on the Monitoring page.

Referencing a screen

Screens can be referenced by both `elementid` and `screenname` GET parameters. For example,

`http://zabbix/zabbix/screens.php?screenname=Zabbix%20server`

will open the screen with that name (Zabbix server).

If both `elementid` (screen ID) and `screenname` (screen name) are specified, `screenname` has higher priority.

7 Maps

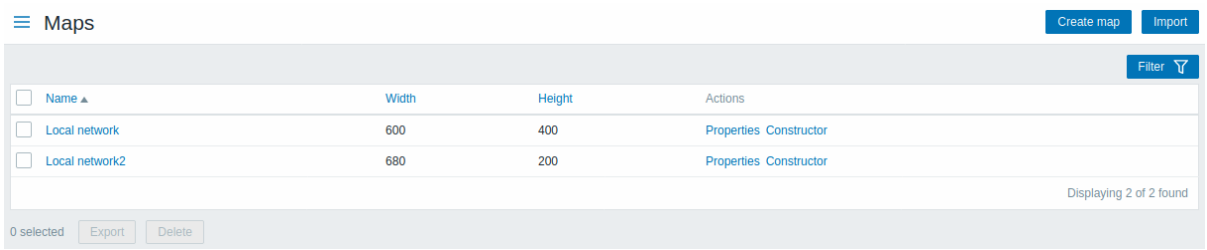
Overview

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. Map listing can be filtered by name.

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:

Column	Description
Name	Name of the map. Click on the name to view the map.
Width	Map width is displayed.

Column	Description
Height	Map height is displayed.
Actions	Two actions are available: **Properties** - edit general map properties **Constructor** - access the grid for adding map elements

To configure a new map, click on the *Create map* button in the top right-hand corner. To import a map from an XML 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 an XML file
- *Delete* - delete the maps

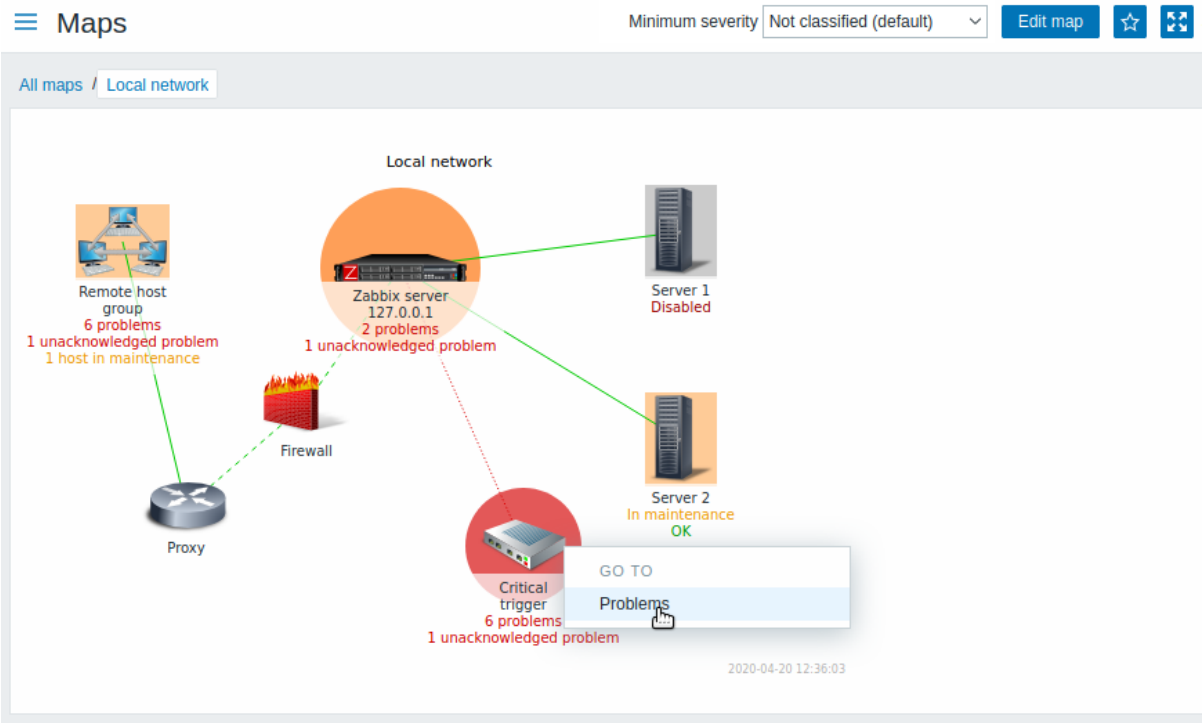
To use these options, mark the checkboxes before the respective maps, then click on the required button.

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 dropdown 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 map configuration. If the map contains a submap, navigating to the submap will retain the higher-level map severity (except if it is *Not classified*, in this case it will not be passed to the submap).

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 (since Zabbix 5.0.24, only 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:

Edit map	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

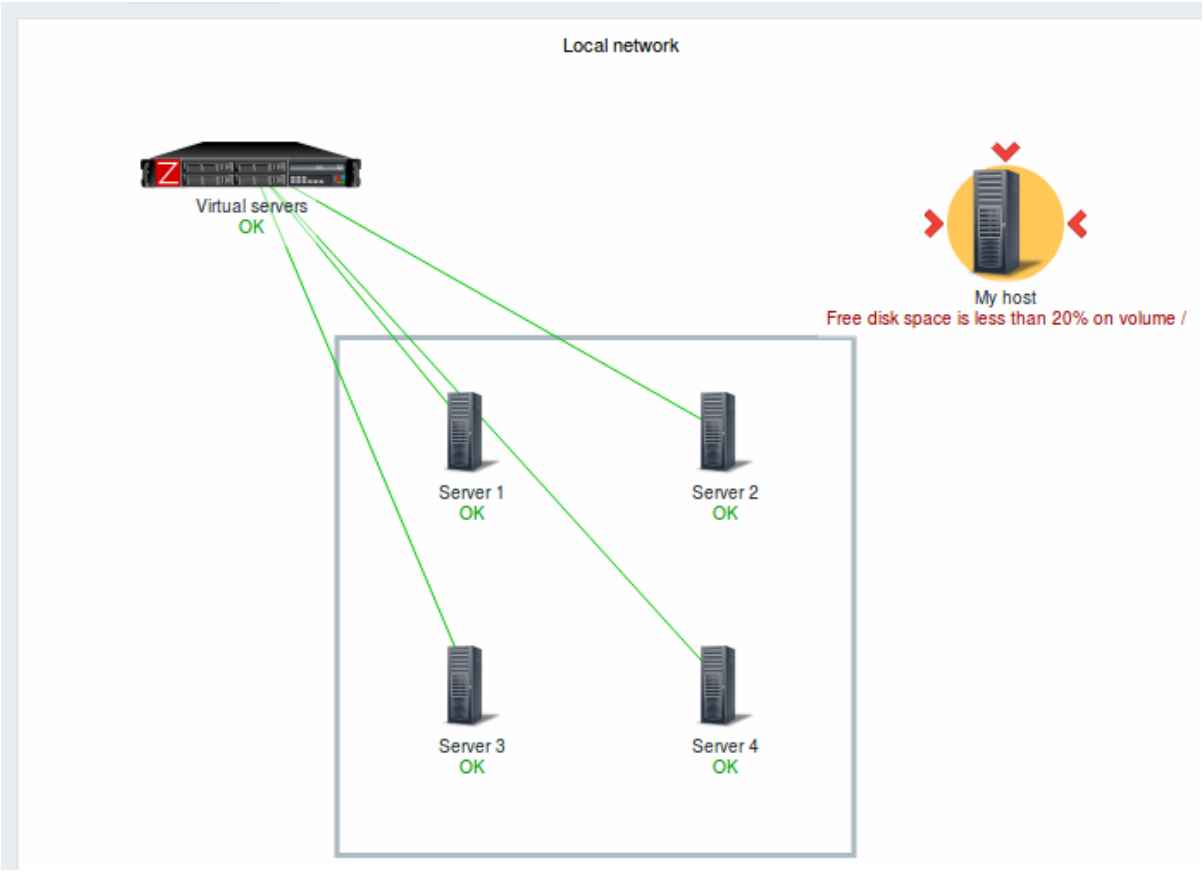
A hidden "aria-label" property is available allowing map information to be read with a screen reader. Both general map description and individual element description is available, in the following format:

- 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
```

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`

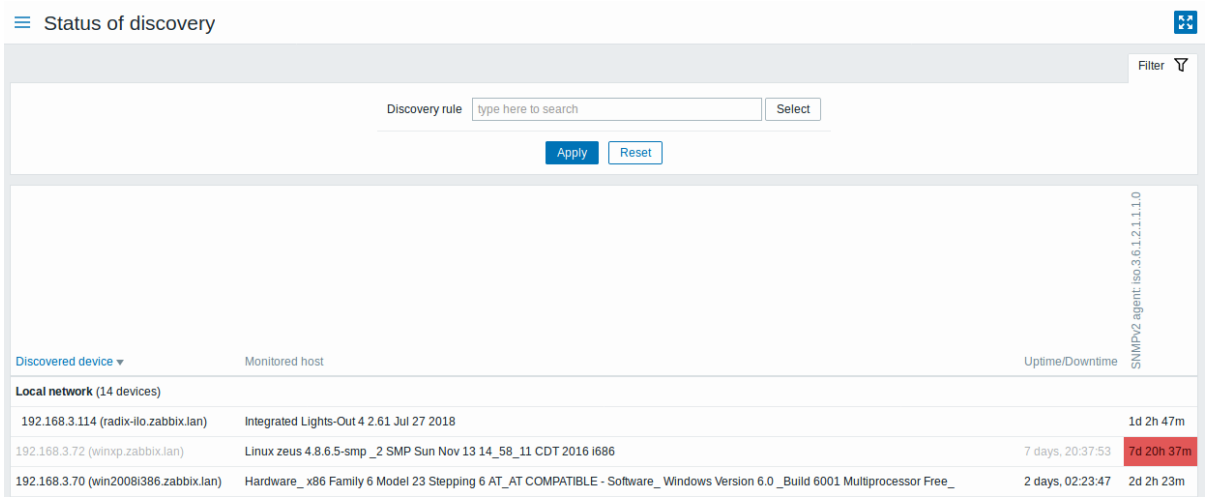
will open the map with that name (Local network).

If both `sysmapid` (map ID) and `mapname` (map name) are specified, `mapname` has higher priority.

8 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.

Attention:
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

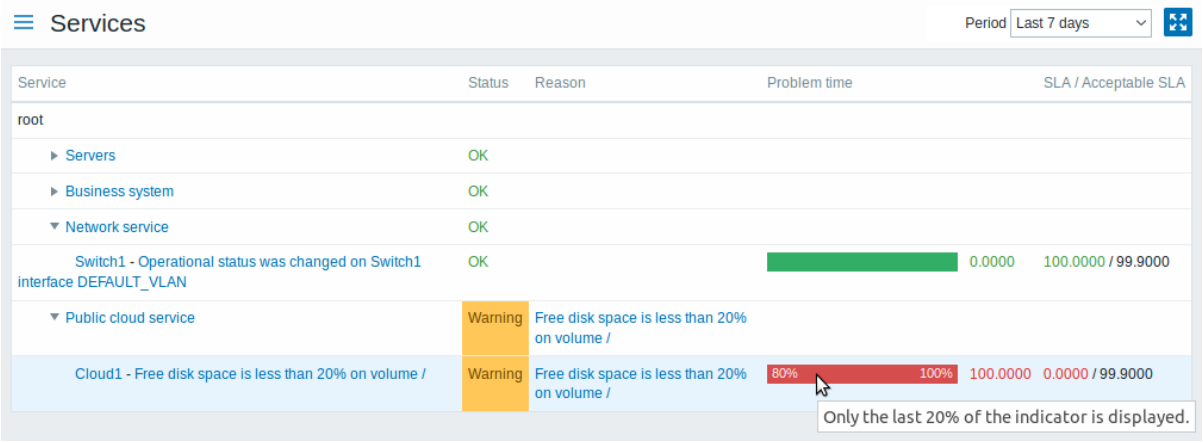
You can use the filter to display only the discovery rules you are interested in. For better search performance, data is searched with macros unresolved.

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.

9 Services

Overview

In the *Monitoring* → *Services* section the status of IT infrastructure or business services is displayed.

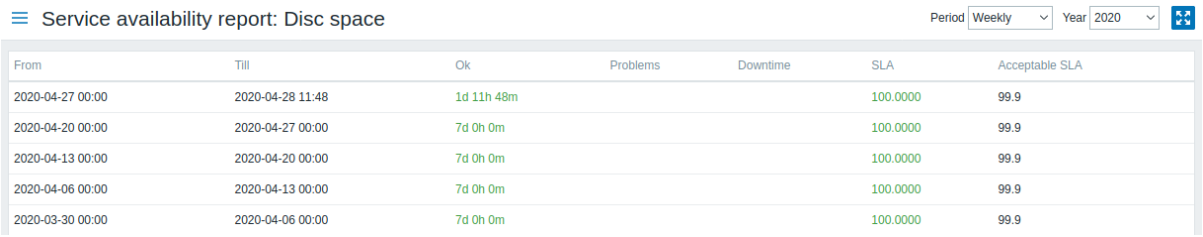


A list of the existing services is displayed along with data of their status and SLA. From the dropdown in the upper right corner you can select a desired period for display.

Displayed data:

Parameter	Description
Service	Service name.
Status	Status of service: **OK** - no problems **(trigger color and severity)** - indicates a problem and its severity
Reason	Indicates the reason of problem (if any).
Problem time	Displays SLA bar. Green/red ratio indicates the proportion of availability/problems. The bar displays the last 20% of SLA (from 80% to 100%). The bar contains a link to a graph of availability data.
SLA/Acceptable SLA	Displays current SLA/expected SLA value. If current value is below the acceptable level, it is displayed in red.

You can also click on the service name to access the *Service availability report*.



Here you can assess service availability data over a longer period of time on daily/weekly/monthly/yearly basis.

Buttons

View mode buttons being common for all sections are described on the Monitoring page.

2 Inventory

Overview

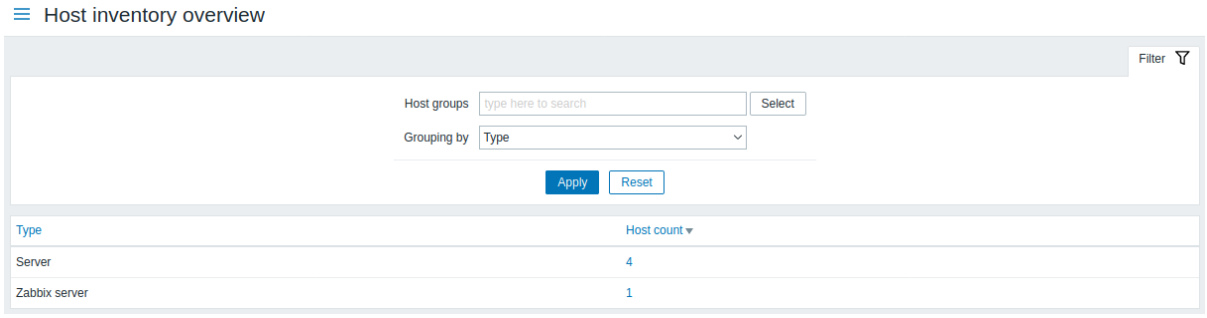
The Inventory menu features sections providing an overview of host inventory data by a chosen parameter as well as the ability to view host inventory details.

1 Overview

Overview

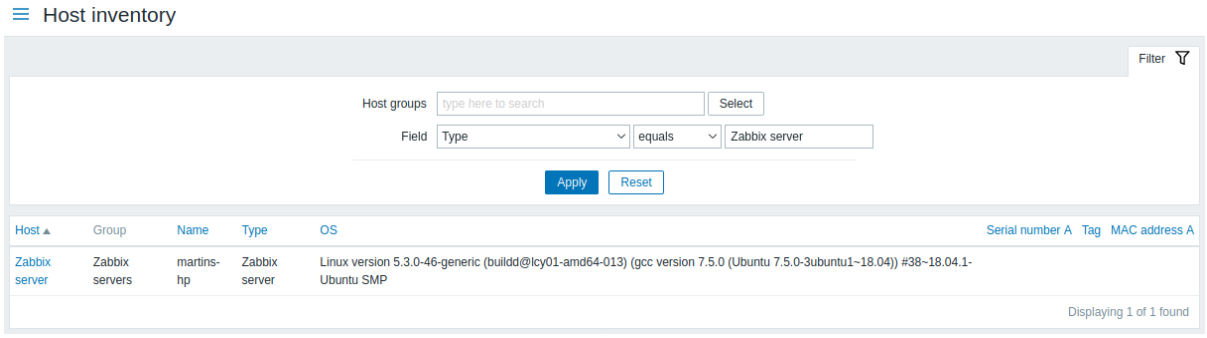
The *Inventory → Overview* section provides ways of having an overview of host inventory data.

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.

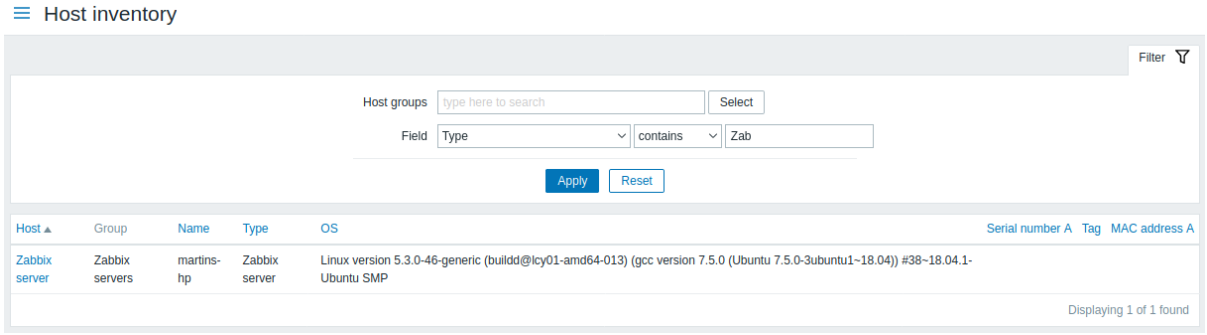


2 Hosts

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.

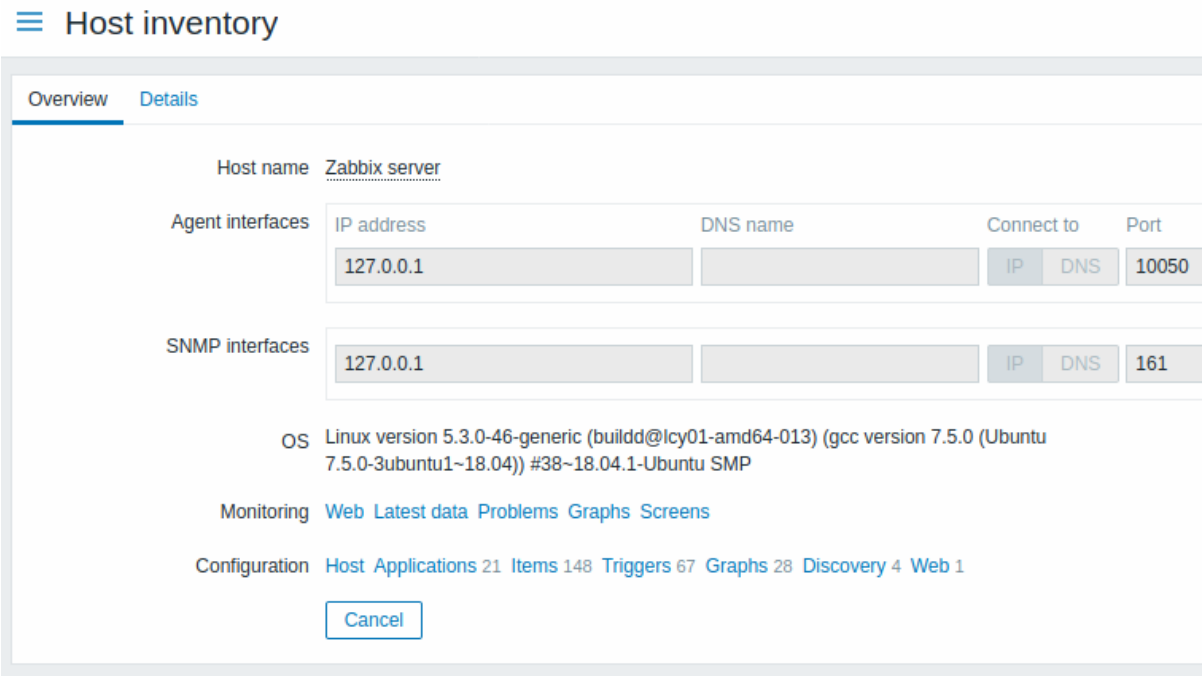


To display all host inventories, select no host group in the filter, clear the comparison field in the filter and press ”Filter”.

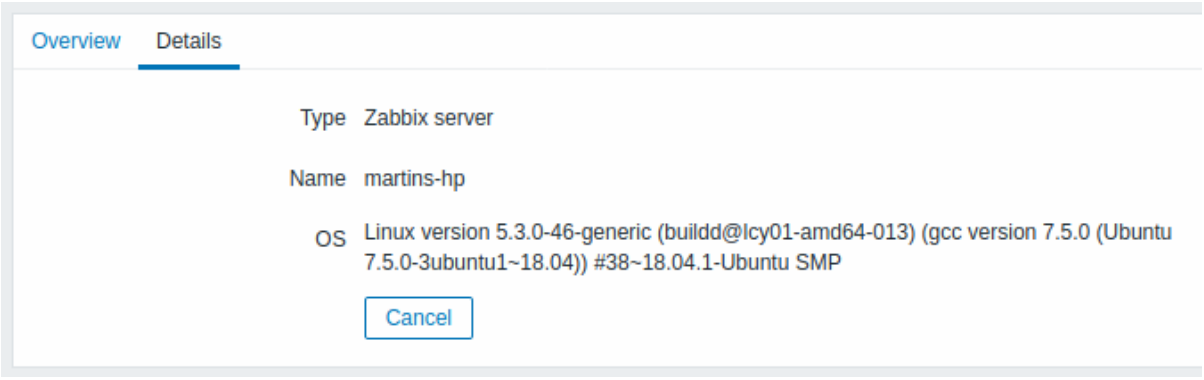
While only some key inventory fields are displayed in the table, you can also view all available inventory information for that host. To do that, click on the host name in the first column.

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.

3 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 system data is displayed.

System information

Parameter	Value	Details
Zabbix server is running	Yes	localhost:10051
Number of hosts (enabled/disabled)	8	2 / 6
Number of templates	137	
Number of items (enabled/disabled/not supported)	394	126 / 259 / 9
Number of triggers (enabled/disabled [problem/ok])	128	55 / 73 [1 / 54]
Number of users (online)	2	1
Required server performance, new values per second	1.67	
Database history tables upgraded	No	

This report is also displayed as a widget in the Dashboard.

Displayed data

Parameter	Value	Details
Zabbix server is running	Status of Zabbix server: **Yes** - server is running **No** - server is not running *Note:* To display the rest of the information the web frontend needs the server to be running and there must be at least one trapper process started on the server (StartTrappers parameter in zabbix_server.conf file > 0).	Location and port of Zabbix server.
Number of hosts	Total number of hosts configured is displayed.	Number of monitored hosts/not monitored hosts.
Number of templates	Total number of templates is displayed.	
Number of items	Total number of items is displayed.	Number of monitored/disabled/unsupported items. Items on disabled hosts are counted as disabled.
Number of triggers	Total number of triggers is displayed.	Number of enabled/disabled triggers. [Triggers in problem/ok state.] Triggers assigned to disabled hosts or depending on disabled items are counted as disabled.
Number of users	Total number of users configured is displayed.	Number of users online.
Required server performance, new values per second	The expected number of new values processed by Zabbix server per second is displayed.	*Required server performance* is an estimate and can be useful as a guideline. For precise numbers of values processed, use the `zabbix[wcache,values,all]` internal item. 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 "nodata" maintenance period. Trapper items are not counted.

Parameter	Value	Details
Database history tables upgraded	Database upgrade status: **No** - database history tables have not been upgraded	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.

System information will also display an error message in the following conditions:

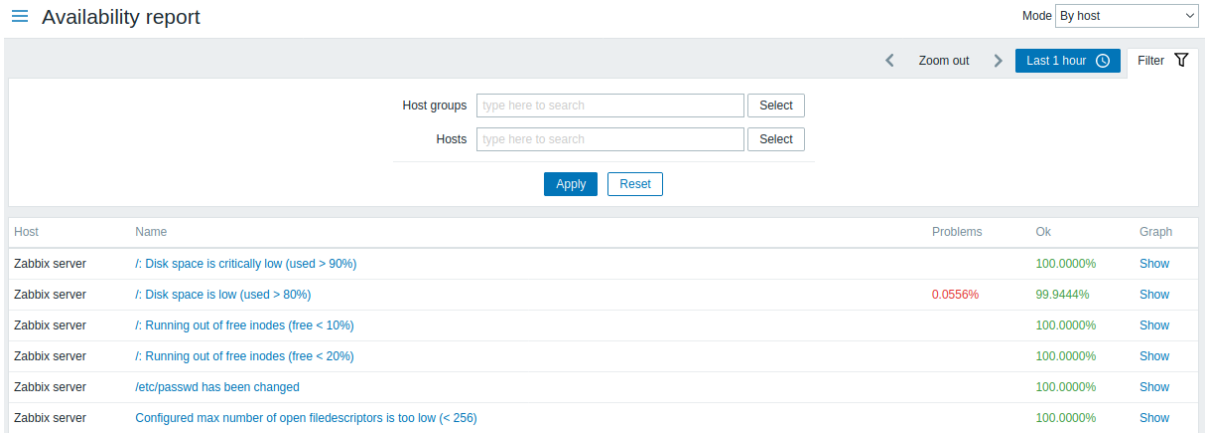
- The database used does not have the required character set or collation (UTF-8).
- Housekeeping for TimescaleDB is incorrectly configured (history or trend tables contain compressed chunks, but *Override item history period* or *Override item trend period* options are disabled).

2 Availability report

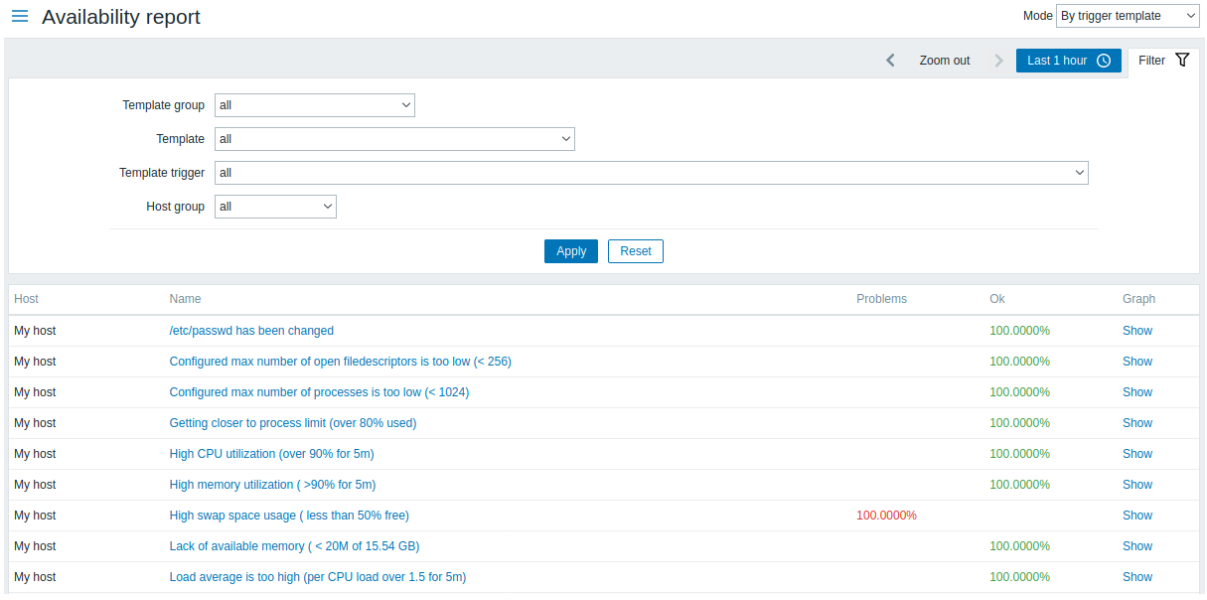
Overview

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 dropdown 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

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.

Parameter	Description
Template group	Select all hosts with triggers from templates belonging to that group. Any host group that includes at least one template can be selected.
Template	Select hosts with triggers from 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.
Template trigger	Select hosts with chosen trigger. Other triggers of the selected hosts will not be displayed.
Host group	Select hosts belonging to the group.

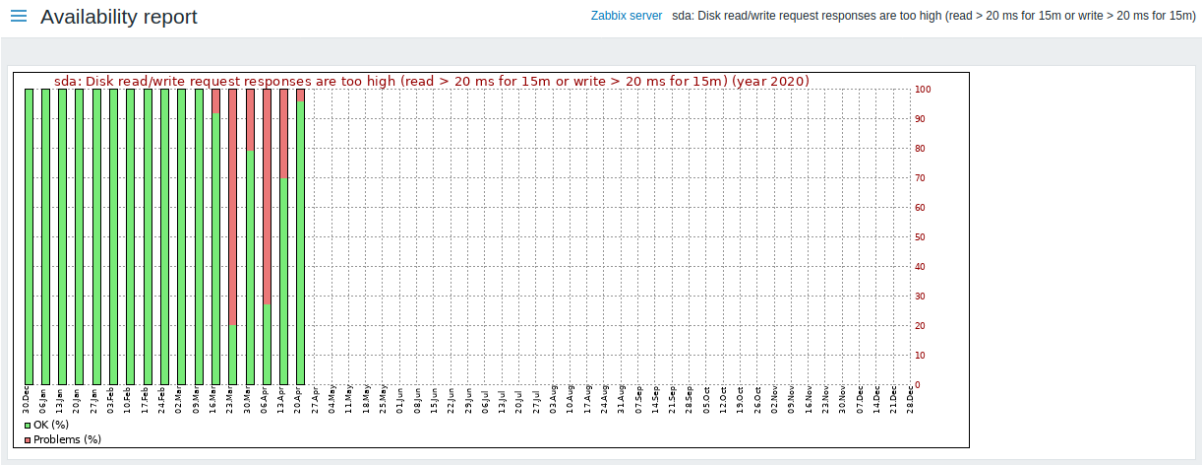
Filtering by host

In the *by host* mode results can be filtered by host or by 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.

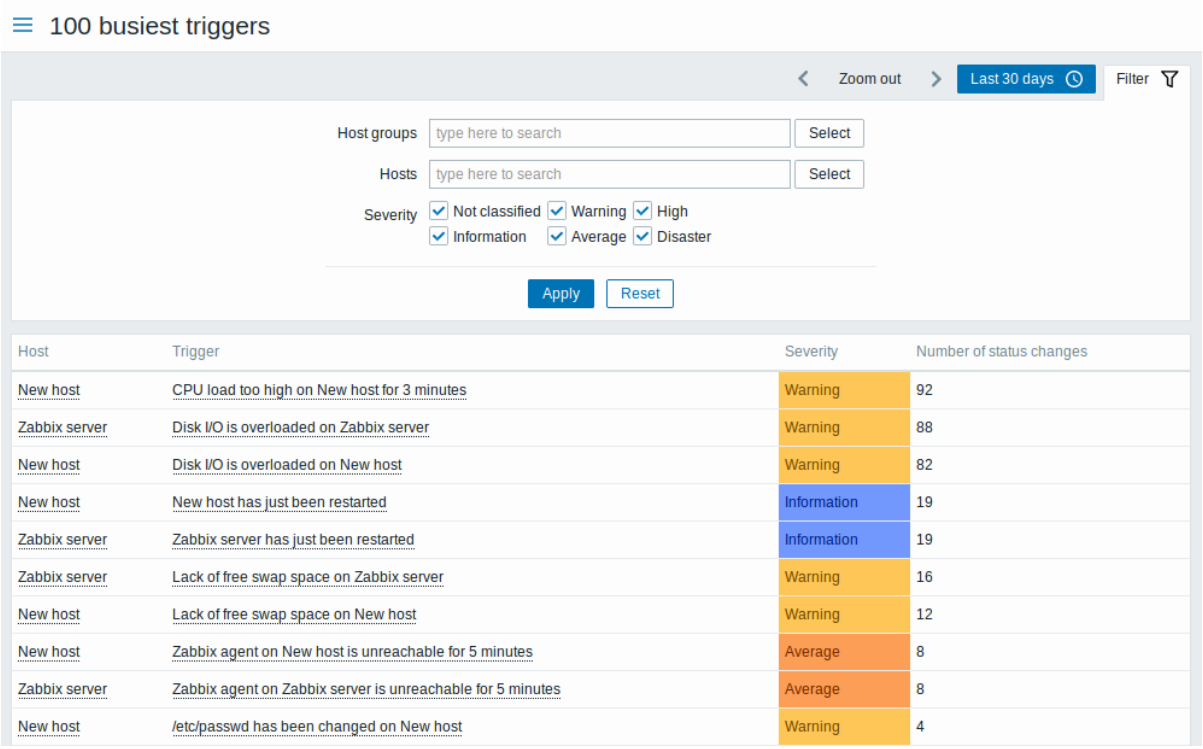


The green part of a bar stands for OK time and red for problem time.

3 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 screens 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.

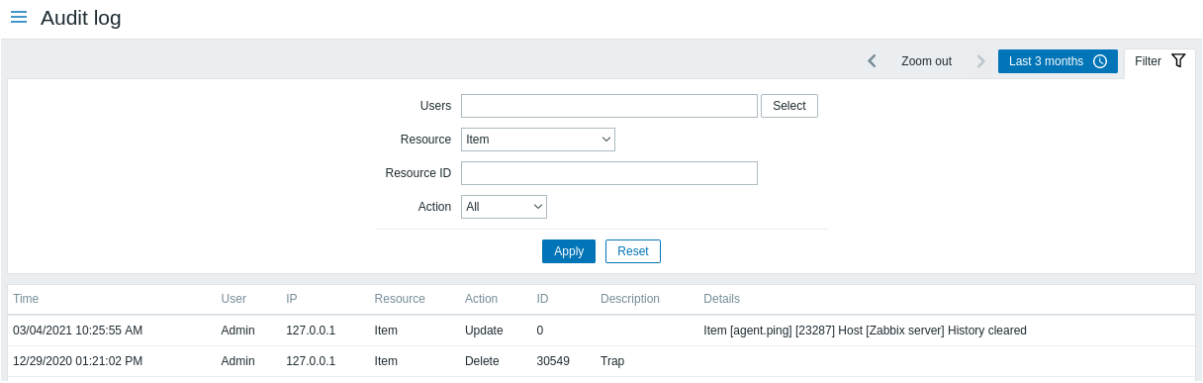
Time period selector

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.

4 Audit

Overview

In the *Reports* → *Audit* section, the records of user and system activity can be viewed.



Displayed data:

Column	Description
Time	Timestamp of the audit record.
User	User who performed the activity.
IP	IP from which the activity was initiated.

Column	Description
Resource	Type of the affected resource (*All*, *Action*, *Application*, *Autoregistration*, etc.).
Action	Type of the activity (*Add*, *Delete*, *Disable*, *Enable*, *Execute*, *Login*, *Logout*, *Update*).
ID	ID of the affected resource.
Description	Description of the resource is displayed.
Details	Detailed information on the performed activity is displayed.

Using filter

You may use the filter to narrow down the records by user, activity type and affected resource. For better search performance, data is searched with macros unresolved.

The filter is located below the *Audit log* bar. It can be opened and collapsed by clicking on the *Filter* tab on the left.

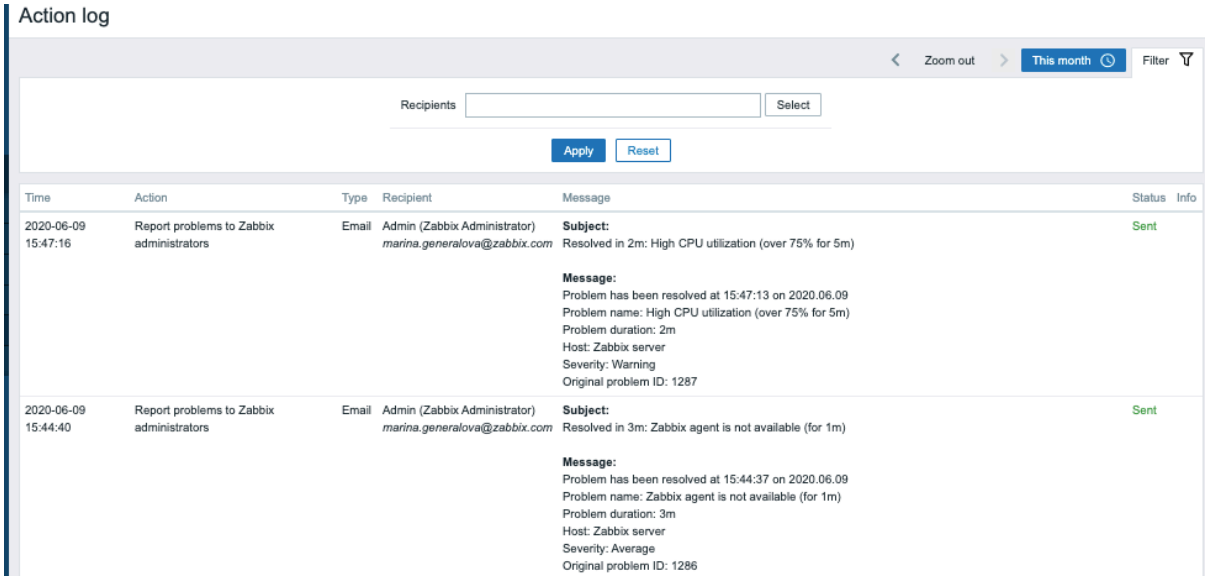
Time period selector

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.

5 Action log

Overview

In the Reports → Action log section users can view details of operations (notifications, remote commands) executed within an action.



Displayed data:

Column	Description
Time	Timestamp of the operation.
Action	Name of the action causing operations is displayed.
Type	Operation type is displayed - *Email* or *Command*.
Recipient(s)	User alias, name and surname (in parenthesis) and e-mail address of the notification recipient is displayed.
Message	The content of the message/remote command is displayed. A remote command is separated from the target host with a colon symbol: `<host>:<command>`. If the remote command is executed on Zabbix server, then the information has the following format: `Zabbix server:<command>`

Column	Description
Status	Operation status is displayed: *In progress* - action is in progress For actions in progress the number of retries left is displayed - the remaining number of times the server will try to send the notification. *Sent* - notification has been sent *Executed* - command has been executed *Not sent* - action has not been completed.
Info	Error information (if any) regarding the action execution is displayed.

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

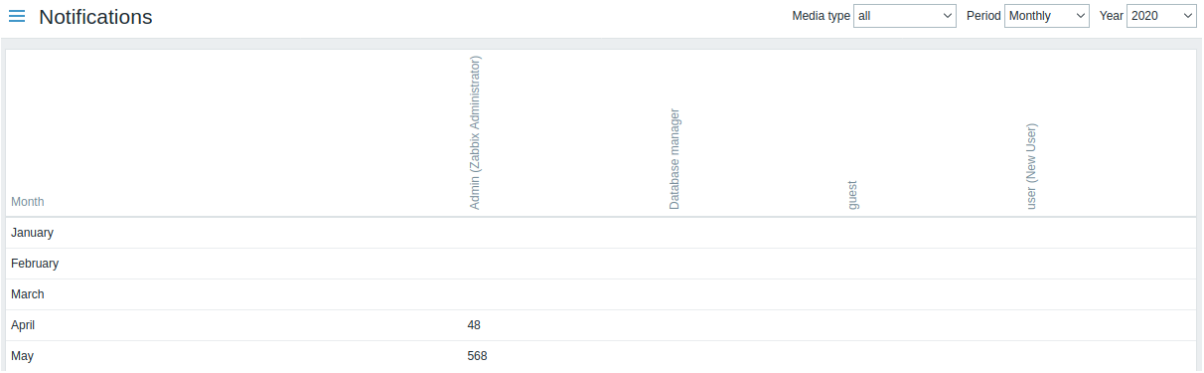
The time period selector allows to select often required periods with one mouse click. The time period selector can be opened by clicking on the time period tab next to the filter.

6 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.

4 Configuration

Overview

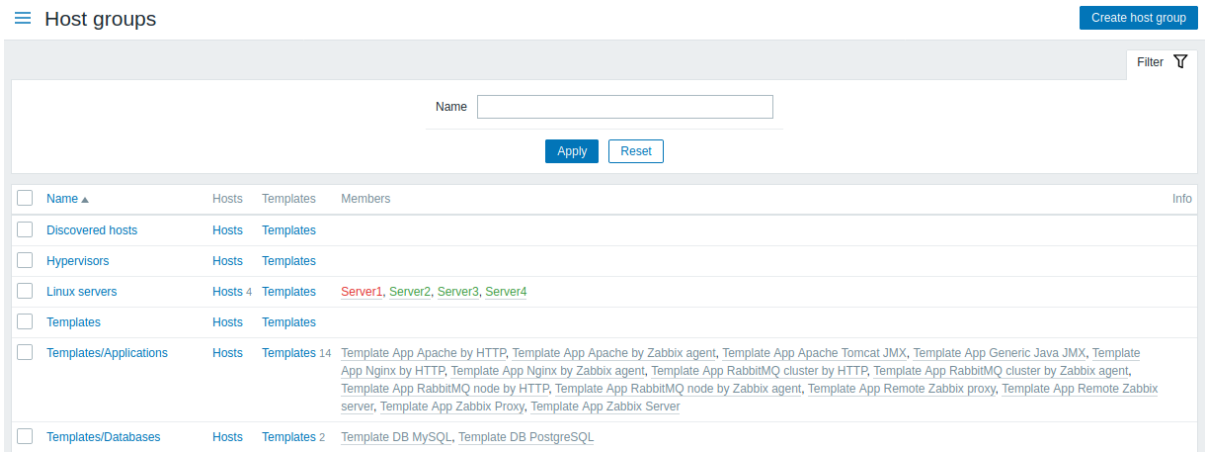
The Configuration menu contains sections for setting up major Zabbix functions, such as hosts and host groups, data gathering, data thresholds, sending problem notifications, creating data visualization and others.

1 Host groups

Overview

In the *Configuration* → *Host groups* section users can configure and maintain host groups. A host group can contain both templates and hosts.

A listing of existing host groups with their details is displayed. You can search and filter host groups by name.



Displayed data:

Column	Description
Name	Name of the host group. Clicking on the group name opens the host group configuration form.
Hosts	Number of hosts in the group (displayed in gray). Clicking on "Hosts" will, in the whole listing of hosts, filter out those that belong to the group.
Templates	Number of templates in the group (displayed in gray). Clicking on "Templates" will, in the whole listing of templates, filter out those that belong to the group.
Members	Names of group members. Template names are displayed in gray, monitored host names in blue and non-monitored host names in red. Clicking on a name will open the template/host configuration form.
Info	Error information (if any) regarding the host group is displayed.

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

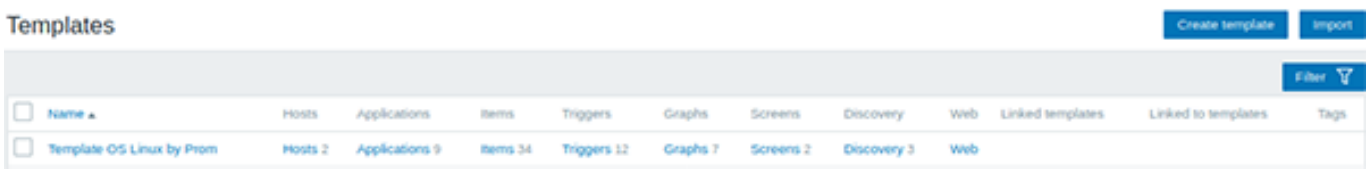
You can use the filter to display only the host groups you are interested in. For better search performance, data is searched with macros unresolved.

2 Templates

Overview

In the *Configuration* → *Templates* section users can configure and maintain templates.

A listing of existing templates with their details is displayed.

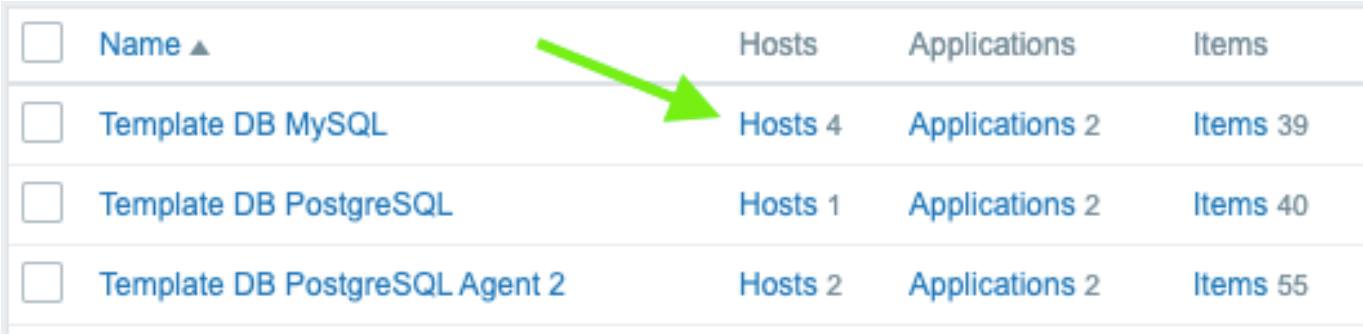


Displayed data:

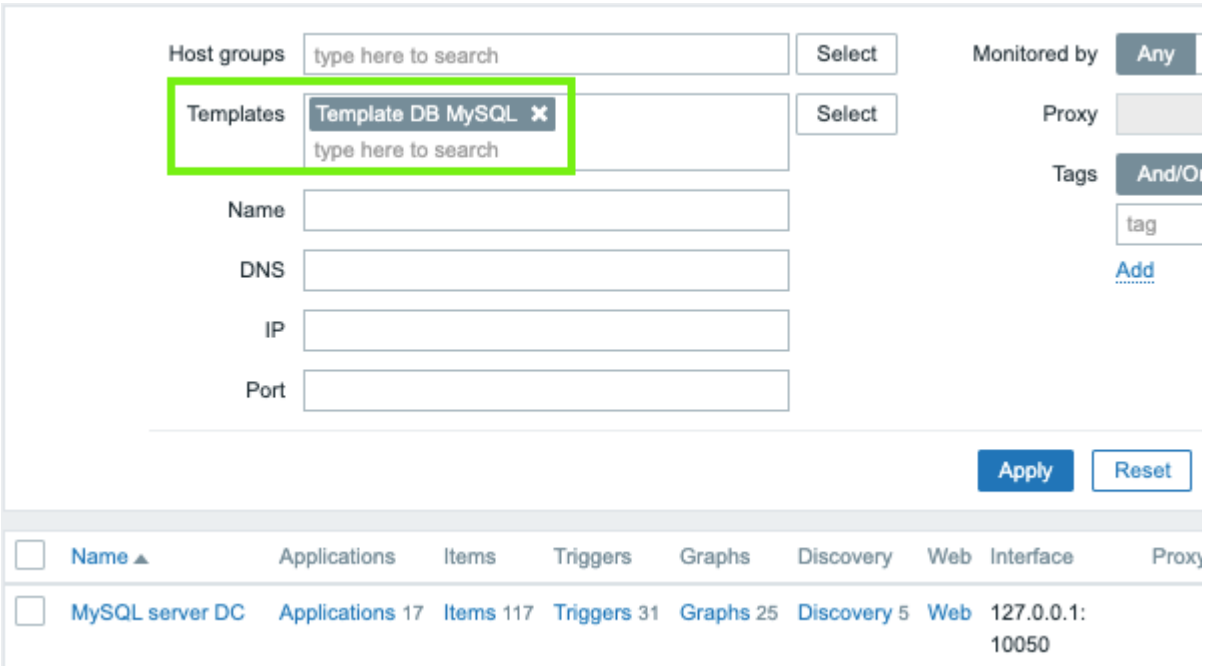
Column	Description
Templates	Name of the template. Clicking on the template name opens the template configuration form.

Column	Description
Hosts	Number of editable hosts to which the template is linked; read-only hosts are not included. Hosts column is available since Zabbix 5.0.3.
Entities (Applications, Items, Triggers, Graphs, Screens, Discovery, Web)	Number of the respective entities in the template (displayed in gray). Clicking on the entity name will, in the whole listing of that entity, filter out those that belong to the template.
Linked templates	Templates that are linked to the template, in a nested setup where the template will inherit all entities of the linked templates.
Linked to templates	The templates that the template is linked to ("children" templates that inherit all entities from this template). Since Zabbix 5.0.3, this column no longer includes hosts.
Tags	Tags of the template, with macros unresolved.

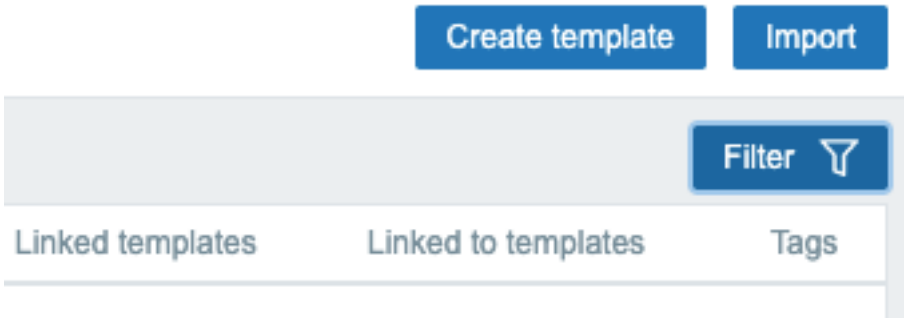
To view all hosts, connected to a template, click on the *Hosts* hyperlink.



This will open host configuration section filtered out by the template name (thus, only those hosts to which the template is linked to will be displayed):



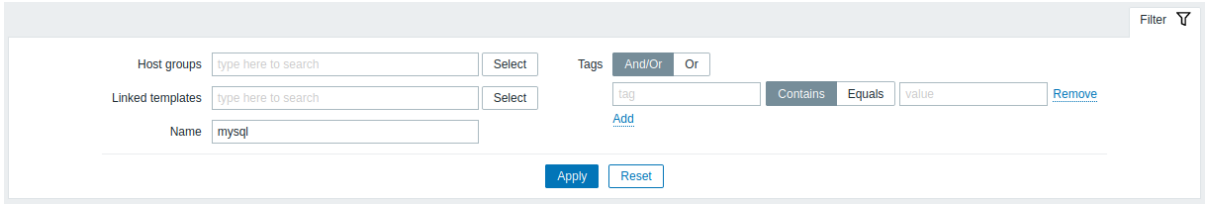
To configure a new template, click on the *Create template* button in the top right-hand corner. To import a template from an XML file, click on the *Import* button in the top right-hand corner.



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 host group, directly linked templates and name.



Filtering is possible only by template-level tags (not inherited ones).

Mass editing options

Buttons below the list offer some mass-editing options:

- *Export* - export the template to an XML 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

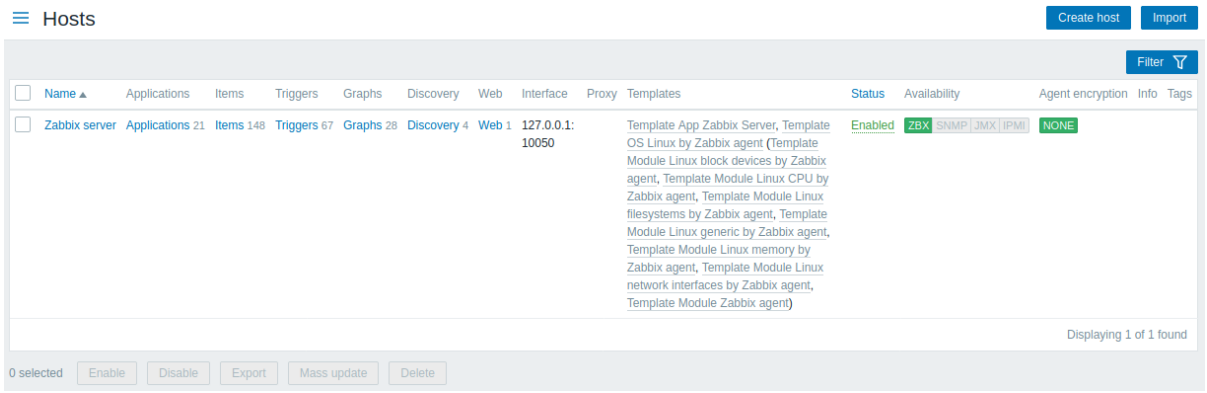
To use these options, mark the checkboxes before the respective templates, then click on the required button.

3 Hosts

Overview

In the *Configuration* → *Hosts* section users can configure and maintain hosts.

A listing of existing hosts with their details is displayed.



Displayed data:

Column	Description
Name	Name of the host. Clicking on the host name opens the host configuration form.
Elements (Applications, Items, Triggers, Graphs, Discovery, Web)	Clicking on the element name will display items, triggers etc. of the host. The number of the respective elements is displayed in gray.

Column	Description
Interface	The main interface of the host is displayed.
Proxy	Proxy name is displayed, if the host is monitored by a proxy. This column is only displayed if the *Monitored by* filter option is set to 'Any' or 'Proxy'.
Templates	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.
Status	Host status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it. An orange wrench icon before the host status indicates that this host is in maintenance. Maintenance details are displayed when the mouse pointer is positioned over the icon.
Availability	Availability of the host is displayed. Four icons each represent a supported interface (Zabbix agent, SNMP, IPMI, JMX). The current status of the interface is displayed by the respective color: **Green** - available **Red** - not available (upon mouseover, details of why the interface cannot be reached are displayed) **Gray** - unknown or not configured Note that active Zabbix agent items do not affect host availability.
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 an XML file, click on the *Import* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

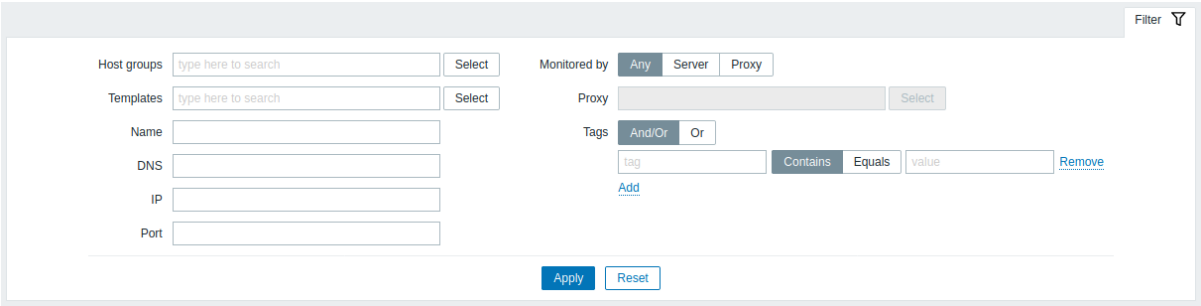
- *Enable* - change host status to *Monitored*
- *Disable* - change host status to *Not monitored*
- *Export* - export the hosts to an XML file
- *Mass update* - update several properties for a number of hosts at once
- *Delete* - delete the hosts

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, if they are monitored by server or by proxy, proxy name, DNS, IP or port number.



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 host status

Zabbix server sets the host availability icon to gray (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 host availability to unknown is done after server configuration cache synchronization. Restoring host availability (available/unavailable) on hosts monitored by proxies is done after proxy configuration cache synchronization.

See also more details about host unreachability.

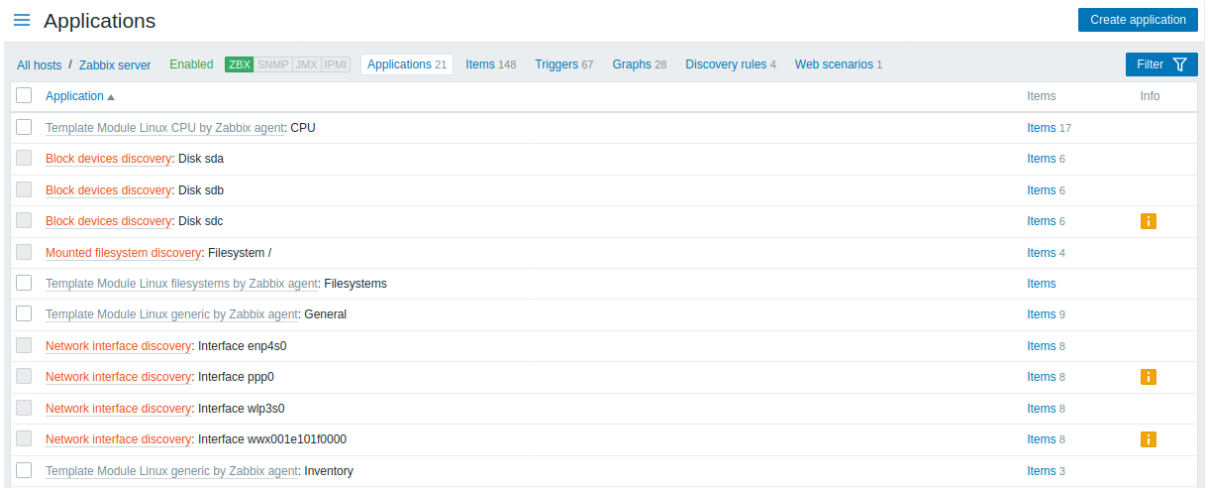
1 Applications

Overview

The application list for a template can be accessed from *Configuration* → *Templates* and then clicking on Applications for the respective template.

The application list for a host can be accessed from *Configuration* → *Hosts* and then clicking on Applications for the respective host.

A list of existing applications is displayed.



Displayed data:

Column	Description
Application	Name of the application, displayed as a blue link for directly created applications. Clicking on the application name link opens the application configuration form. If the host application belongs to a template, the template name is displayed before the application name, as a gray link. Clicking on the template link will open the application list on the template level.
Items	Click on Items to view the items contained in the application. The number of items is displayed in gray.
Info	Error information (if any) regarding the application is displayed.

To configure a new application, click on the *Create application* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Enable* - change application status to *Enabled*
- *Disable* - change application status to *Disabled*
- *Delete* - delete the applications

To use these options, mark the checkboxes before the respective applications, then click on the required button.

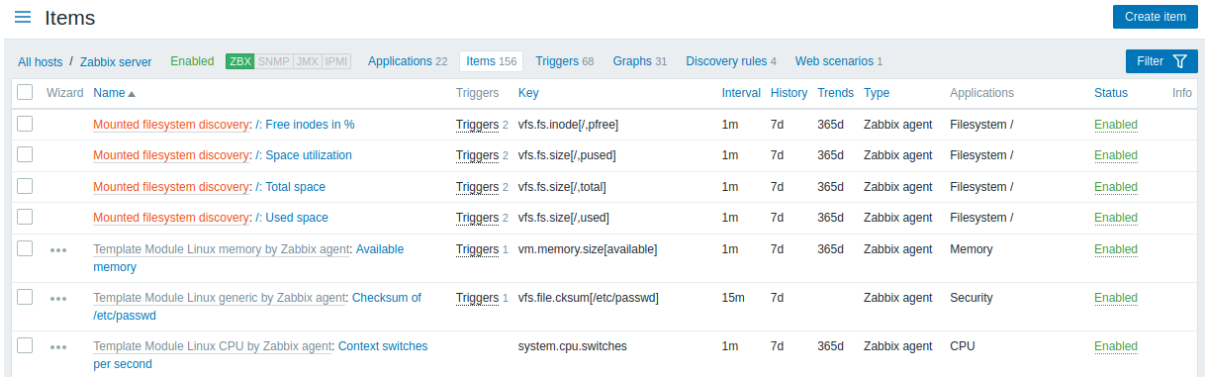
2 Items

Overview

The item list for a template can be accessed from *Configuration* → *Templates* and then clicking on Items for the respective template.

The item list for a host can be accessed from *Configuration* → *Hosts* and then clicking on Items for the respective host.

A list of existing items is displayed.



Displayed data:

Column	Description
Wizard	The wizard icon is a link to a wizard for creating a trigger based on the 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 info box 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 *Check 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).
Applications	Item applications are displayed.
Status	Item 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).
Info	If everything is fine, no icon is displayed in this column. If there are errors, a red square icon with a cross is displayed. Move the mouse over the icon and you will see a tooltip with the error description.

To configure a new item, click on the *Create item* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

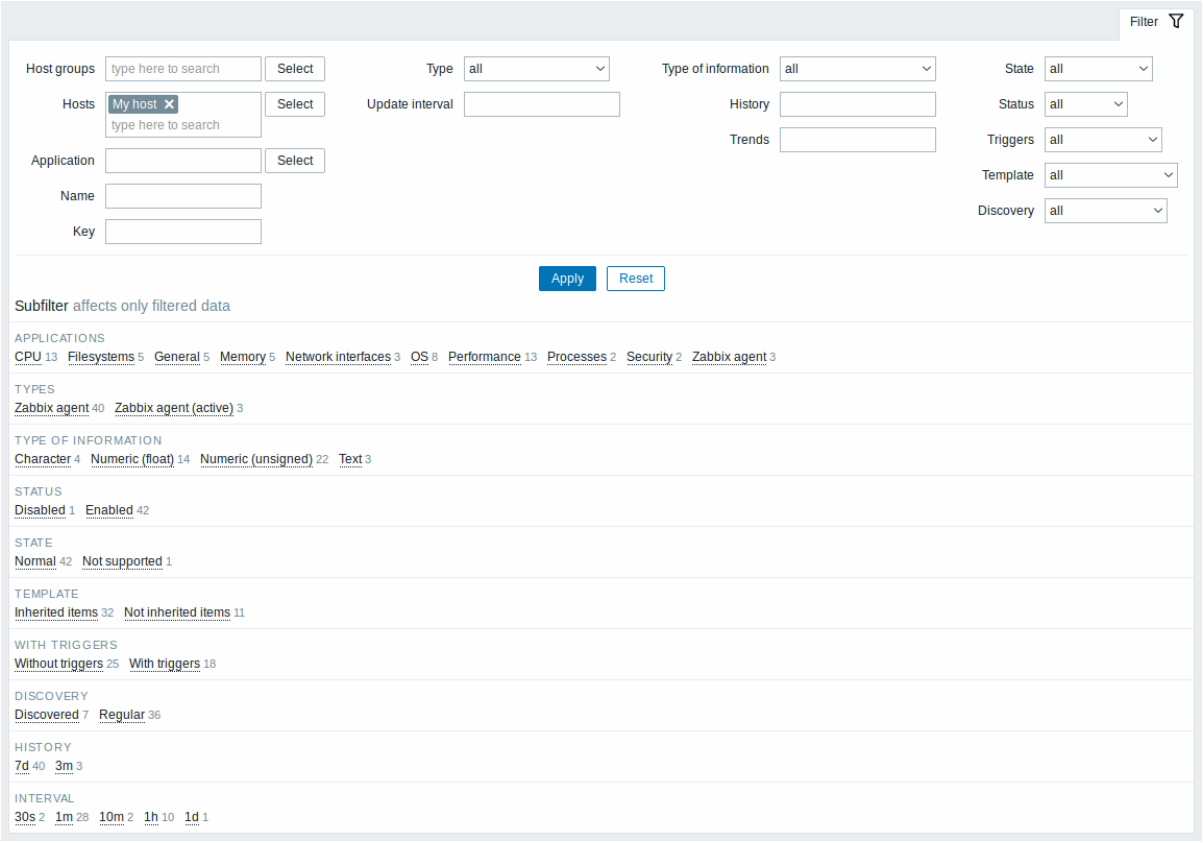
- *Enable* - change item status to *Enabled*
- *Disable* - change item status to *Disabled*
- *Check now* - execute a check for new item values immediately. Supported for **passive** checks only (see more details). Note that when checking for values immediately, configuration cache is not updated, thus 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* link is available above the list. If you click on it, a filter becomes available where you can filter items by several properties.



Parameter	Description
Host groups	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups.
Hosts	Filter by one or more hosts.
Application	Filter by application.
Name	Filter by item name.
Key	Filter by item key.
Type	Filter by item type (Zabbix agent, SNMP agent, etc.).
Update interval	Filter by item update interval.
Type of information	Filter by type of information (Numeric unsigned, float, etc.).
History	Filter by how long item history is kept.
Trends	Filter by how long item trends are kept.
State	Filter by item state - *Normal* or *Not supported*.

Parameter	Description
Status	Filter by item status - *Enabled* or *Disabled*.
Triggers	Filter items with (or without) triggers.
Template	Filter items inherited (or not inherited) from a template.
Discovery	Filter items discovered (or not discovered) by low-level discovery.

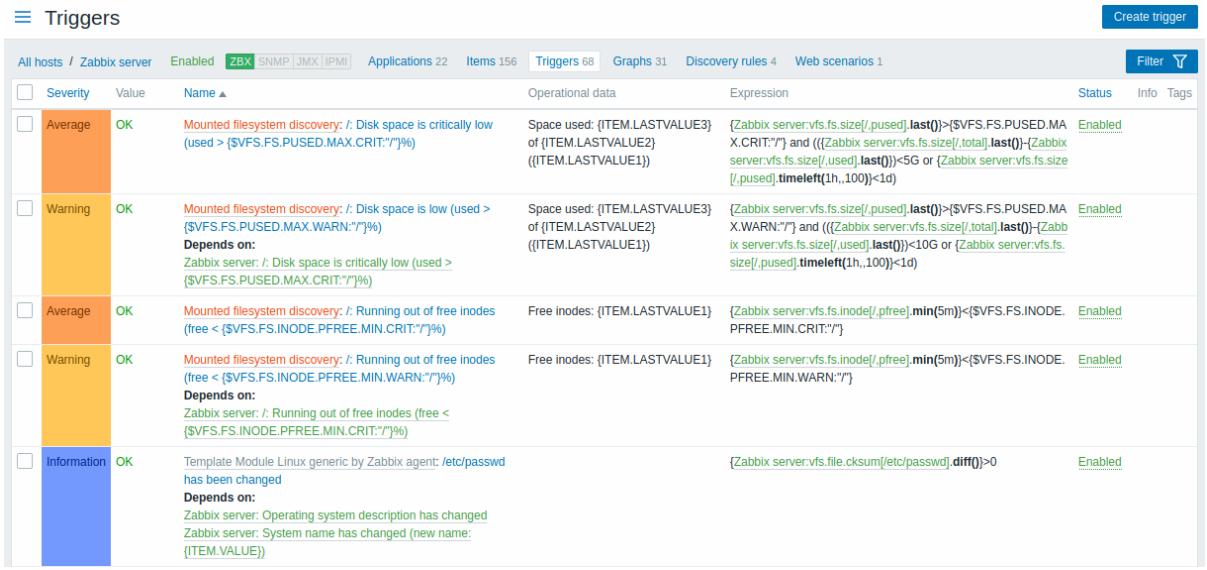
The **Subfilter** below the filter offers further filtering options (for the data already filtered). You can select groups of items with a common parameter value. If you click on a group it gets highlighted and only the items with this parameter value remain in the list.

3 Triggers

Overview

The trigger list for a template can be accessed from *Configuration* → *Templates* and then clicking on Triggers for the respective template.

The trigger list for a host can be accessed from *Configuration* → *Hosts* and then clicking on Triggers for the respective host.



Displayed data:

Column	Description
Severity	Severity of the trigger is displayed by both name and cell background color.
Value	Trigger value is displayed: **OK** - trigger is in OK state **PROBLEM** - trigger is in problem state
Host	Host of the trigger. This column is displayed only if multiple hosts are selected in the filter.
Name	Name of the trigger, displayed as a blue link to trigger details. Clicking on the trigger name link opens the trigger configuration form. 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. 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.
Operational data	Operational data definition of the trigger, containing arbitrary strings and macros that will resolve dynamically in *Monitoring* → *Problems*.
Expression	Trigger expression is displayed. The host-item part of the expression is displayed as a link, leading to the item configuration form.

Column	Description
Status	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.
Info	If everything is fine, no icon is displayed in this column. If there are errors, a red square icon with a cross is displayed. Move the mouse over the icon and you will see a tooltip with the error description.
Tags	If trigger contains tags, tag name and value are displayed in this column.

To configure a new trigger, click on the *Create trigger* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

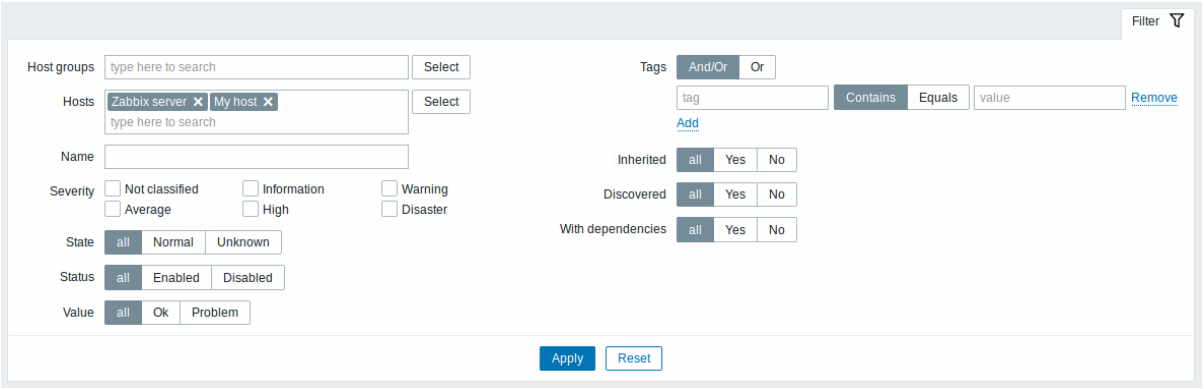
- *Enable* - change trigger status to *Enabled*
- *Disable* - change trigger status to *Disabled*
- *Copy* - copy the triggers to other hosts or templates
- *Mass update* - update several properties for a number of triggers at once
- *Delete* - delete the triggers

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 is located above the table.



Parameter	Description
Host groups	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups.
Hosts	Filter by one or more hosts. If host groups are already selected above, host selection is limited to those groups.
Name	Filter by trigger name.
Severity	Select to filter by one or several trigger severities.
State	Filter by trigger state.
Status	Filter by trigger status.
Value	Filter by trigger value.

Parameter	Description
Tags	Filter by trigger tag name and tag value. Several conditions can be set. There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having same tag name will be grouped by Or condition **Or** - enough if one condition is met 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) When filtered, the tags specified here will be displayed first with the trigger. 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.

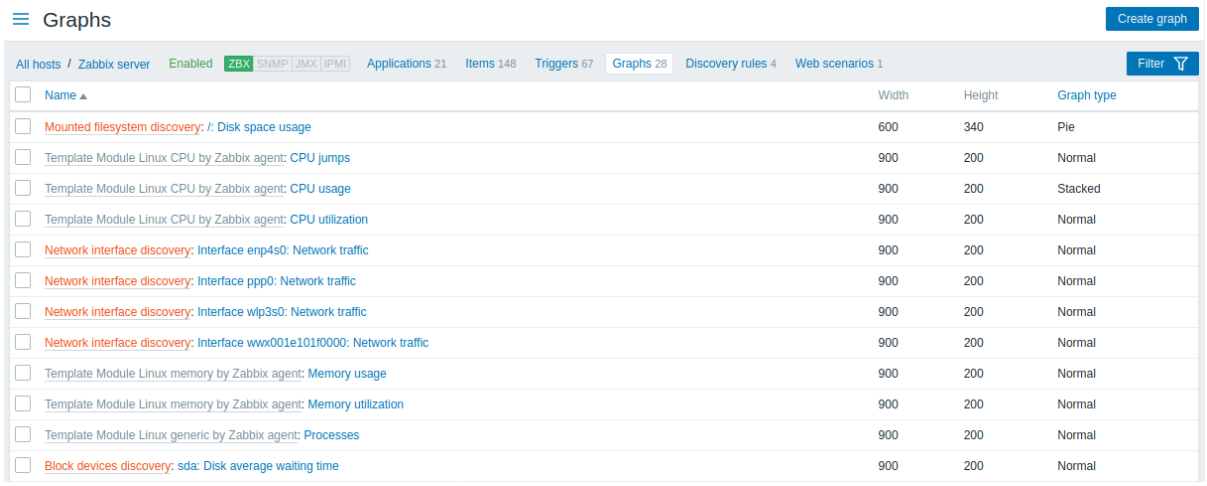
4 Graphs

Overview

The custom graph list for a template can be accessed from *Configuration* → *Templates* and then clicking on Graphs for the respective template.

The custom graph list for a host can be accessed from *Configuration* → *Hosts* and then clicking on Graphs for the respective host.

A list of existing graphs is displayed.



Displayed data:

Column	Description
Name	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 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.
Width	Graph width is displayed.
Height	Graph height is displayed.
Graph type	Graph type is displayed - *Normal*, *Stacked*, *Pie* or *Exploded*.

To configure a new graph, click on the *Create graph* button in the top right-hand 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 template/host. For better search performance, data is searched with macros unresolved.

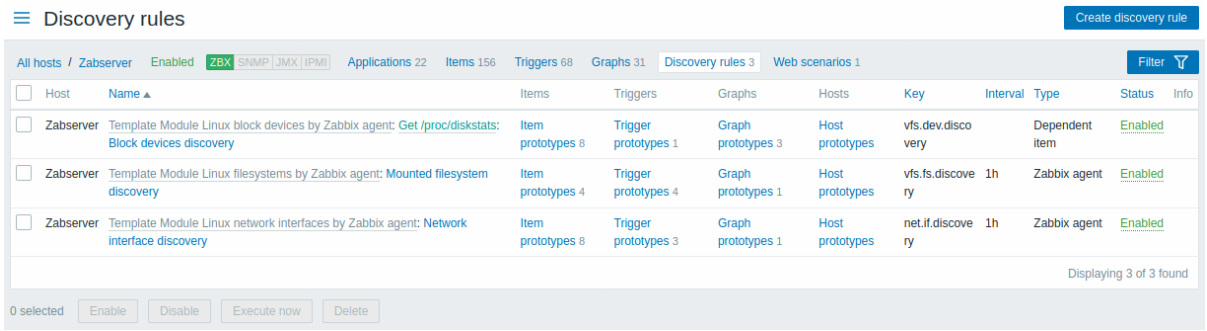
5 Discovery rules

Overview

The list of low-level discovery rules for a template can be accessed from *Configuration* → *Templates* and then clicking on Discovery for the respective template.

The list of low-level discovery rules for a host can be accessed from *Configuration* → *Hosts* and then 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 template/host, or all discovery rules of a specific host group by making selections in the filter.



Displayed data:

Column	Description
Host	The visible host name is displayed. In the absence of a visible host name, the technical host name is displayed.
Name	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.
Items	A link to the list of item prototypes is displayed. The number of existing item prototypes is displayed in gray.
Triggers	A link to the list of trigger prototypes is displayed. The number of existing trigger prototypes is displayed in gray.
Graphs	A link to the list of graph prototypes displayed. The number of existing graph prototypes is displayed in gray.
Hosts	A link to the list of host prototypes displayed. The number of existing host prototypes is displayed in gray.
Key	The item key used for discovery is displayed.
Interval	The frequency of performing discovery is displayed. *Note* that discovery can also be performed immediately by pushing the *Check now* button below the list.
Type	The item type used for discovery is displayed (Zabbix agent, SNMP agent, etc).
Status	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).

Column	Description
Info	If everything is fine, no icon is displayed in this column. If there are errors, a red square icon with a cross is displayed. Move the mouse over the icon and you will see a tooltip with the error description.

To configure a new low-level 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:

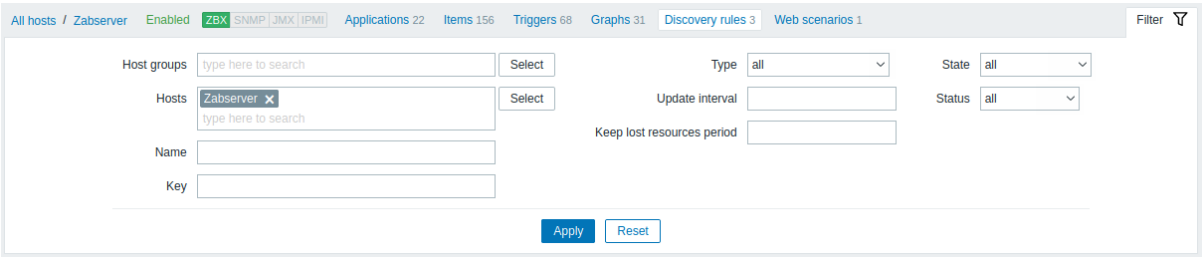
- *Enable* - change the low-level discovery rule status to *Enabled*
- *Disable* - change the low-level discovery rule status to *Disabled*
- *Check now* - perform discovery based on the discovery rules immediately. See more details. Note that when performing discovery immediately, the configuration 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.



Parameter	Description
Host groups	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups.
Hosts	Filter by one or more hosts.
Name	Filter by discovery rule name.
Key	Filter by discovery item key.
Type	Filter by discovery item type.
Update interval	Filter by update interval. Not available for Zabbix trapper and dependent items.
Keep lost resources period	Filter by keep lost resources period.
SNMP OID	Filter by SNMP OID. Only available if *SNMP agent* is selected as type.
State	Filter by discovery rule state (all/normal/unsupported).
Status	Filter by discovery rule status (all/enabled/disabled).

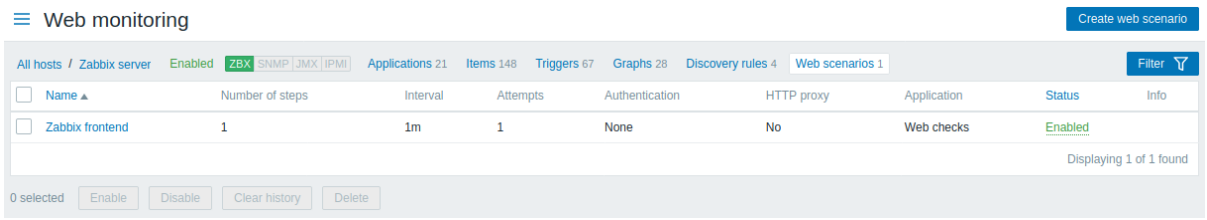
6 Web scenarios

Overview

The web scenario list for a template can be accessed from *Configuration* → *Templates* and then clicking on Web for the respective template.

The web scenario list for a host can be accessed from *Configuration* → *Hosts* and then clicking on Web for the respective host.

A list of existing web scenarios is displayed.



Displayed data:

Column	Description
Name	Name of the web scenario. Clicking on the web scenario name opens the web scenario configuration form.
Number of steps	The number of steps contained in the scenario.
Update interval	How often the scenario is performed.
Attempts	How many attempts for executing web scenario steps are performed.
Authentication	Authentication method is displayed - Basic, NTLM on None.
HTTP proxy	Displays HTTP proxy or 'No' if not used.
Application	Web scenario application is displayed.
Status	Web scenario status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.
Info	If everything is fine, no icon is displayed in this column. If there are errors, a red square icon with a cross is displayed. Move the mouse over the icon and you will see a tooltip with the error description.

To configure a new web scenario, click on the *Create web scenario* button in the top right-hand 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.

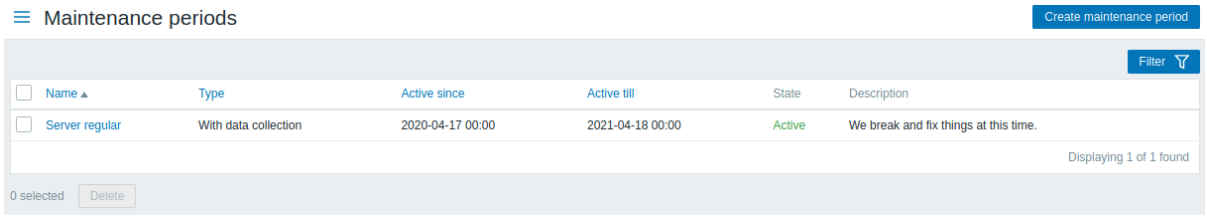
The *Filter* link is available above the list of web scenarios. If you click on it, a filter becomes available where you can filter scenarios by host group, host and status.

4 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:

Column	Description
Name	Name of the maintenance period. Clicking on the maintenance period name opens the maintenance period configuration form.
Type	The type of maintenance is displayed: *With data collection* or *No data collection*
Active since	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.
Active till	The date and time when executing maintenance periods stops being active.
State	The state of the maintenance period: **Approaching** - will become active soon **Active** - is active **Expired** - is not active any more
Description	Description of the maintenance period is displayed.

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:

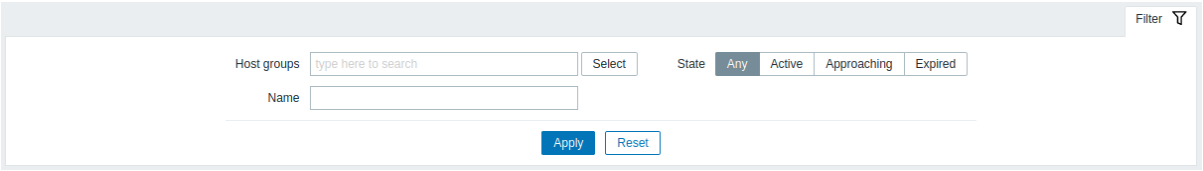
- *Delete* - delete the maintenance periods

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.



5 Actions

Overview

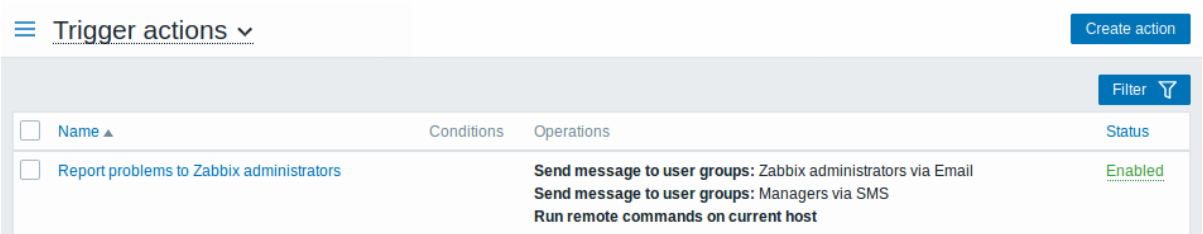
In the *Configuration* → *Actions* section users can configure and maintain actions.

A listing of existing actions with their details is displayed. The actions displayed are actions assigned to the selected event source (trigger, discovery, autoregistration actions).

To view actions assigned to a different event source, change the source from the title dropdown.

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 fulfillled:

- 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



Displayed data:

Column	Description
Name	Name of the action. Clicking on the action name opens the action configuration form.
Conditions	Action conditions are displayed.
Operations	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 alias) of a notification recipient. Action operation can both be a notification or a remote command depending on the selected type of operation.
Status	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.

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:

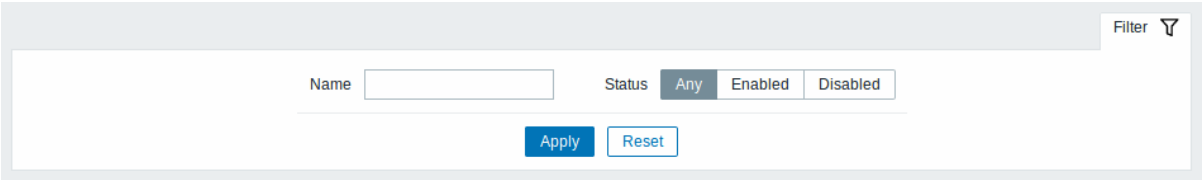
- *Enable* - change the action status to *Enabled*
- *Disable* - change the action status to *Disabled*
- *Delete* - delete the actions

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.

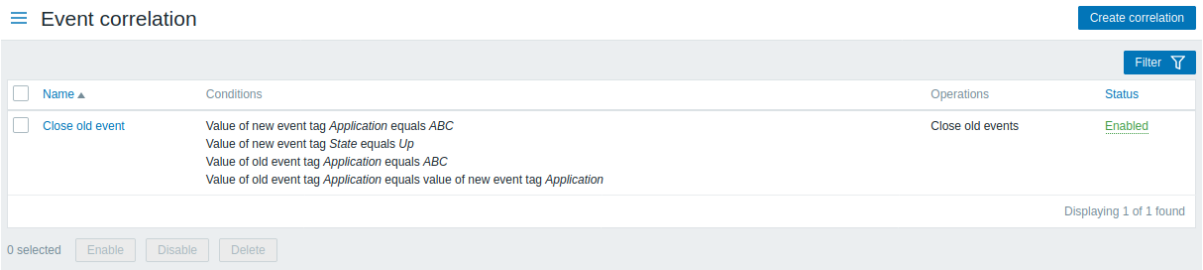
The *Filter* link is available above the list of actions. If you click on it, a filter becomes available where you can filter actions by name and status.



6 Event correlation

Overview

In the *Configuration* → *Event correlation* section users can configure and maintain global correlation rules for Zabbix events.



Displayed data:

Column	Description
Name	Name of the correlation rule. Clicking on the correlation rule name opens the rule configuration form.
Conditions	Correlation rule conditions are displayed.

Column	Description
Operations	Correlation rule operations are displayed.
Status	Correlation rule status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.

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:

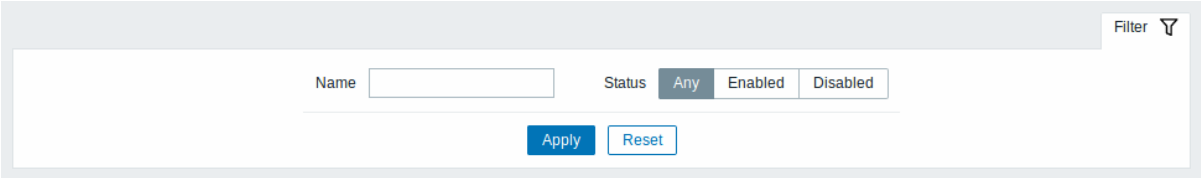
- *Enable* - change the correlation rule status to *Enabled*
- *Disable* - change the correlation rule status to *Disabled*
- *Delete* - delete the correlation rules

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.

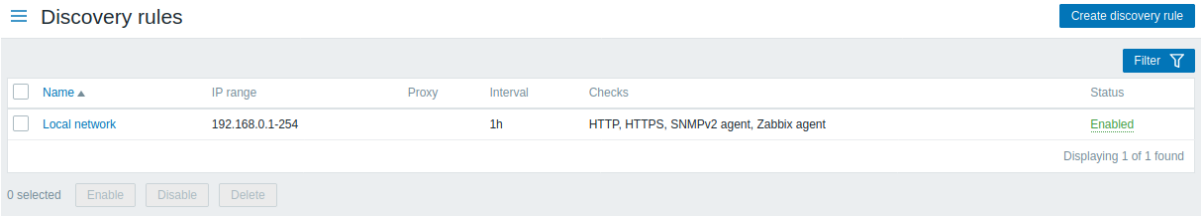


7 Discovery

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:

Column	Description
Name	Name of the discovery rule. Clicking on the discovery rule name opens the discovery rule configuration form.
IP range	The range of IP addresses to use for network scanning is displayed.
Proxy	The proxy name is displayed, if discovery is performed by the proxy.
Interval	The frequency of performing discovery displayed.
Checks	The types of checks used for discovery are displayed.
Status	Action status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.

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*

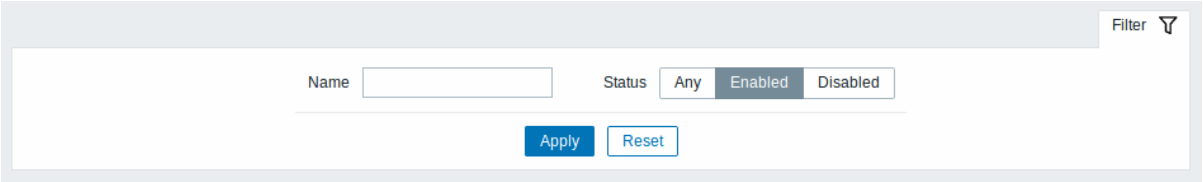
- *Disable* - change the discovery rule status to *Disabled*
- *Delete* - delete the discovery rules

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.



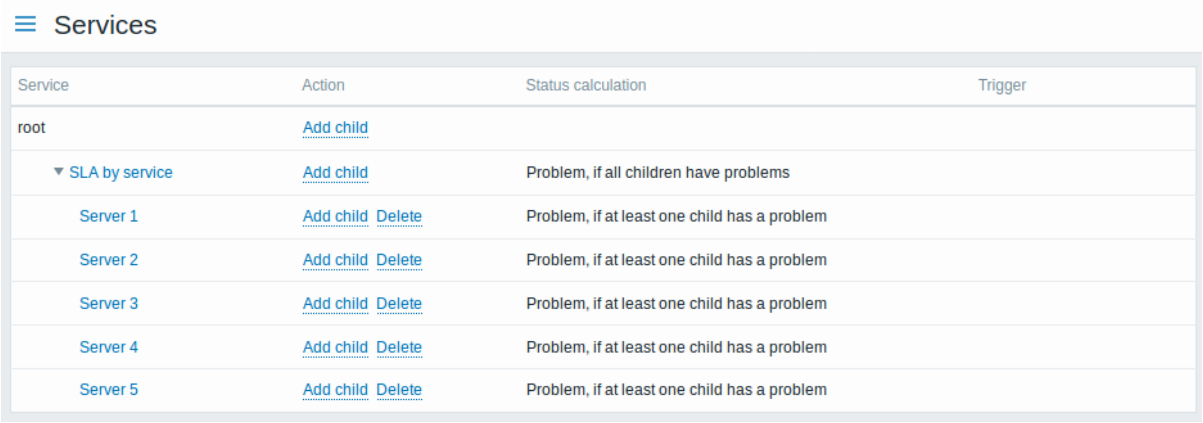
8 Services

Overview

In the *Configuration* → *Services* section users can configure and maintain an IT services hierarchy.

When you first open this section it only contains a *root* entry.

You can use it as a starting point of building the hierarchy of monitored infrastructure. Click on *Add child* to add services and then other services below the ones you have added.



For details on adding services, see the Service monitoring section.

5 Administration

Overview

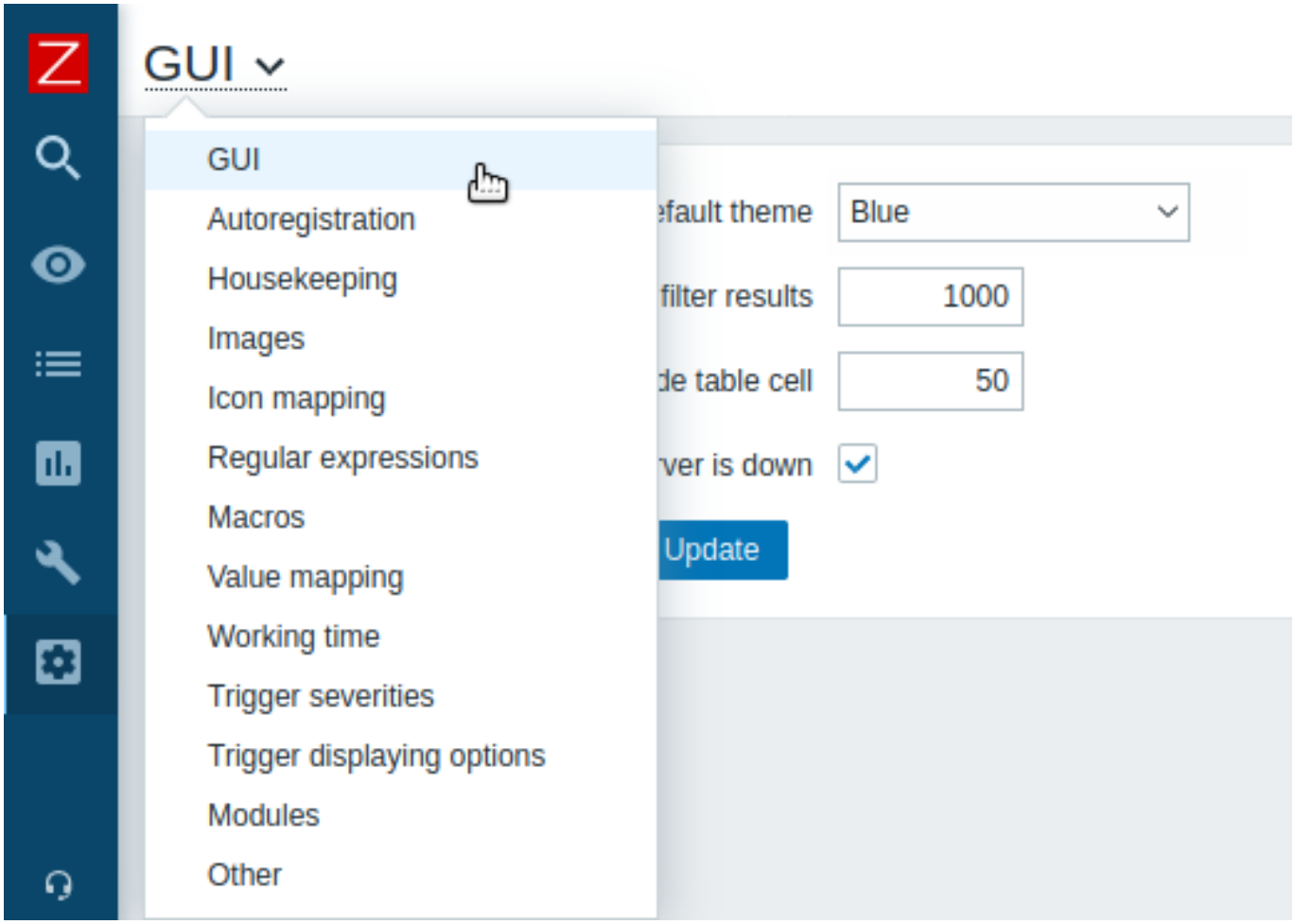
The Administration menu is for administrative functions of Zabbix. This menu is available to users of Super Administrators type only.

1 General

Overview

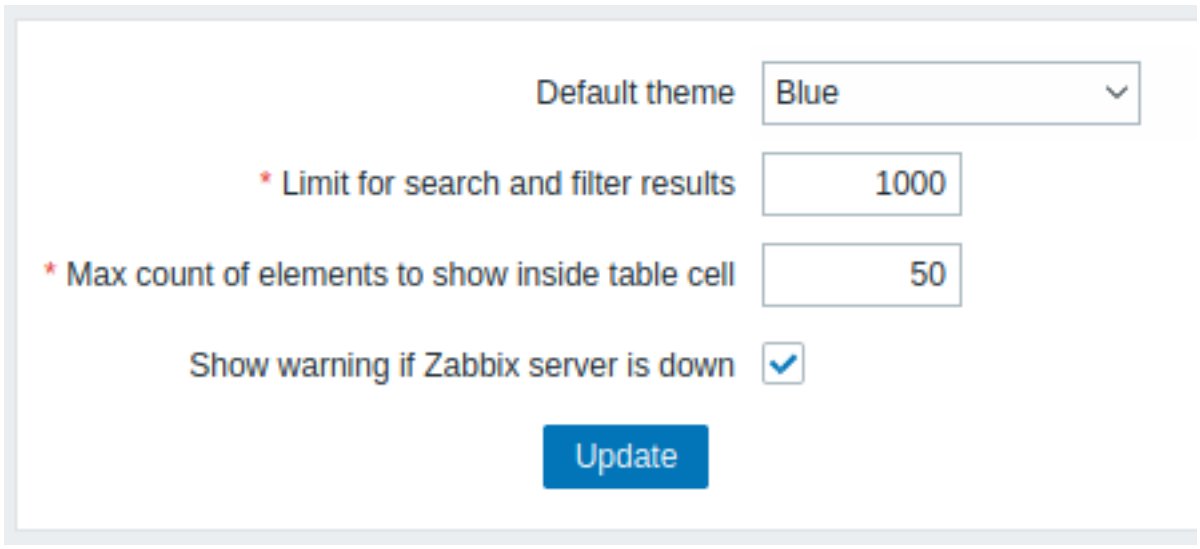
The *Administration* → *General* section contains a number of screens for setting frontend-related defaults and customizing Zabbix.

The title dropdown allows you to switch between different administration screens.



1 GUI

This screen provides customization of several frontend-related defaults.



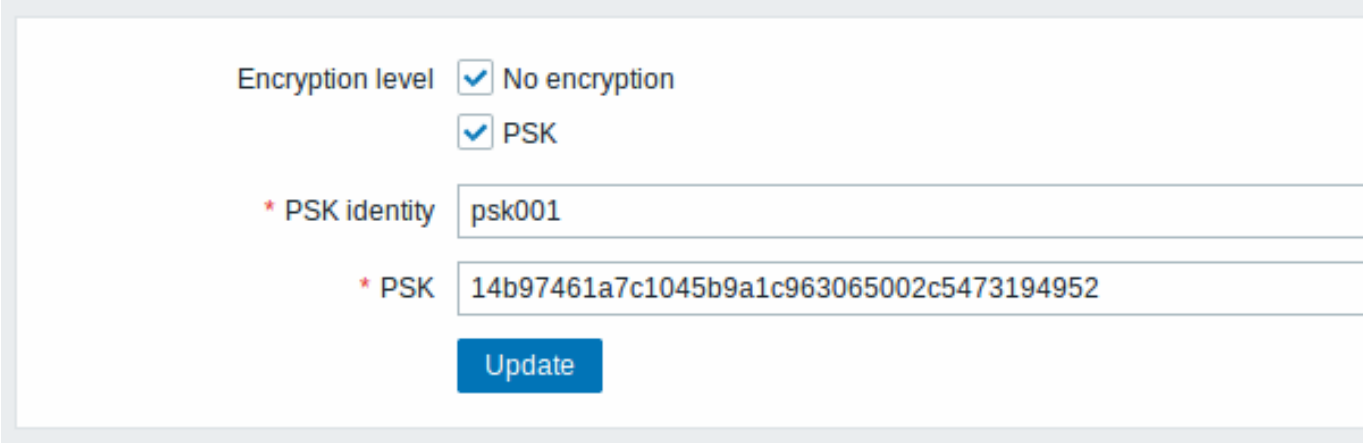
Configuration parameters:

Parameter	Description
Default theme	Default theme for users who have not set a specific one in their profiles.

Parameter	Description
Limit for search and filter results	Maximum amount of elements (rows) that will be displayed in a web-interface list, like, 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 *"Displaying 1 to 50 of **50+** found"*. Also, if filtering is used and still there are more than 50 matches, only the first 50 will be displayed.
Max count of elements to show inside table cell	For entries that are displayed in a single table cell, no more than configured here will be shown.
Show warning if Zabbix server is down	This parameter enables a warning message to be displayed in the browser window if Zabbix server cannot be reached (may be down). The message remains visible even if the user scrolls down the page. If the mouse is moved over it, the message is temporarily hidden to reveal the contents below. This parameter is supported since Zabbix 2.0.1.

2 Autoregistration

In this screen you can configure the encryption level for active agent autoregistration.



Parameters marked with an asterisk are mandatory.

Configuration parameters:

Parameter	Description
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*.

See also: Secure autoregistration

3 Housekeeper

The housekeeper is a periodical process, executed by Zabbix server. The process removes outdated information and information deleted by user.

Events and alerts

Enable internal housekeeping ☑

* Trigger data storage period: 365d

* Internal data storage period: 1d

* Network discovery data storage period: 1d

* Auto-registration data storage period: 1d

Services

Enable internal housekeeping ☑

* Data storage period: 365d

Audit

Enable internal housekeeping ☑

* Data storage period: 365d

User sessions

Enable internal housekeeping ☑

* Data storage period: 365d

History

Enable internal housekeeping ☑

Override item history period ☐

* Data storage period: 90d

Trends

Enable internal housekeeping ☑

Override item trend period ☐

* Data storage period: 365d

Update | Reset defaults

In this section housekeeping tasks can be enabled or disabled on a per-task basis separately for: events and alerts/IT services/audit/user sessions/history/trends. 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.

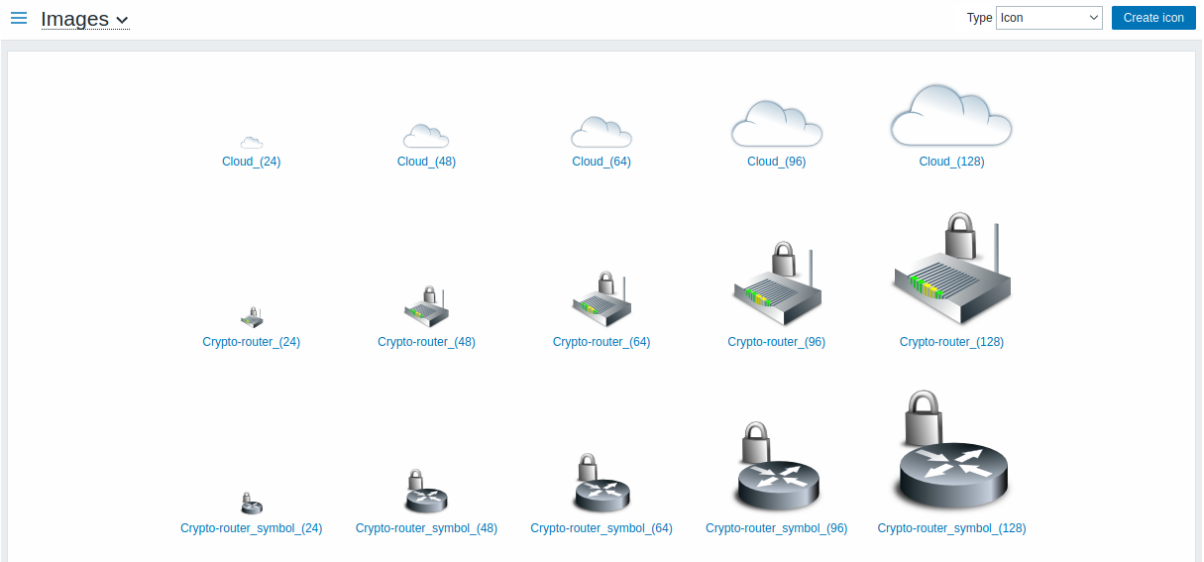
Attention:
If using TimescaleDB, in order to take full advantage of TimescaleDB automatic partitioning of history and trends tables, *Override item history period* and *Override item trend period* options must be enabled as well as *Enable internal housekeeping* option for history and trends. Otherwise, data kept in these tables will still be stored in partitions, however, the housekeeper will not drop outdated partitions, and warnings about incorrect configuration will be displayed. 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). Minimum is 1 day (1 hour for history), maximum 25 years.

Reset defaults button allows to revert any changes made.

4 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.

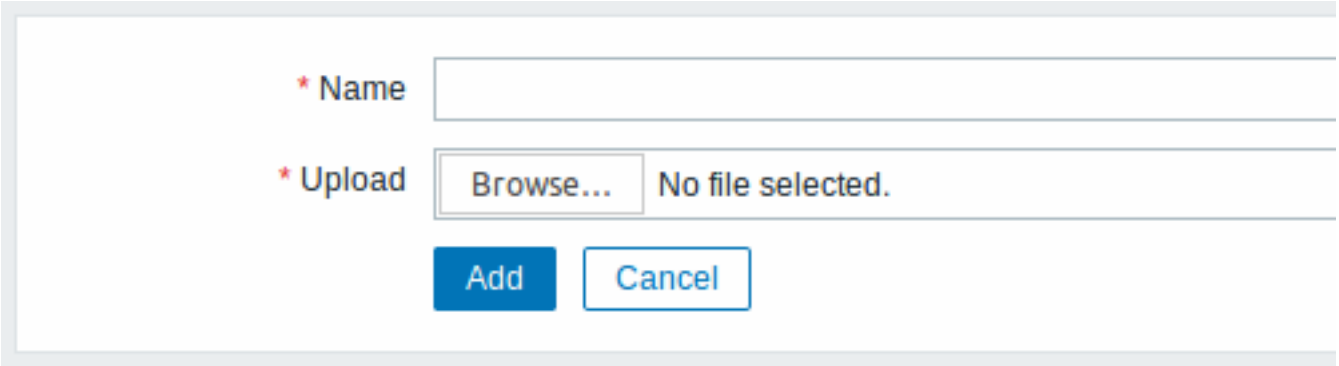


Image attributes:

Parameter	Description
Name	Unique name of an image.
Upload	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).

Note:
Maximum size of the upload file is limited by value of ZBX_MAX_IMAGE_SIZE that is 1024x1024 bytes or 1 MB.

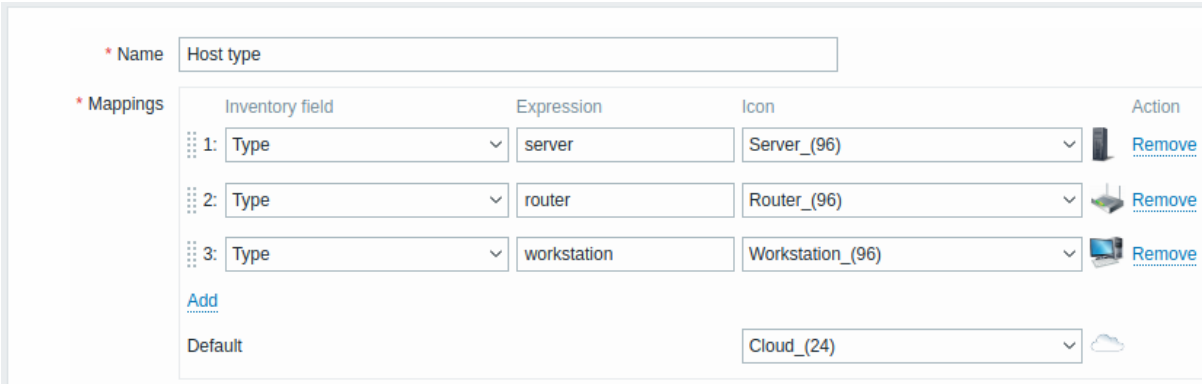
The upload of an image may fail if the image size is close to 1 MB and the `max_allowed_packet` MySQL configuration parameter is at a default of 1MB. In this case, increase the max_allowed_packet parameter.

5 Icon mapping

This section allows to create 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:

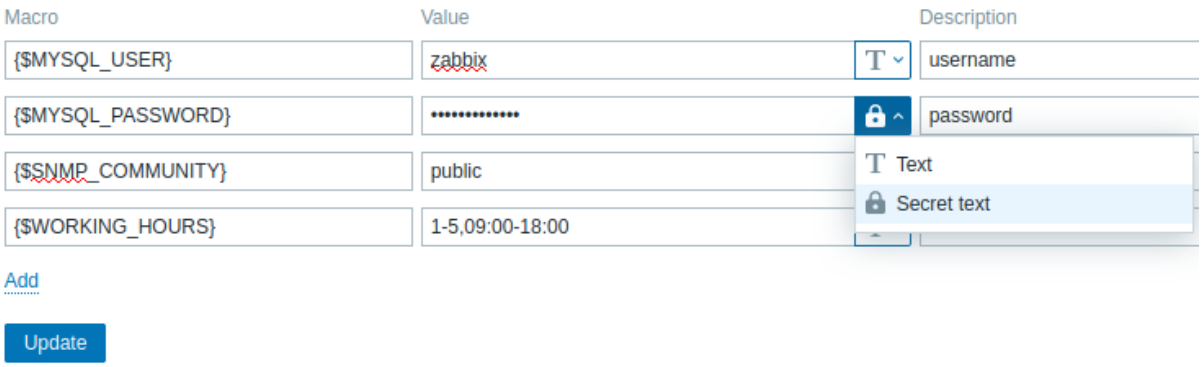
Parameter	Description
Name	Unique name of icon map.
Mappings	A list of mappings. The order of mappings determines which one will have priority. You can move mappings up and down the list with drag-and-drop.
Inventory field	Host inventory field that will be looked into to seek a match.
Expression	Regular expression describing the match.
Icon	Icon to use if a match for the expression is found.
Default	Default icon to use.

6 Regular expressions

This section allows to create custom regular expressions that can be used in several places in the frontend. See Regular expressions section for details.

7 Macros

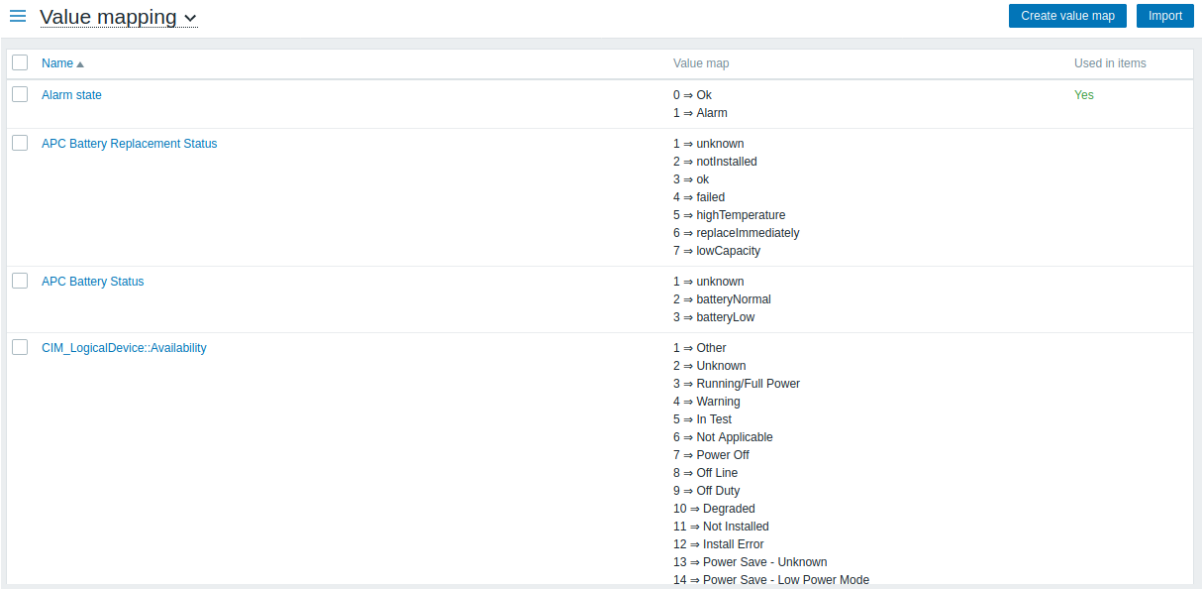
This section allows to define system-wide macros as macro-value pairs. Adding a description is also supported.



See User macros section for more details.

8 Value mapping

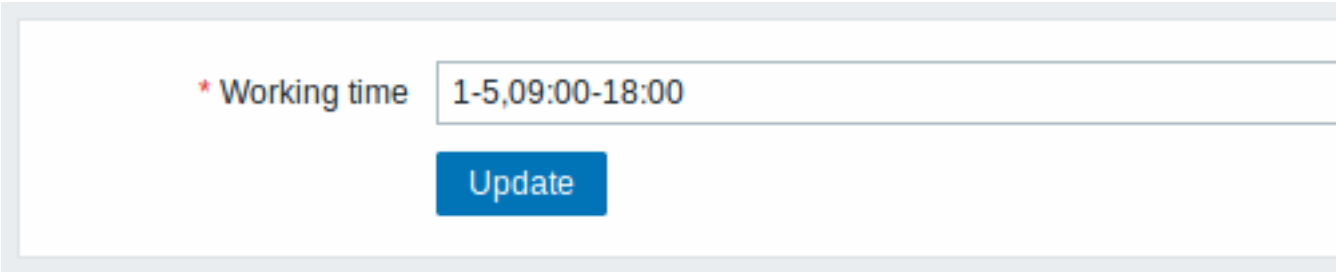
This section allows to manage value maps that are useful for human-readable representation of incoming data in Zabbix frontend.



See Value mapping section for more details.

9 Working time

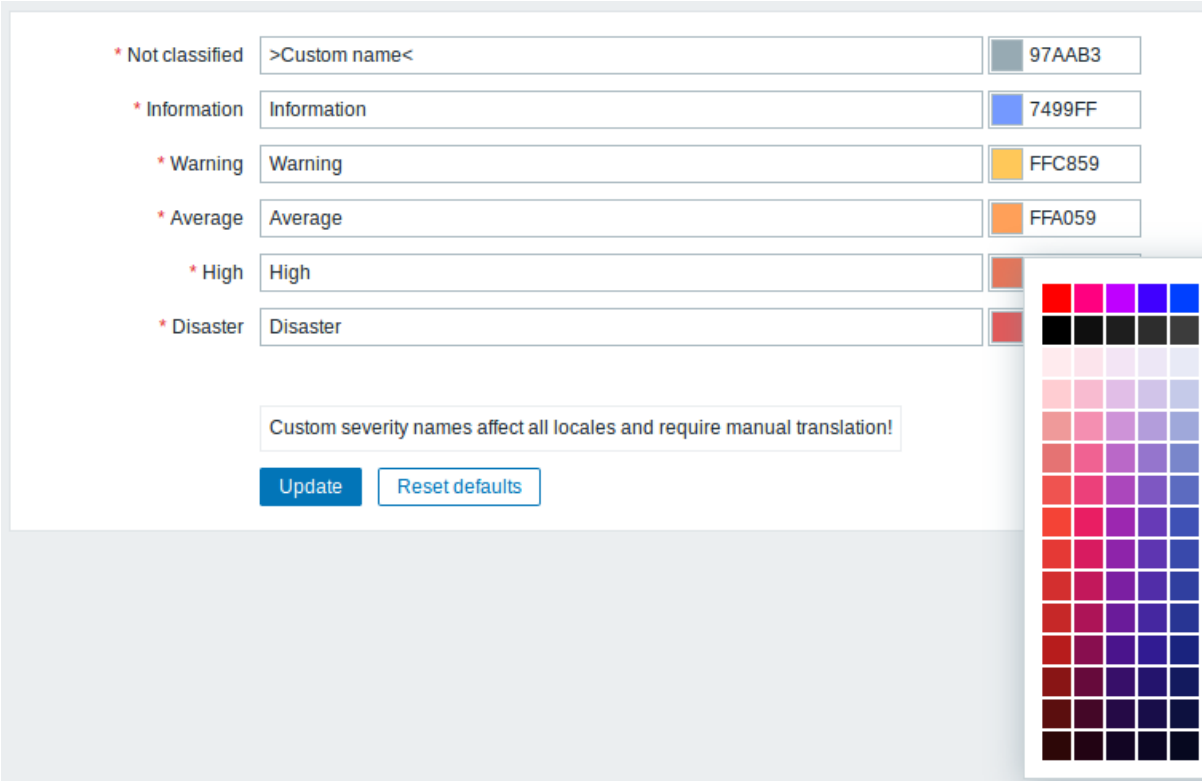
Working time is system-wide parameter, which defines working time. Working time is displayed as a white background in graphs, while non-working time is displayed in gray.



See Time period specification page for description of the time format. User macros are supported (since Zabbix 3.4.0).

10 Trigger severities

This section allows to customize trigger severity names and colors.

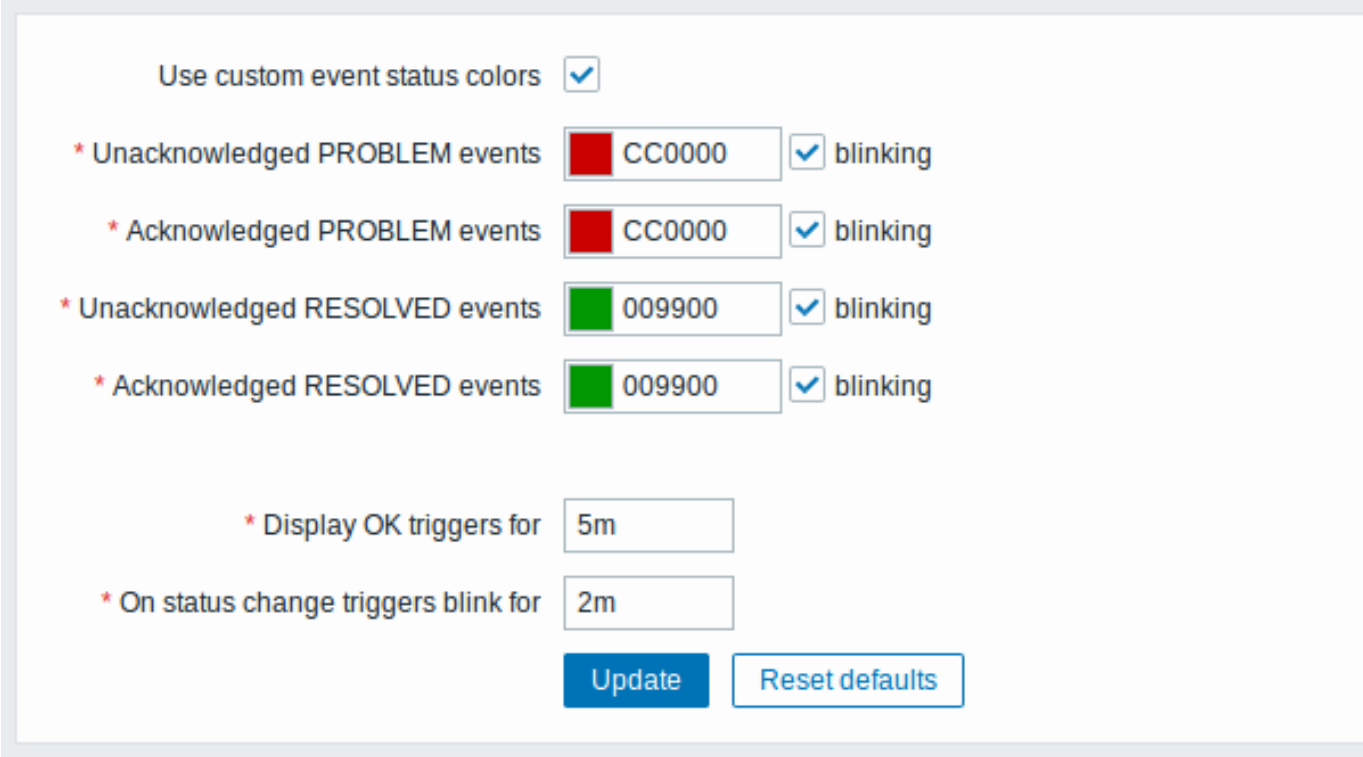


You can enter new names and color codes or click on the color to select another from the provided palette.

See Customizing trigger severities page for more information.

11 Trigger displaying options

This section allows to customize how trigger status is displayed in the frontend.



The *Use custom event status colors* option allows to turn on the customization of colors for acknowledged/unacknowledged problems.

Also the time period for displaying OK triggers and for blinking upon trigger status change can be customized. The maximum value is 86400 seconds (24 hours). Time suffixes are supported in the period fields, e.g. 5m, 2h, 1d.

12 Modules

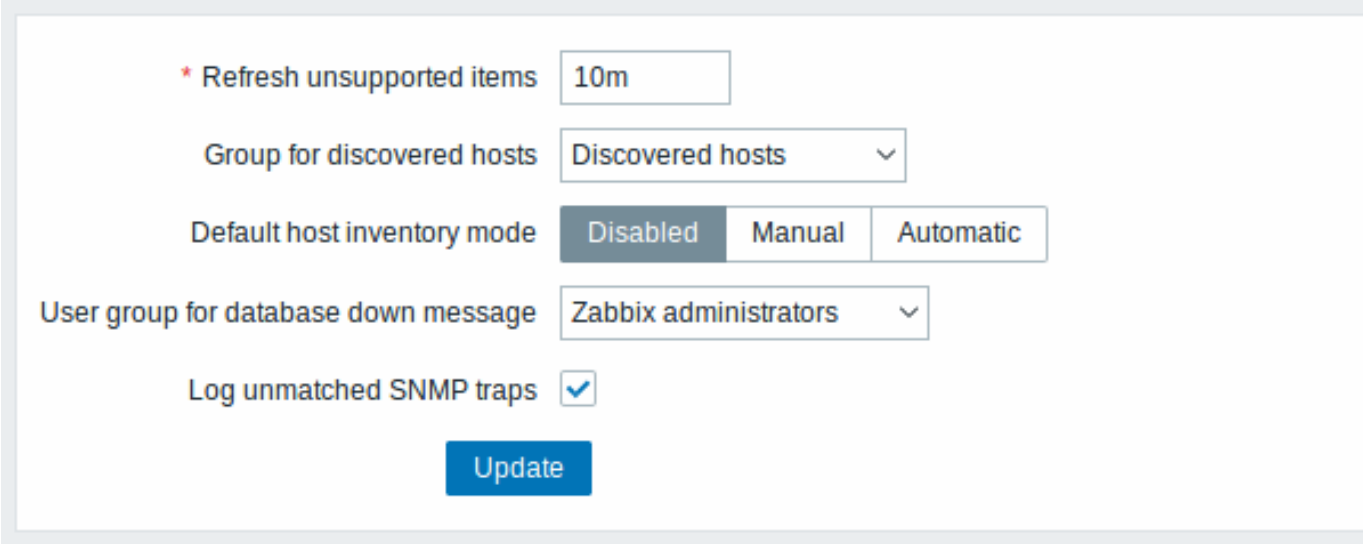
This section allows to administer custom frontend modules.

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.

13 Other parameters

This section allows to configure several other frontend parameters.



Parameter	Description
Refresh unsupported items	Some items may become unsupported due to errors in user parameters or because of an item not being supported by agent. Zabbix can be configured to periodically make unsupported items active. Zabbix server will activate unsupported item every N period set here (1 day maximum). If set to 0, the automatic activation will be disabled. Note that the first attempt to reactivate the unsupported item may occur earlier than the value configured here. Time suffixes are supported, e.g. 60s, 5m, 2h, 1d. The configured value also applies to how often Zabbix proxies reactivate unsupported items. This value is limited for active checks as it only postpones the item inclusion into the list of active checks, while afterwards the agent will poll that item according to the previously scheduled update interval. This value is not taken into account when items become unsupported because of a failed preprocessing step or data normalization.
Group for discovered hosts	Hosts discovered by network discovery and agent autoregistration will be automatically placed in the host group, selected here.
Default host inventory mode	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.

Parameter	Description
User group for database down message	User group for sending alarm message or 'None'. Zabbix server depends on the availability of 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: `[MySQL|PostgreSQL|Oracle] database <DB Name> [on <DB Host>:<DB Port>] is not available: <error message depending on the type of DBMS (database)>` <DB Host> is not added to the message if it is defined as an empty value and <DB Port> is not added if it is the default value ("0"). The alert manager (a special Zabbix server process) tries to establish a new connection to the database every 10 seconds. If the database is still down the alert manager repeats sending alerts, but not more often than every 15 minutes.
Log unmatched SNMP traps	Log SNMP trap if no corresponding SNMP interfaces have been found.

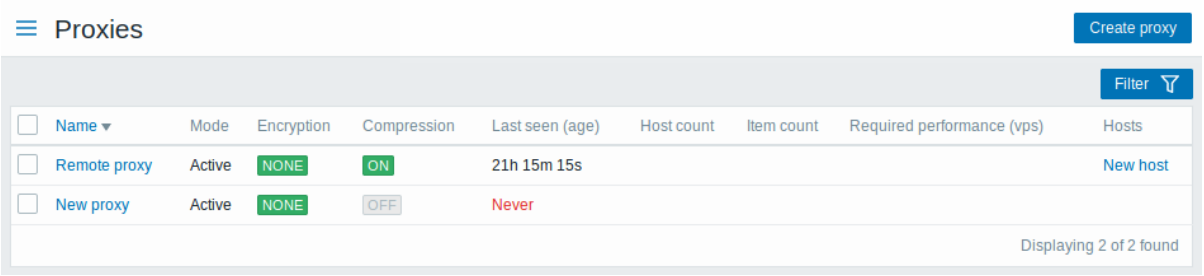
2 Proxies

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:

Column	Description
Name	Name of the proxy. Clicking on the proxy name opens the proxy configuration form.
Mode	Proxy mode is displayed - *Active* or *Passive*.
Encryption	Encryption status for connections from the proxy is displayed: **None** - no encryption **PSK** - using pre-shared key **Cert** - using certificate
Last seen (age)	The time when the proxy was last seen by the server is displayed.
Host count	The number of enabled hosts assigned to the proxy is displayed.
Item count	The number of enabled items on enabled hosts assigned to the proxy is displayed.
Required performance (vps)	Required proxy performance is displayed (the number of values that need to be collected per second).
Hosts	All hosts monitored by the proxy are listed. Clicking on the host name opens the host configuration form.

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:

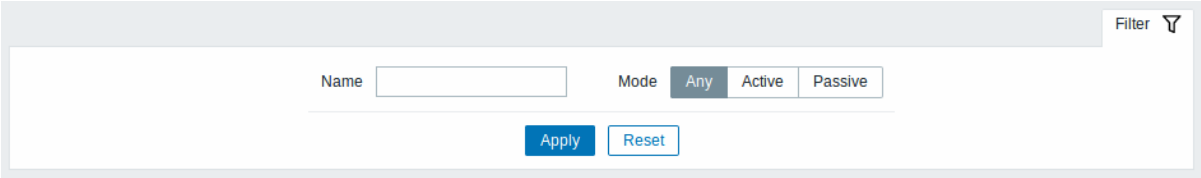
- *Enable hosts* - change the status of hosts monitored by the proxy to *Monitored*
- *Disable hosts* - change the status of hosts monitored by the proxy to *Not monitored*
- *Delete* - delete the proxies

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.

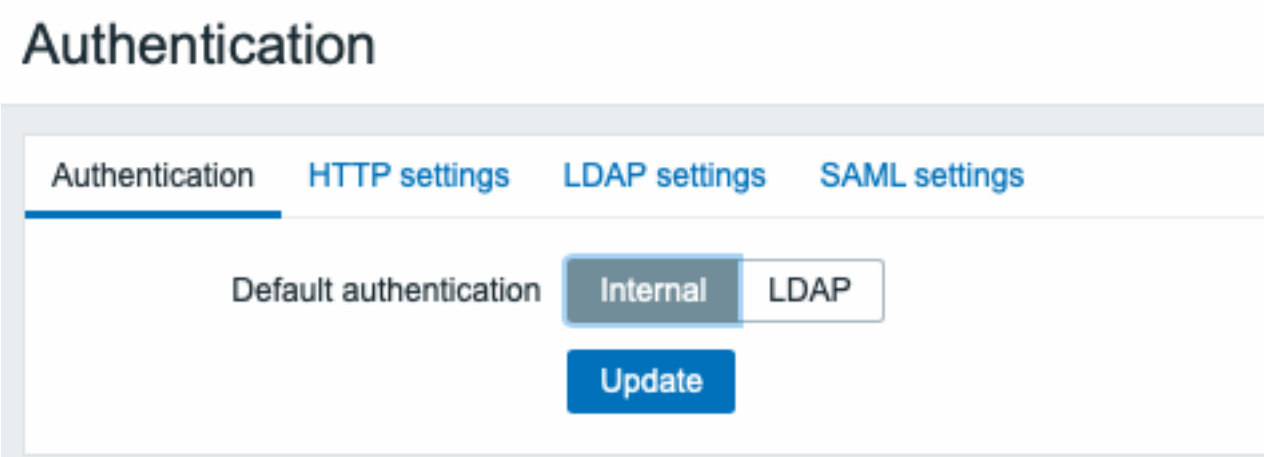
The *Filter* link is available above the list of proxies. If you click on it, a filter becomes available where you can filter proxies by name and mode.



3 Authentication

Overview

In *Administration* → *Authentication* the global user authentication method to Zabbix can be specified. The available methods are internal, HTTP, LDAP and SAML 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.

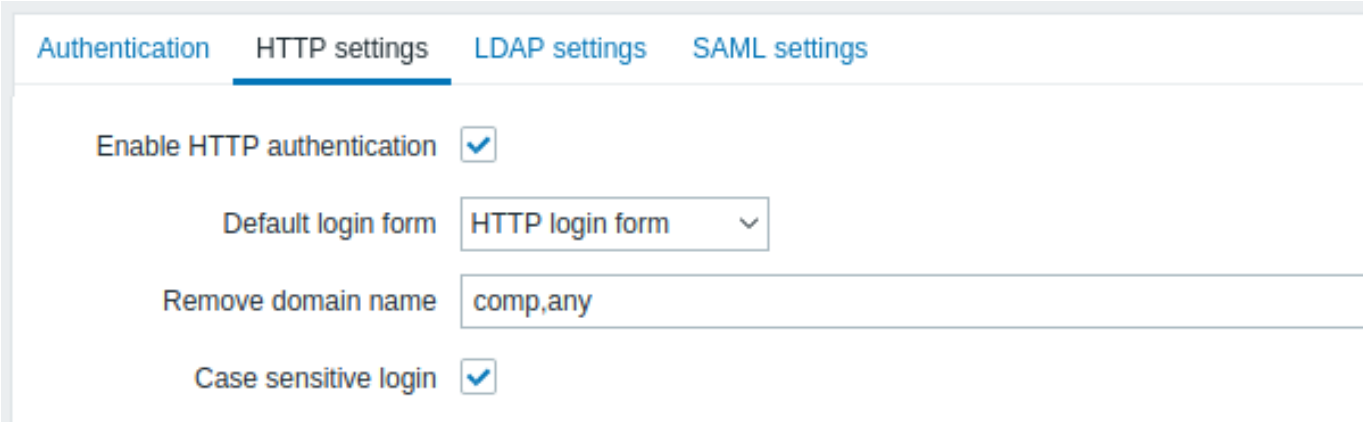
Note that the authentication method can be fine-tuned on the user group level. Even if LDAP authentication is set globally, some user groups can still be authenticated by Zabbix. These groups must have frontend access set to Internal. Vice versa, if internal authentication is used globally, LDAP authentication 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.

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.

Attention:
Be careful! Make sure that web server authentication is configured and works properly before switching it on.



Configuration parameters:

Parameter	Description
Enable HTTP authentication	Mark the checkbox to enable HTTP authentication.
Default login form	Specify whether to direct non-authenticated users to: **Zabbix login form** - standard Zabbix login page. **HTTP login form** - HTTP login page. It is recommended to enable web-server based authentication for the `index_http.php` page only. If *Default login form* is set to 'HTTP login page' the user will be logged in automatically if web server authentication module will set valid user login in the `$_SERVER` variable. Supported `$_SERVER` keys are `PHP_AUTH_USER`, `REMOTE_USER`, `AUTH_USER`.
Remove domain name	A comma-delimited list of domain names that should be removed from the username. E.g. `comp,any` - if username is 'Admin@any', 'comp\Admin', user will be logged in as 'Admin'; if username is 'notacompany\Admin', login will be denied.
Case sensitive login	Unmark the checkbox to disable case-sensitive login for usernames (enabled by default). Disabling case-sensitive login allows, for example, to log in as "admin" even if the Zabbix user is "Admin" or "ADMIN". Please note that if case-sensitive login is disabled and there are multiple Zabbix users with similar usernames (e.g., Admin and admin), the login for those users will always be denied with the following error message: "Authentication failed: supplied credentials are not unique."

Note:
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!

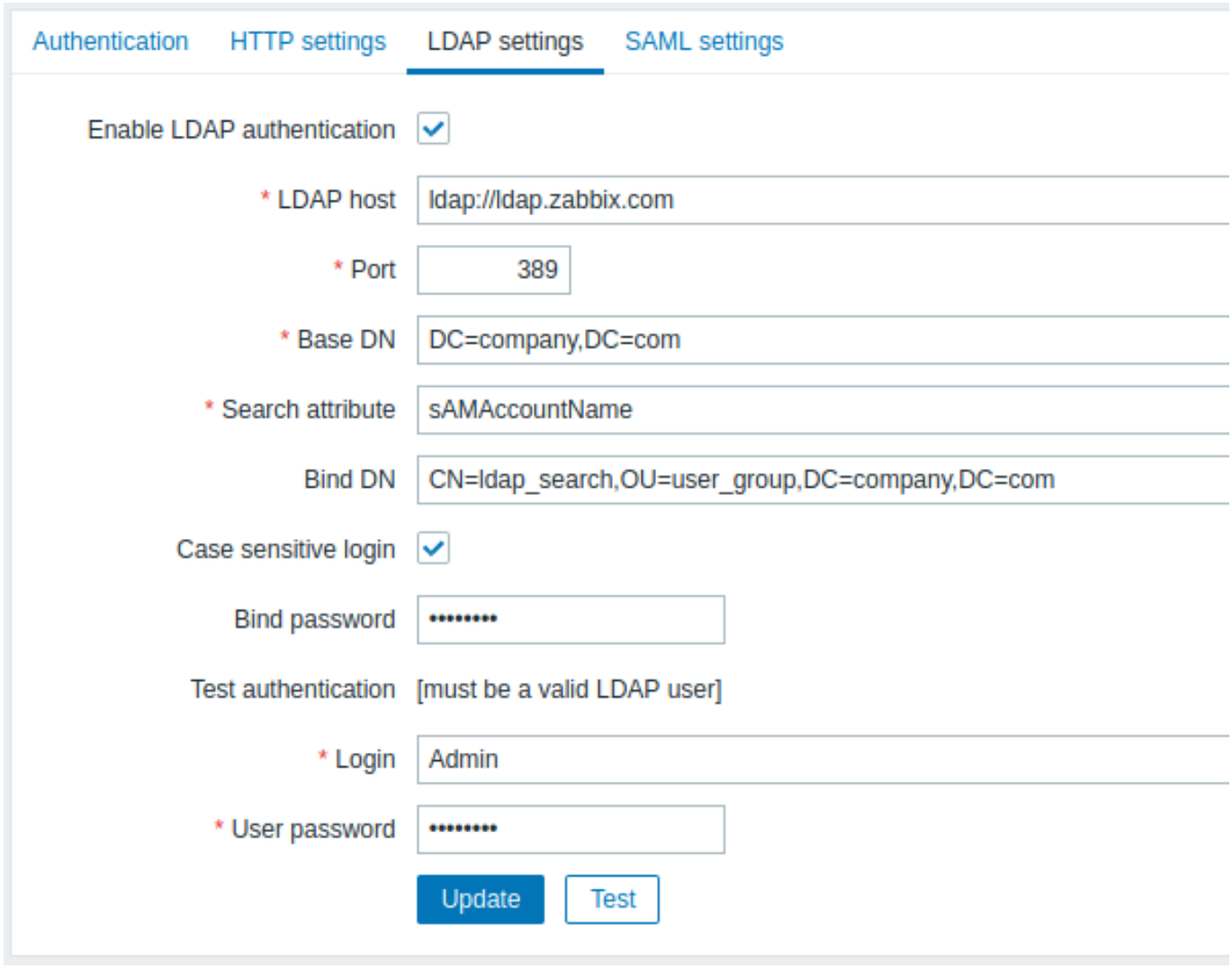
Note:
For internal users who are unable to log in using HTTP credentials (with HTTP login form set as default) leading to the 401 error, you may want to add a `ErrorDocument 401 /index.php?form=default` line to basic authentication directives, which will redirect to the regular Zabbix login form.

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.

While 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.

Zabbix LDAP authentication works at least with Microsoft Active Directory and OpenLDAP.



Configuration parameters:

Parameter	Description
Enable LDAP authentication	Mark the checkbox to enable LDAP authentication.
LDAP host	Name of LDAP server. For example: ldap://ldap.zabbix.com For secure LDAP server use *ldaps* protocol. ldaps://ldap.zabbix.com With OpenLDAP 2.x.x and later, a full LDAP URI of the form ldap://hostname:port or ldaps://hostname:port may be used.
Port	Port of LDAP server. Default is 389. For secure LDAP connection port number is normally 636. Not used when using full LDAP URIs.
Base DN	Base path to search accounts: ou=Users,ou=system (for OpenLDAP), DC=company,DC=com (for Microsoft Active Directory)
Search attribute	LDAP account attribute used for search: uid (for OpenLDAP), sAMAccountName (for Microsoft Active Directory)
Bind DN	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.

Parameter	Description
Case sensitive login	Unmark the checkbox to disable case-sensitive login for usernames (enabled by default). Disabling case-sensitive login allows, for example, to log in as "admin" even if the Zabbix user is "Admin" or "ADMIN". Please note that if case-sensitive login is disabled and there are multiple Zabbix users with similar usernames (e.g., Admin and admin), the login for those users will always be denied with the following error message: "Authentication failed: supplied credentials are not unique."
Bind password	LDAP password of the account for binding and searching over the LDAP server.
Test authentication	Header of a section for testing
Login	Name of a test user (which is currently logged in the Zabbix frontend). This user name must exist in the LDAP server. Zabbix will not activate LDAP authentication if it is unable to authenticate the test user.
User password	LDAP password of the test user.

Warning:
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.

Note:
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 (*alias*) with the username attribute returned by SAML.

Note:
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`

`<path_to_zabbix_ui>` examples: %% https://example.com/zabbix/ui, http://another.example.com/zabbix, http://<any_public_ip_address> %%

Setting up Zabbix

Attention:
It is required to install php-openssl if you want to use SAML authentication in the frontend.

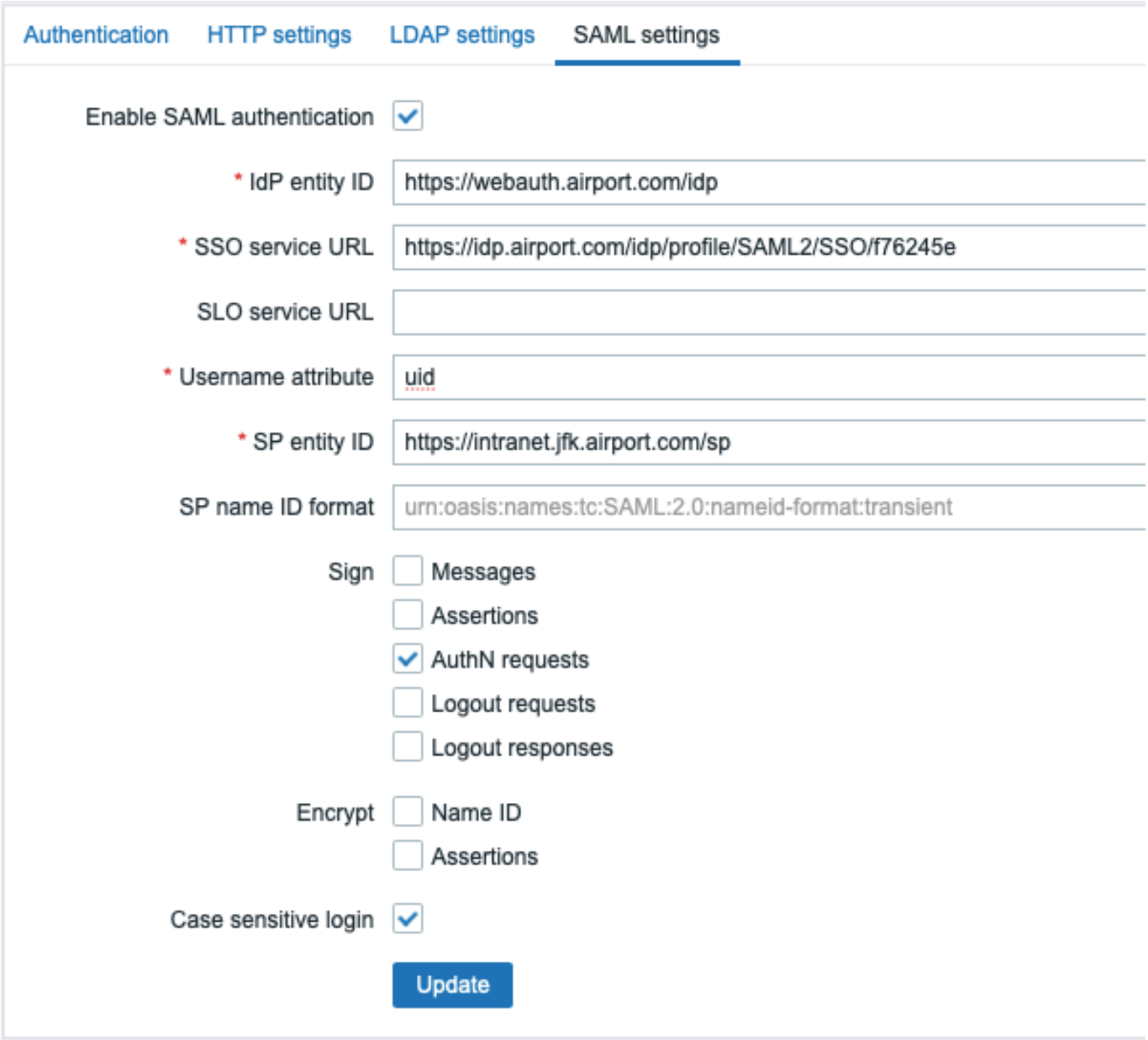
To use SAML authentication Zabbix should be configured in the following way:

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:

Parameter	Description
Enable SAML authentication	Mark the checkbox to enable SAML authentication.
IDP entity ID	The unique identifier of SAML identity provider.
SSO service URL	The URL users will be redirected to when logging in.
SLO Service URL	The URL users will be redirected to when logging out. If left empty, the SLO service will not be used.
// Username attribute//	SAML attribute to be used as a username when logging into Zabbix. List of supported values is determined by the identity provider. Examples: uid userprincipalname samaccountname username userusername urn:oid:0.9.2342.19200300.100.1.1 urn:oid:1.3.6.1.4.1.5923.1.1.1.13 urn:oid:0.9.2342.19200300.100.1.44

Parameter	Description
SP entity ID	The unique identifier of SAML service provider.
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	Unmark the checkbox to disable case-sensitive login for usernames (enabled by default). Disabling case-sensitive login allows, for example, to log in as "admin" even if the Zabbix user is "Admin" or "ADMIN". Please note that if case-sensitive login is disabled and there are multiple Zabbix users with similar usernames (e.g., Admin and admin), the login for those users will always be denied with the following error message: "Authentication failed: supplied credentials are not unique."

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']

Note:
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:

```
$SSO['SETTINGS']=['strict' => false, 'baseurl' => "https://zabbix.example.com/zabbix/", 'use_proxy_headers
```

Configuration example:

```
$SSO['SETTINGS'] = [
    'security' => [
        'signatureAlgorithm' => 'http://www.w3.org/2001/04/xmldsig-more#rsa-sha384'
        'digestAlgorithm' => 'http://www.w3.org/2001/04/xmldsig-more#sha384',
        // ...
    ],
    // ...
];
```

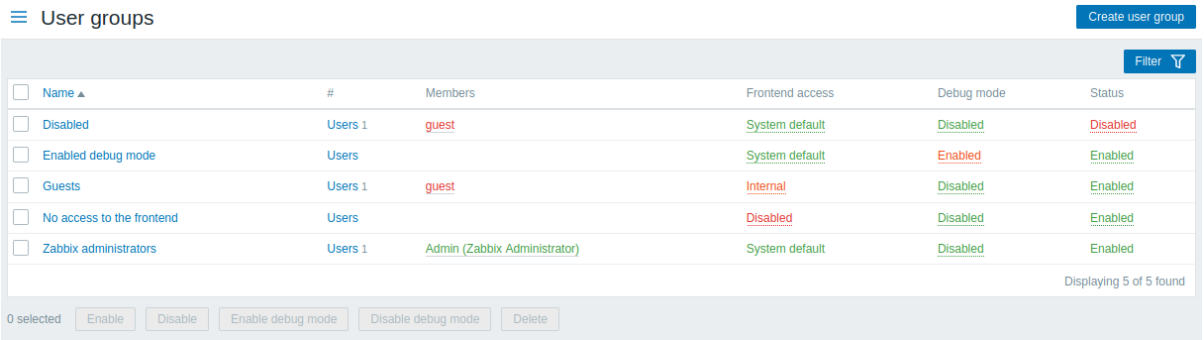
4 User groups

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:

Column	Description
Name	Name of the user group. Clicking on the user group name opens the user group configuration form.
#	The number of users in the group. Clicking on *Users* will display the respective users filtered out in the user list.
Members	Aliases of individual users in the user group (with name and surname in parentheses). Clicking on the alias will open the user configuration form. Users from disabled groups are displayed in red.

Column	Description
Frontend access	Frontend access level is displayed: **System default** - users are authenticated by Zabbix, LDAP or HTTP (depending on the authentication method set globally); **Internal** - users are authenticated by Zabbix; ignored if HTTP authentication is the global default; **LDAP** - users are authenticated by LDAP; ignored if HTTP authentication is the global default; **Disabled** - access to Zabbix frontend is forbidden for this group. By clicking on the current level, you can change it.
Debug mode	Debug mode status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.
Status	User group status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.

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:

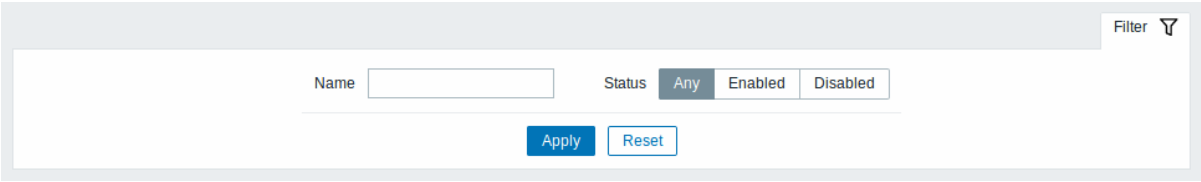
- *Enable* - change the user group status to *Enabled*
- *Disable* - change the user group status to *Disabled*
- *Enable debug mode* - enable debug mode for the user groups
- *Disable debug mode* - disable debug mode for the user groups
- *Delete* - delete the user groups

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.



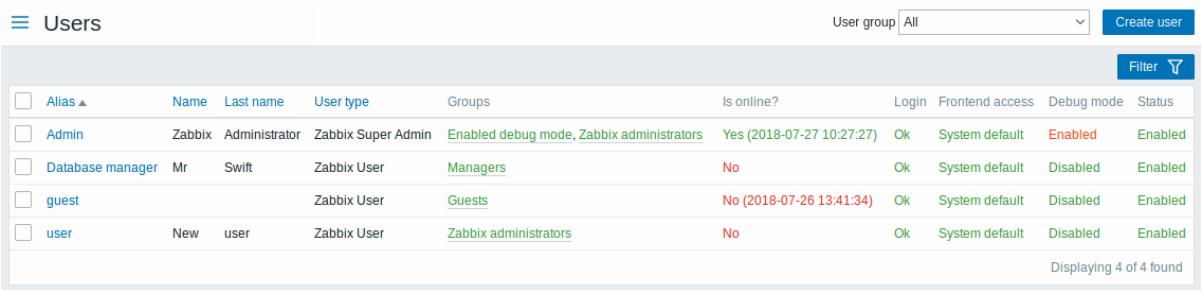
5 Users

Overview

In the *Administration* → *Users* section users of the system are maintained.

Users

A listing of existing users with their details is displayed.



From the dropdown to the right in the *Users* bar you can choose whether to display all users or those belonging to one particular group.

Displayed data:

Column	Description
Alias	Alias of the user, used for logging into Zabbix. Clicking on the alias opens the user configuration form.
Name	First name of the user.
Last name	Second name of the user.
User type	User type is displayed - *Zabbix Super Admin*, *Zabbix Admin* or *Zabbix User*.
Groups	Groups that the user is member of are listed. Clicking on the user group name opens the user group configuration form. Disabled groups are displayed in red.
Is online?	The on-line status of the user is displayed - *Yes* or *No*. The time of last user activity is displayed in parentheses.
Login	The login status of the user is displayed - *Ok* or *Blocked*. A user can become temporarily blocked upon more than five unsuccessful login attempts. By clicking on *Blocked* you can unblock the user.
Frontend access	Frontend access level is displayed - *System default*, *Internal*, *LDAP*, or *Disabled*, depending on the one set for the whole user group.
Debug mode	Debug mode status is displayed - *Enabled* or *Disabled*, depending on the one set for the whole user group.
Status	User status is displayed - *Enabled* or *Disabled*, depending on the one set for the whole user group.

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:

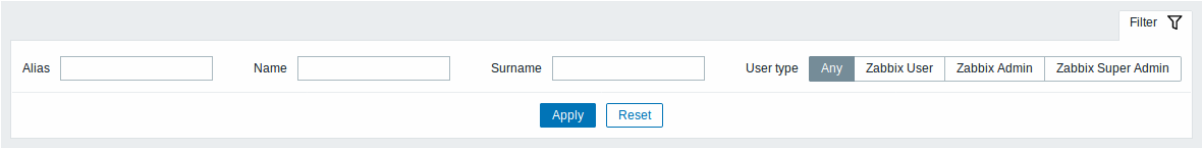
- *Unblock* - re-enable system access to blocked users
- *Delete* - delete the users

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 alias, name, surname and user type.



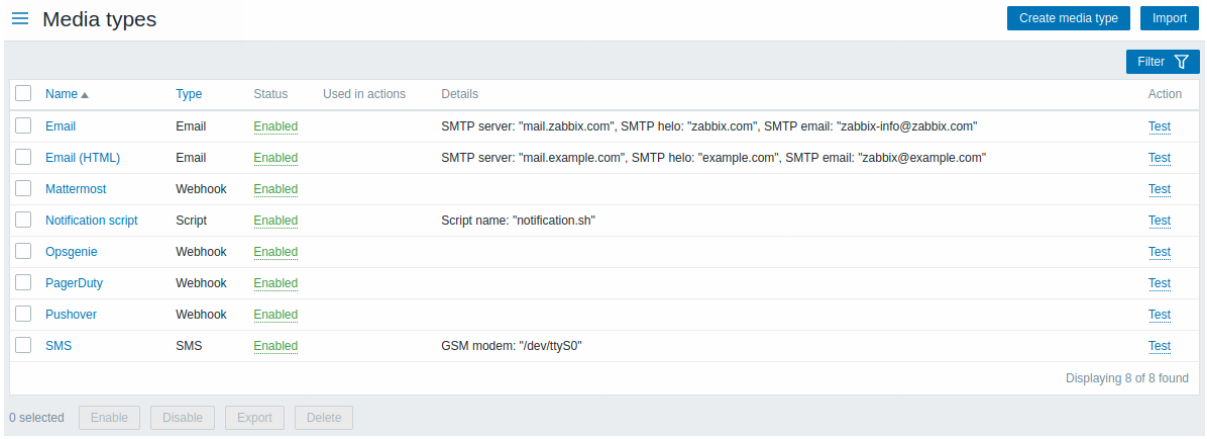
6 Media types

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
Name	Name of the media type. Clicking on the name opens the media type configuration form.
Type	Type of the media (e-mail, SMS, etc) is displayed.
Status	Media type status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.
Used in actions	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.
Details	Detailed information of the media type is displayed.
Actions	The following action is available: **Test** - 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.

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:

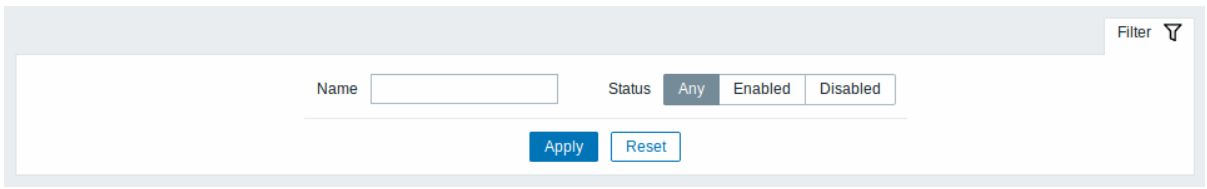
- *Enable* - change the media type status to *Enabled*
- *Disable* - change the media type status to *Disabled*
- *Export* - export the media types to an XML file
- *Delete* - delete the media types

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.



7 Scripts

Overview

In the *Administration* → *Scripts* section user-defined global scripts can be configured and maintained. Each script can be applied to different hosts as needed. See also Command execution.

The scripts, depending on the set user permissions, are available for execution by clicking on the host in various frontend locations (*Dashboard*, *Problems*, *Latest data*, *Maps*) and can also be run as an action operation. The scripts are executed on the Zabbix server (proxy) or agent.

Both on Zabbix agent and Zabbix proxy remote scripts are disabled by default. They can be enabled by:

- adding the `AllowKey=system.run[*]` parameter in agent configuration;
- setting the EnableRemoteCommands parameter to '1' in proxy configuration (also in agent configuration before Zabbix 5.0.2)

A listing of existing scripts with their details is displayed.

Displayed data:

Column	Description
Name	Name of the script. Clicking on the script name opens the script configuration form.
Type	Script type is displayed - *Script* or *IPMI* command.
Execute on	It is displayed whether the script will be executed on Zabbix server or agent.
Commands	All commands to be executed within the script are displayed.
User group	The user group that the script is available to is displayed (or *All* for all user groups).
Host group	The host group that the script is available for is displayed (or *All* for all host groups).
Host access	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.

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:

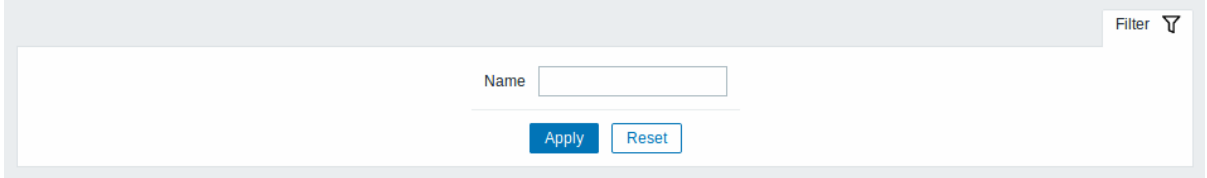
- *Delete* - delete the scripts

To use this option, mark the checkboxes before the respective scripts and click on *Delete*.

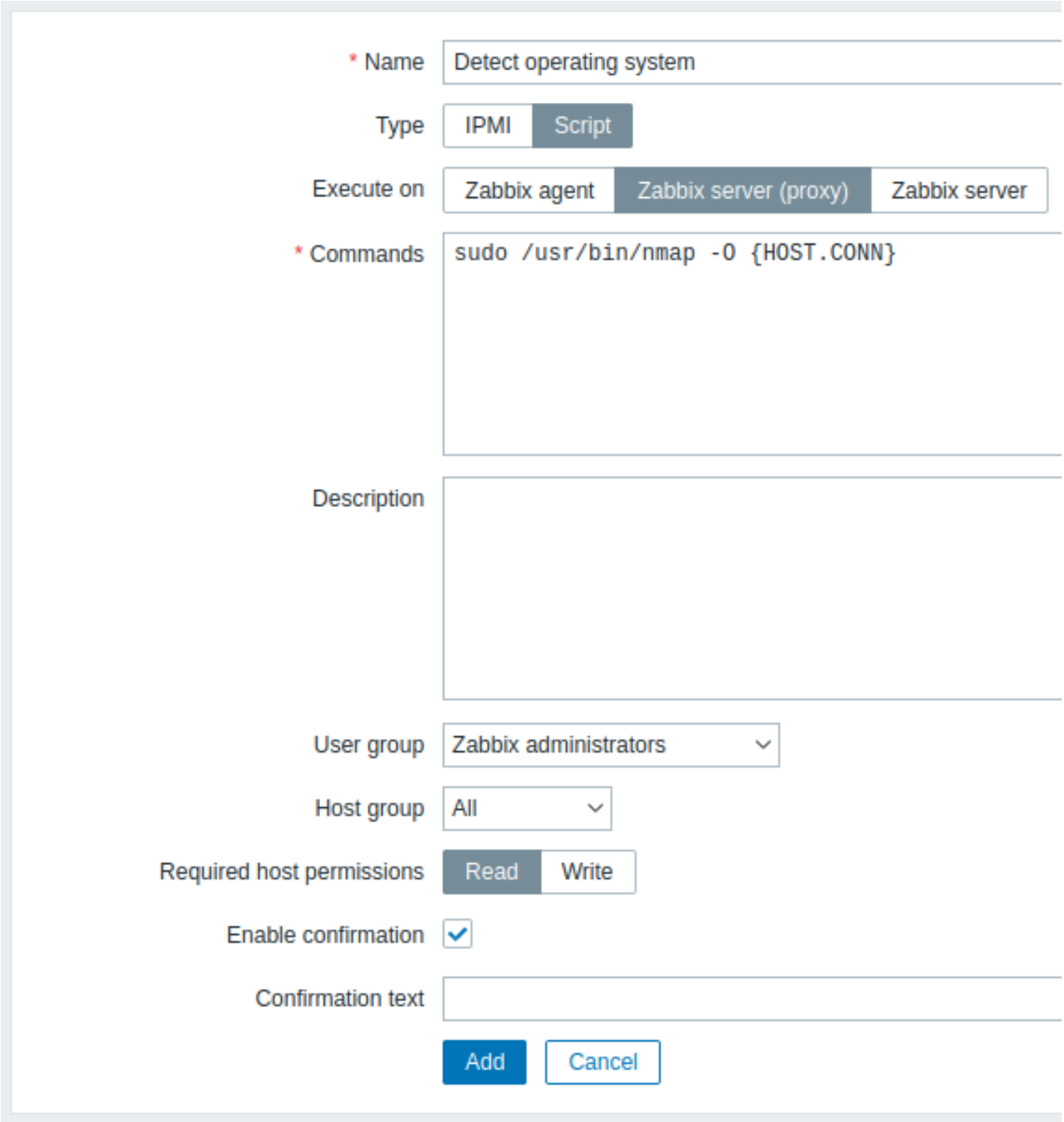
Using filter

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.



Configuring a global script



Script attributes:

Parameter	Description
Name	Unique name of the script. Since Zabbix 2.2 the name can be prefixed with the desired path, for example, `Default/`, putting the script into the respective directory. When accessing scripts through the menu in monitoring sections, they will be organized according to the given directories. A script cannot have the same name as an existing directory (and vice versa). A script name must be unique within its directory. Unescaped script names are validated for uniqueness, i.e. "Ping" and "\Ping" cannot be added in the same folder. A single backslash escapes any symbol directly after it. For example, characters '/' and '\' can be escaped by backslash, i.e. \/ or \\.
Type	Click the respective button to select script type - IPMI command or Script.

Parameter	Description
Execute on	Click the respective button to execute the script on: **Zabbix agent** - the script will be executed by Zabbix agent (if the system.run item is allowed) on the host **Zabbix server (proxy)** - 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 **Zabbix server** - the script will be executed by Zabbix server only
Commands	Enter full path to the commands to be executed within the script. The following macros are supported: host-related - {HOST.CONN}, {HOST.IP}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}; user-related - {USER.ALIAS}, {USER.FULLNAME}, {USER.NAME}, {USER.SURNAME}; custom user macros. *Notes:* If a macro may resolve to a value with spaces (for example, host name), don't forget to quote as needed. User-related macros are supported since Zabbix 5.0.2 to allow passing information about the user that launched the script. If the script was executed automatically under an action operation, these macros will not be resolved.
Description	Enter a description for the script.
User group	Select the user group that the script will be available to (or *All* for all user groups).
Host group	Select the host group that the script will be available for (or *All* for all host groups).
Required host permissions	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.
Enable confirmation	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.
Confirmation text	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. All macros supported for script commands are also supported for the confirmation text. *Note:* the macros will not be expanded when testing the confirmation message.

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:

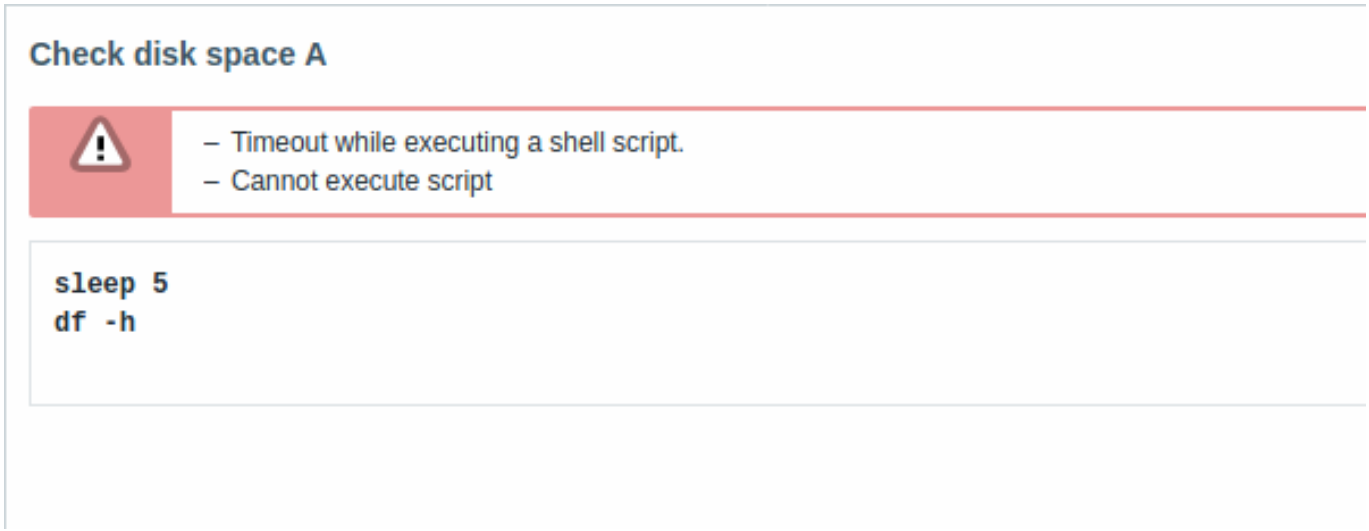
```
uname -v
/tmp/non_existing_script.sh
echo "This script was started by {USER.ALIAS}"
```

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:



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).

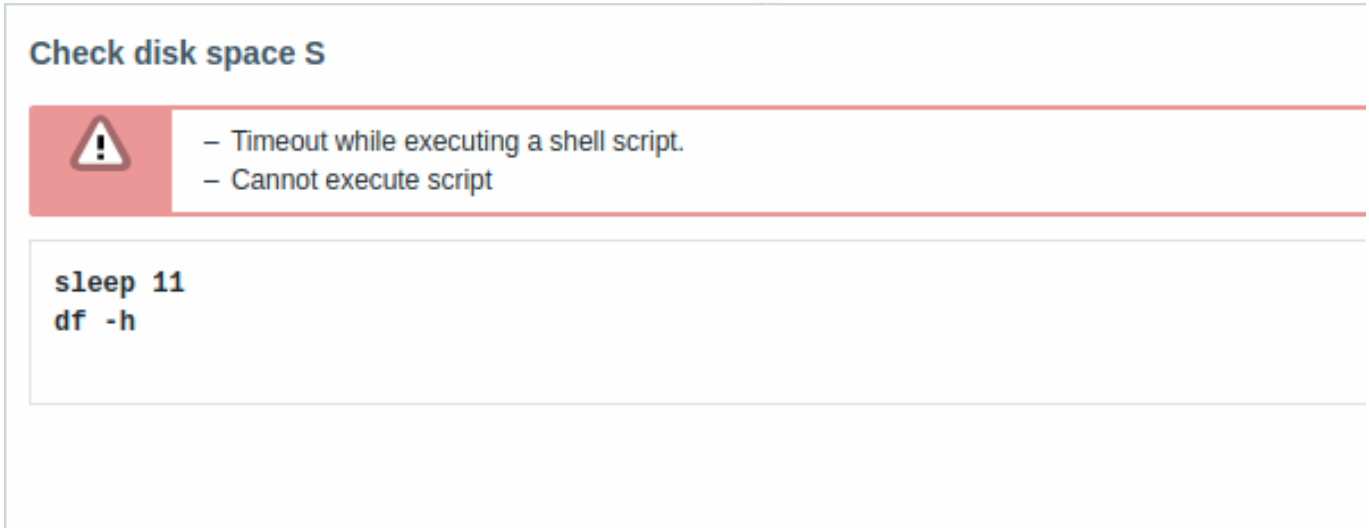
In case still the Timeout parameter is changed in Zabbix agent configuration following error message appears:

```
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:



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).

8 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, some items may get delayed and the information is displayed in this section. For more details, see the Queue section.

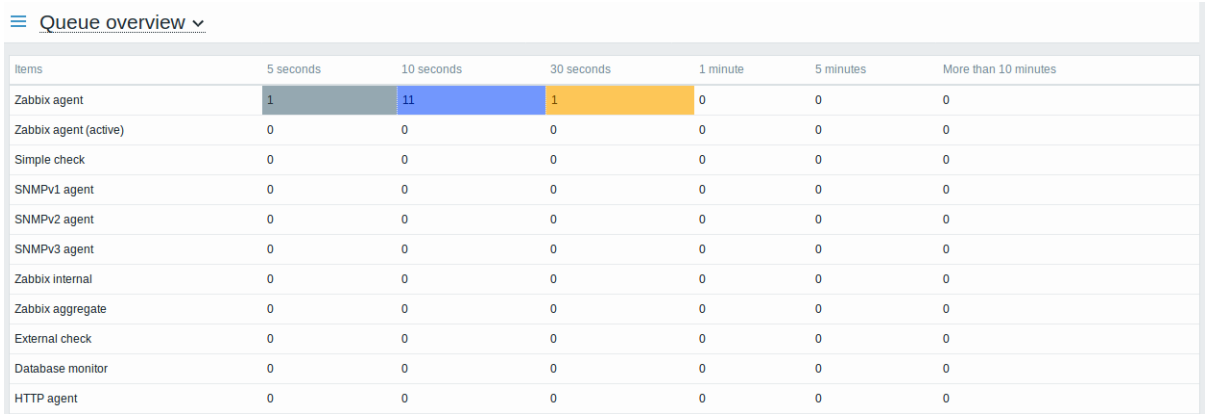
Note:
The queue is available only if Zabbix server is running. Items are not counted in the queue if the item interface becomes unavailable due to connection problems or agent not working properly.

From the title dropdown you can select:

- queue overview by item type
- queue overview by proxy
- list of delayed items

Overview by item type

In this screen it is easy to locate if the problem is related to one or several item types.

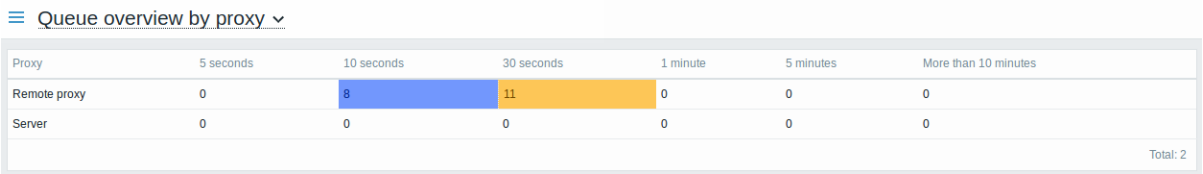


Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	1	11	1	0	0	0
Zabbix agent (active)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	0	0
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	0	0	0	0	0	0
Zabbix aggregate	0	0	0	0	0	0
External check	0	0	0	0	0	0
Database monitor	0	0	0	0	0	0
HTTP agent	0	0	0	0	0	0

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

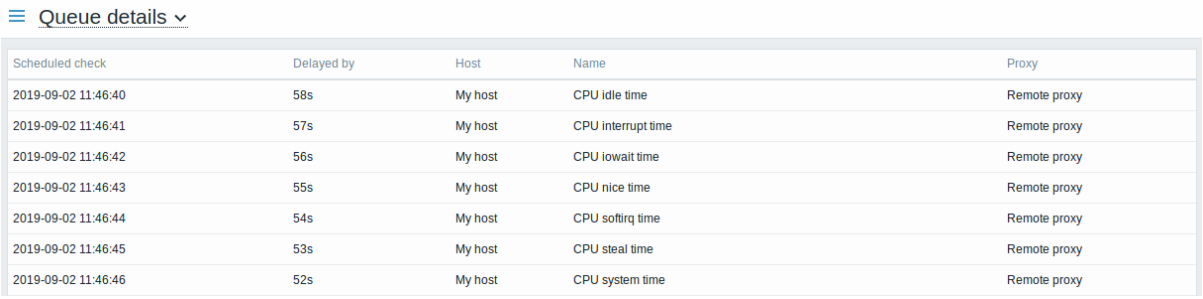
In this screen it is easy to locate if the problem is related to one of the proxies or the server.



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:

Column	Description
Scheduled check	The time when the check was due is displayed.
Delayed by	The length of the delay is displayed.
Host	Host of the item is displayed.
Name	Name of the waiting item is displayed.
Proxy	The proxy name is displayed, if the host is monitored by proxy.

Possible error messages

You may encounter a situation when no data is displayed and the following error message appears:

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 profile

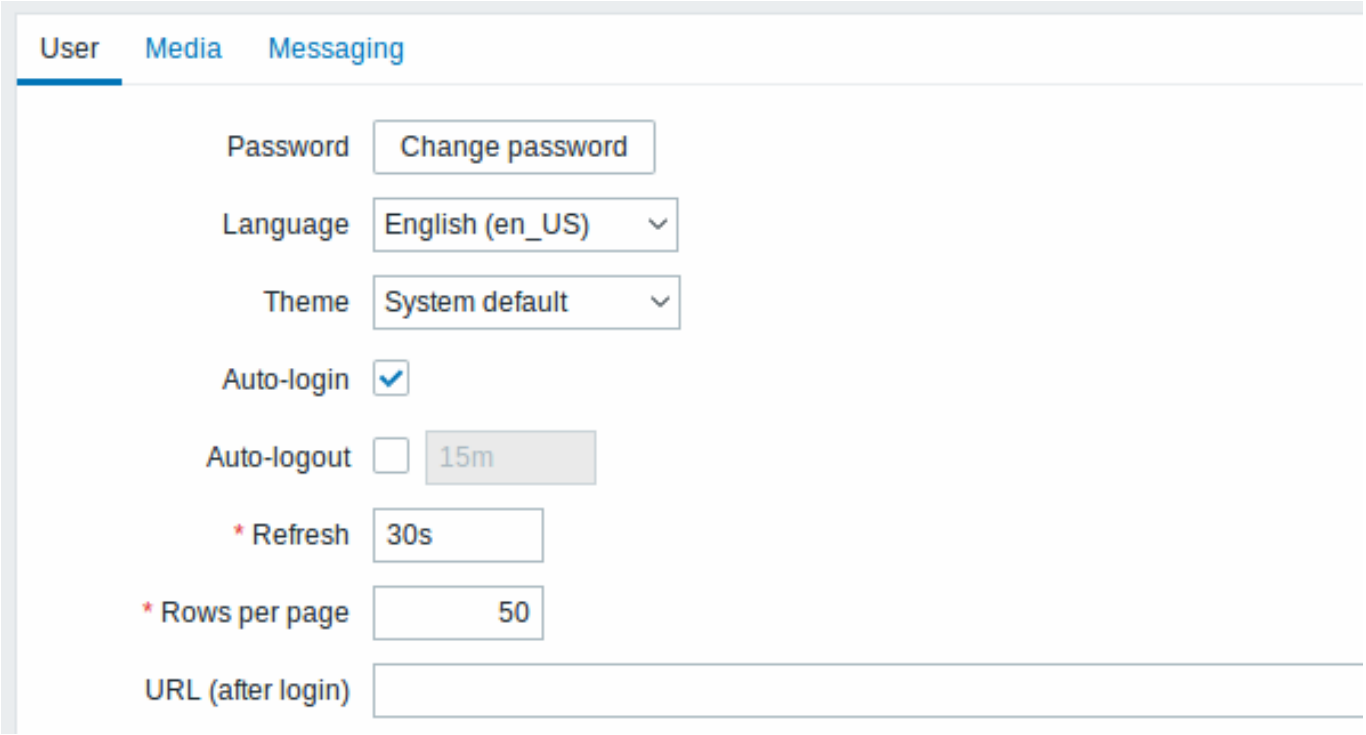
Overview

In the user profile you can customize some Zabbix frontend features, such as the interface language, color theme, number of rows displayed in the lists etc. The changes made here will apply for the user only.

To access the user profile configuration form, click on the user settings near the bottom of the Zabbix menu.

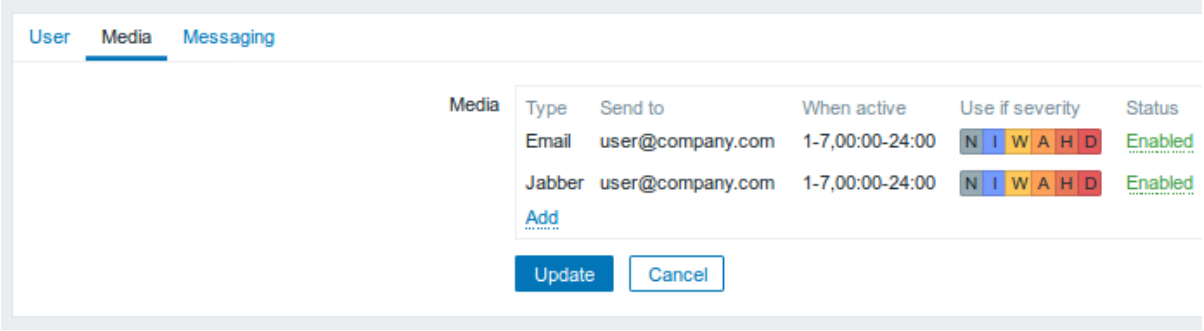
Configuration

The **User** tab allows you to set various user preferences.



Parameter	Description
Password	Click on the link to display two fields for entering a new password.
Language	Select the interface language of your choice. For more information, see Installation of additional frontend languages.
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. Note that this option will not work: * When Monitoring menu pages perform background information refreshes. In case pages refreshing data in a specific time interval (dashboards, graphs, screens, latest data, etc.) are left open session lifetime is extended, respectively disabling auto-logout feature; * If logging in with the *Remember me for 30 days* option checked. Auto-logout can accept 0, meaning that Auto-logout becomes disabled after profile settings update.
Refresh	Set how often the information on the *Monitoring* menu pages will be refreshed (minimum 0 seconds, maximum 1 hour), except for *Dashboard*, which uses its own refresh parameters for every widget. Time suffixes are supported, e.g. 30s, 90s, 1m, 1h.
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.
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.



Note:
Only admin level users (Admin and Super Admin) can change their own media details.

The **Messaging** tab allows you to set global notifications.

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.

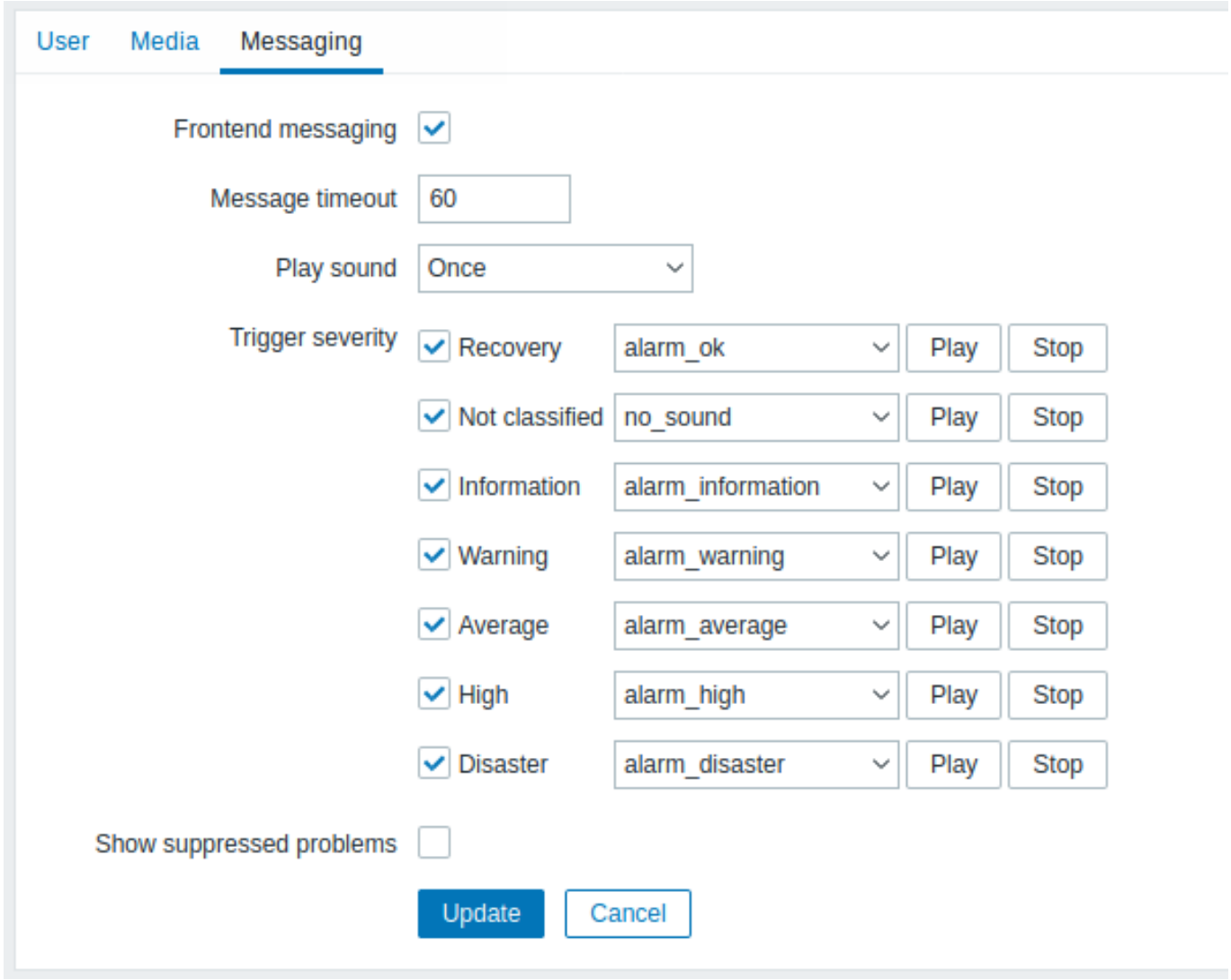
Without global notifications, working in some other location than *Problems* or the *Dashboard* would not show any information regarding issues that are currently happening. Global notifications will display this information regardless of where you are.

Global notifications involve both showing a message and playing a sound.

Attention:
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

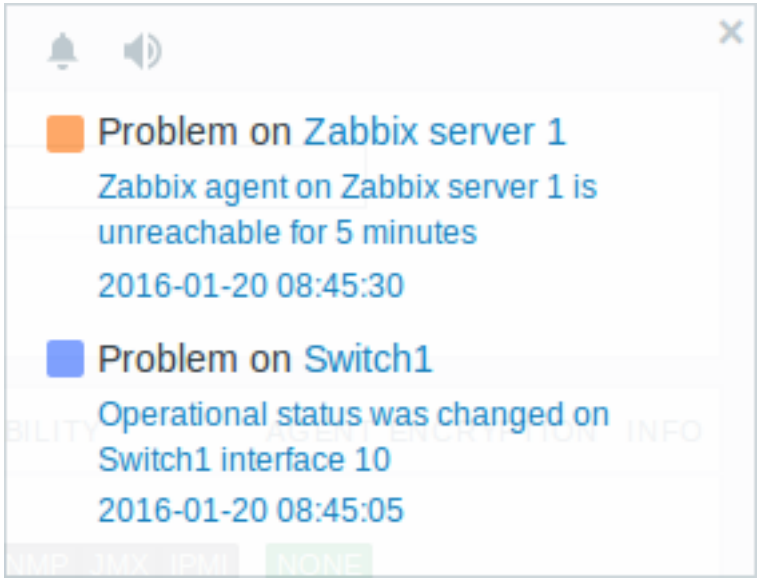
Global notifications can be enabled per user in the *Messaging* tab of profile configuration.



Parameter	Description
Frontend messaging	Mark the checkbox to enable global notifications.
Message timeout	You can set for how long the message will be displayed. By default, messages will stay on screen for 60 seconds. Time suffixes are supported, e.g. 30s, 5m, 2h, 1d.
Play sound	You can set how long the sound will be played. **Once** - sound is played once and fully. **10 seconds** - sound is repeated for 10 seconds. **Message timeout** - sound is repeated while the message is visible.
Trigger severity	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. If no severity is marked then no messages will be displayed at all. 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.
Show suppressed problems	Mark the checkbox to display notifications for problems which would otherwise be suppressed (not shown) because of host maintenance.

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 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 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 in Chrome, Firefox, Microsoft Edge, and Opera browsers on Windows.

Attention:
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 and templates.

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:

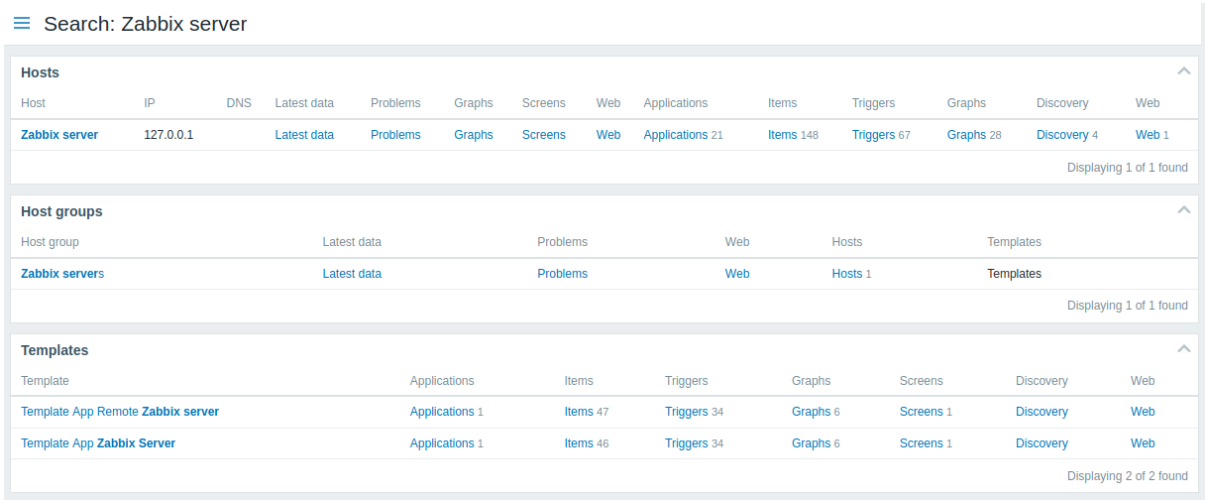
- Host name
- Visible name
- IP address
- DNS name

Host groups can be searched by name. Specifying a parent host group implicitly selects all nested host groups.

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.

Search results

Search results consist of three separate blocks for hosts, host groups and templates.



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
        * Host screens
        * Web scenarios
    - Configuration
        * Applications
        * Items
        * Triggers
        * Graphs
        * Discovery rules
        * Web scenarios

- Host groups
    - Monitoring
        * Latest data
        * Problems
        * Web scenarios
    - Configuration
        * Hosts
        * Templates

- Templates
    - Configuration
        * Applications
        * Items
        * Triggers
        * Graphs
        * Template screens
        * Discovery rules
        * Web scenarios

5 Frontend maintenance mode

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 web server) must be modified to uncomment the following lines:

```
// 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...';
```

Note:
Mostly the `maintenance.inc.php` file is located in `/conf` of Zabbix HTML document directory on the web server. However, the location of the directory may differ depending on the operating system and a web server it uses.
For example, the location for:

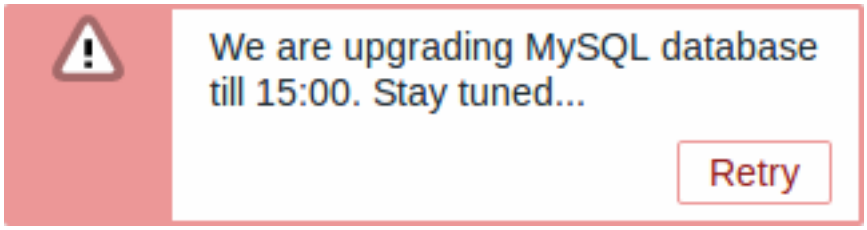
- SUSE and RedHat is `/etc/zabbix/web/maintenance.inc.php`.
- Debian-based systems is `/usr/share/zabbix/conf/`.

See also Copying PHP files.

Parameter	Details
ZBX_DENY_GUI_ACCESS	Enable maintenance mode: 1 – maintenance mode is enabled, disabled otherwise
ZBX_GUI_ACCESS_IP_RANGE	Array of IP addresses, which are allowed to connect to frontend (optional). For example: `array('192.168.1.1', '192.168.1.2')`
ZBX_GUI_ACCESS_MESSAGE	A message you can enter to inform users about the maintenance (optional).

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.



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:

- `sort` - sort column: clock, host, severity, name
- `sortorder` - sort order or results: DESC - descending, ASC - ascending
- `filter_set` - should be 'filter_set=1' to use "filter_*" parameters
- `filter_rst` - should be 'filter_rst=1' to reset filter elements
- `filter_show` - filter option "Show": 1 - recent problems, 2 - all, 3 - in problem state
- `filter_groupids` - filter option "Host groups": array of host groups IDs
- `filter_hostids` - filter option "Hosts": array of host IDs
- `filter_application` - filter option "Application": freeform string
- `filter_triggerids` - filter option "Triggers": array of trigger IDs
- `filter_name` - filter option "Problem": freeform string
- `filter_severity` - filter option "Minimum severity": 0 - not classified, 1 - information, 2 - warning, 3 - average, 4 - high, 5 - disaster
- `filter_age_state` - filter option "Age less than": should be 'filter_age_state=1' to enable 'filter_age'. Is used only when 'filter_show' equals 3.
- `filter_age` - filter option "Age less than": days
- `filter_inventory` - filter option "Host inventory": array of inventory fields: [field], [value]
- `filter_evaltype` - filter option "Tags", tag filtering strategy: 0 - And/Or, 2 - Or
- `filter_tags` - filter option "Tags": array of defined tags: [tag], [operator], [value]
- `filter_show_tags` - filter option "Show tags": 0 - none, 1 - one, 2 - two, 3 - three
- `filter_tag_name_format` - filter option "Tag name": 0 - full name, 1 - shortened, 2 - none
- `filter_tag_priority` - filter option "Tag display priority": comma-separated string of tag display priority
- `filter_show_suppressed` - filter option "Show suppressed problems": should be 'filter_show_suppressed=1' to show
- `filter_unacknowledged` - filter option "Show unacknowledged only": should be 'filter_unacknowledged=1' to show
- `filter_compact_view` - filter option "Compact view": should be 'filter_compact_view=1' to show
- `filter_show_timeline` - filter option "Show timeline": should be 'filter_show_timeline=1' to show

- `filter_details` - filter option "Show details": should be 'filter_details=1' to show
- `filter_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 'filter_compact_view' is set
- `from` - date range start, can be 'relative' (e.g.: now-1m)
- `to` - date range end, can be 'relative' (e.g.: now-1m)

Fullscreen/kiosk mode

Fullscreen and kiosk modes in supported frontend pages can be activated using URL parameters. For example, in dashboards:

- `/zabbix.php?action=dashboard.view&fullscreen=1` - activate fullscreen mode
- `/zabbix.php?action=dashboard.view&kiosk=1` - activate kiosk mode
- `/zabbix.php?action=dashboard.view&fullscreen=0` - activate normal mode

7 Definitions

Overview

While many things in the frontend can be configured using the frontend itself, some customisations 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_LOGIN_ATTEMPTS

Number of unsuccessful login attempts that is allowed to an existing system user before a login block in applied (see ZBX_LOGIN_BLOCK). By default 5 attempts. Once the set number of login attempts is tried unsuccessfully, each additional unsuccessful attempt results in a login block. Used with internal authentication only.

- ZBX_LOGIN_BLOCK

Number of seconds for blocking a user from accessing Zabbix frontend after a number of unsuccessful login attempts (see ZBX_LOGIN_ATTEMPTS). By default 30 seconds. Used with internal authentication only.

- ZBX_PERIOD_DEFAULT

Default graph period, in seconds. One hour by default.

- ZBX_MIN_PERIOD

Minimum graph period, in seconds. One minute by default.

- ZBX_MAX_PERIOD

Maximum graph period, in seconds. Two years by default since 1.6.7, one year before that.

- ZBX_HISTORY_PERIOD

The maximum period to display history data in *Latest data*, *Web*, *Overview* pages and *Data overview* screen element in seconds. By default set to 86400 seconds (24 hours). Unlimited period, if set to 0 seconds. This constant value also affects how far in the past the value is searched when {ITEM.VALUE} macro in trigger name is resolved.

- 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

- SCREEN_REFRESH_RESPONSIVENESS (available since 2.0.4)

Used in screens and defines the number of seconds after which query skipping will be switched off. Otherwise, if a screen element is in update status all queries on update are skipped until a response is received. With this parameter in use, another update query might be sent after N seconds without having to wait for the response to the first one.

Default: 10

- QUEUE_DETAIL_ITEM_COUNT

Defines retrieval limit of the total items queued. Since Zabbix 3.2.4 may be set higher than default value.

Default: 500

- ZBX_SHOW_SQL_ERRORS (available since 3.4.0)

Show SQL errors in the frontend, if 'true'. If changed to 'false' then SQL errors will still be displayed to all users with *Debug mode* enabled. With *Debug mode* disabled, only Zabbix Super Admin users will see SQL errors. Others will see a generic message: "SQL error. Please contact Zabbix administrator."

Default: true

- VALIDATE_URI_SCHEMES (available since 3.4.5)

Validate a URI against the scheme whitelist defined in ZBX_URI_VALID_SCHEMES.

Default: true

- ZBX_URI_VALID_SCHEMES (available since 3.4.2)

A comma-separated list of allowed URI schemes. Affects all places in the frontend where URIs are used, for example, in map element URLs.

Default: http,https,ftp,file,mailto,tel,ssh

- ZBX_SHOW_TECHNICAL_ERRORS (available since 3.4.4)

Show technical errors (PHP/SQL) to non-Zabbix Super admin users and to users that are not part of user groups with debug mode enabled.

Default: false

- ZBX_SESSION_NAME (available since 4.0.0)

String used as the name of the Zabbix frontend session cookie.

Default: zbx_sessionid

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*:

```
public static function getThemes() {
    return array_merge(parent::getThemes(), [
        'custom-theme' => _('Custom theme')
    ]);
}
```

Attention:
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.

Note:
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

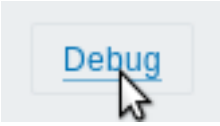
Debug mode may be used to diagnose performance problems with frontend pages.

Configuration

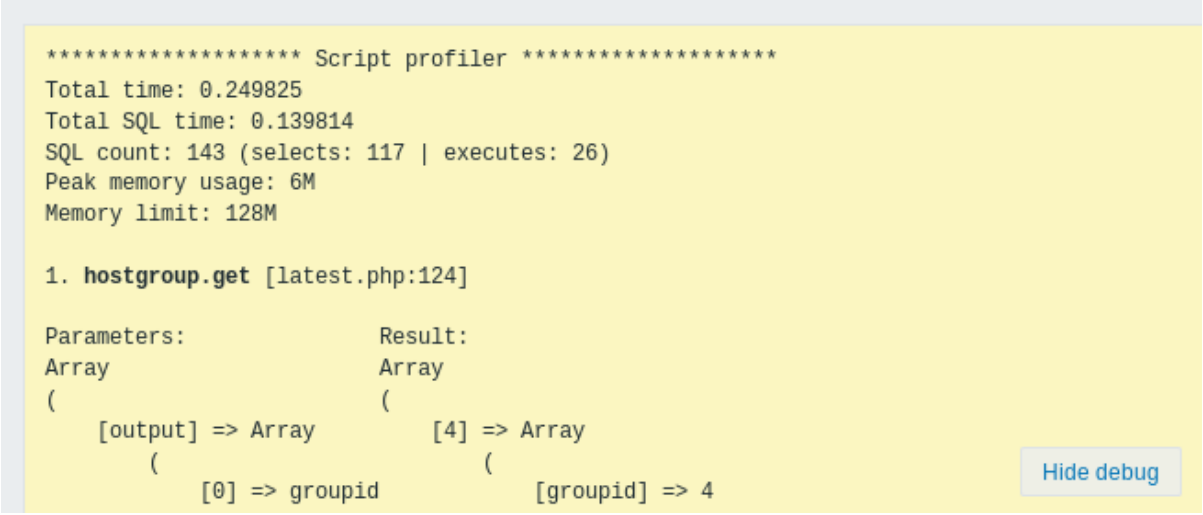
Debug mode can be activated for individual users who belong to a user group:

- 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.

Warning:
Enabled *Debug mode* negatively affects frontend performance.

10 Cookies used by Zabbix

Overview

This page provides a list of cookies used by Zabbix.

Name	Description	Values	Expires/Max-Age	HttpOnly[1]	Secure[2]
PHPSESSID	Unique PHP session ID. The length can be set in *php.ini - session.sid_length*.	Example: kvlp5pu2ru1a2cwfi0g52ms87a	Session (expires when the browsing session ends)	+	+ (only if HTTPS is enabled on a web server)
ZBX_SESSION_NAME (available since 4.0.0). String used as the name of the Zabbix frontend session cookie. Default: zbx_sessionid	Unique session cookie ID - a 32 character string.	Example: 004bc0213e7e8bca87f3c39b9eca5270	Current date and time +1 month (31 days)	+	+ (only if HTTPS is enabled on a web server)
tab	Active tab number; this cookie is only used on pages with multiple tabs (e.g. *Host*, *Trigger* or *Action* configuration page) and is created, when a user navigates from a primary tab to another tab (such as *Tags* or *Dependencies* tab). 0 is used for the primary tab.	Example: 1	Session (expires when the browsing session ends)	-	-
browserwarning_ignore	Whether a warning about using an outdated browser should be ignored.	yes	Session (expires when the browsing session ends)	-	-
messageOk	A message to show as soon as page is reloaded.	Plain text message	Session (expires when the browsing session ends) or as soon as page is reloaded	+	-
messageError	An error message to show as soon as page is reloaded.	Plain text message	Session (expires when the browsing session ends) or as soon as page is reloaded	+	-

Note:
Forcing 'HttpOnly' flag on Zabbix cookies by a webserver directive is not supported.

Footnotes

1 When `HttpOnly` is 'true' the cookie will be made accessible only through the HTTP protocol. This means that the cookie won't be accessible by scripting languages, such as JavaScript. This setting can effectively help to reduce identity theft through XSS attacks (although it is not supported by all browsers).

[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.

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.

For more information about the protocol and JSON, see the JSON-RPC 2.0 specification and the JSON format homepage.

For more information about integrating Zabbix functionality into your Python applications, see the zabbix_utils Python library for Zabbix API.

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".

> **Note:**
> 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://example.com/zabbix*, the HTTP request to call the `apiinfo.version` method may look like this:

```
POST http://example.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`.

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 Zabbix 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"
        ]
    },
    "id": 2,
    "auth": "0424bd59b807674191e7d77572075f33"
}
```

Attention:
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:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "host": "Zabbix server",
            "interfaces": [
                {
                    "interfaceid": "1",
                    "ip": "127.0.0.1"
                }
            ]
        }
    ],
```

```
    "id": 2
}
```

Note:
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:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24759"
        ]
    },
    "id": 3
}
```

Note:
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:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.create",
    "params": [
        {
            "description": "Processor load is too high on {HOST.NAME}",
            "expression": "{Linux server:system.cpu.load[percpu,avg1].last()}>5",
        },
        {
            "description": "Too many processes on {HOST.NAME}",
            "expression": "{Linux server:proc.num[].avg(5m)}>300",
        }
    ],
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 4
}
```

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
}
```

Note:
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": [
        {
            "triggerid": "13938",
            "status": 0
        },
        {
            "triggerid": "13939",
            "status": 0
        }
    ],
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 6
}
```

A successful response will contain the IDs of the updated triggers:

```
{
    "jsonrpc": "2.0",
    "result": {
```

```
        "triggerids": [
            "13938",
            "13939"
        ]
    },
    "id": 6
}
```

Note:
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.

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "error": {
        "code": -32602,
        "message": "Invalid params.",
        "data": "No groups for host \"Linux server\"."
    },
    "id": 7
}
```

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.

Note:
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.

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

Retrieve detailed service layer availability information about any service.

Service SLA calculation

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 | Maintenance API

Items and applications

Define items to monitor. Create or remove applications and assign items to them.

Item API | Application 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

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.

Dashboard API

Screens

Edit global and template-level screens or each screen item individually.

Screen API | Screen item API | Template screen API | Template screen item 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. Configure media types and the ways users will receive alerts.

User API | User group API | Media type API | Audit log API

General

Change certain global configuration options.

Autoregistration API | Icon map API | Image API | User macro API | Value map API

Proxies

Manage the proxies used in your distributed monitoring setup.

Proxy 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.

Property	Type	Description
actionid	string	*(readonly)* ID of the action.
esc_period (required)	string	Default operation step duration. Must be at least 60 seconds. Accepts seconds, time unit with suffix and user macro. Note that escalations are supported only for trigger and internal actions. In trigger actions, escalations are not supported in problem recovery and update operations.
eventsource (required)	integer	*(constant)* Type of events that the action will handle. Refer to the event "source" property for a list of supported event types.
name (required)	string	Name of the action.
status	integer	Whether the action is enabled or disabled. Possible values: 0 - *(default)* enabled; 1 - disabled.

Property	Type	Description
pause_suppressed	integer	Whether to pause escalation during maintenance periods or not. Possible values: 0 - Don't pause escalation; 1 - *(default)* Pause escalation. Note that this parameter is valid for trigger actions only.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Action operation

The action operation object defines an operation that will be performed when an action is executed. It has the following properties.

Property	Type	Description
operationid	string	*(readonly)* ID of the action operation.
operationtype (required)	integer	Type of operation. Possible values: 0 - send message; 1 - remote command; 2 - add host; 3 - remove host; 4 - add to host group; 5 - remove from host group; 6 - link to template; 7 - unlink from template; 8 - enable host; 9 - disable host; 10 - set host inventory mode. Note that only types '0' and '1' are supported for trigger actions, only '0' is supported for internal actions. All types are supported for discovery and autoregistration actions.
actionid	string	ID of the action that the operation belongs to.
esc_period	string	Duration of an escalation step in seconds. Must be greater than 60 seconds. Accepts seconds, time unit with suffix and user macro. If set to 0 or 0s, the default action escalation period will be used. Default: 0s. Note that escalations are supported only for trigger and internal actions. In trigger actions, escalations are not supported in problem recovery and update operations.
esc_step_from	integer	Step to start escalation from. Default: 1. Note that escalations are supported only for trigger and internal actions. In trigger actions, escalations are not supported in problem recovery and update operations.
esc_step_to	integer	Step to end escalation at. Default: 1. Note that escalations are supported only for trigger and internal actions. In trigger actions, escalations are not supported in problem recovery and update operations.

Property	Type	Description
evaltype	integer	Operation condition evaluation method. Possible values: 0 - *(default)* AND / OR; 1 - AND; 2 - OR.
opcommand	object	Object containing the data about the command run by the operation. The operation command object is described in detail below. Required for remote command operations.
opcommand_grp	array	Host groups to run remote commands on. Each object has the following properties: `opcommand_grpid` - *(string, readonly)* ID of the object; `operationid` - *(string)* ID of the operation; `groupid` - *(string)* ID of the host group. Required for remote command operations if `opcommand_hst` is not set.
opcommand_hst	array	Host to run remote commands on. Each object has the following properties: `opcommand_hstid` - *(string, readonly)* ID of the object; `operationid` - *(string)* ID of the operation; `hostid` - *(string)* ID of the host; if set to 0 the command will be run on the current host. Required for remote command operations if `opcommand_grp` is not set.
opconditions	array	Operation conditions used for trigger actions. The operation condition object is described in detail below.
opgroup	array	Host groups to add hosts to. Each object has the following properties: `operationid` - *(string)* ID of the operation; `groupid` - *(string)* ID of the host group. Required for "add to host group" and "remove from host group" operations.
opmessage	object	Object containing the data about the message sent by the operation. The operation message object is described in detail below. Required for message operations.
opmessage_grp	array	User groups to send messages to. Each object has the following properties: `operationid` - *(string)* ID of the operation; `usrgrpid` - *(string)* ID of the user group. Required for message operations if `opmessage_usr` is not set.
opmessage_usr	array	Users to send messages to. Each object has the following properties: `operationid` - *(string)* ID of the operation; `userid` - *(string)* ID of the user. Required for message operations if `opmessage_grp` is not set.

Property	Type	Description
optemplate	array	Templates to link the hosts to. Each object has the following properties: `operationid` - *(string)* ID of the operation; `templateid` - *(string)* ID of the template. Required for ”link to template” and ”unlink from template” operations.
opinventory	object	Inventory mode set host to. Object has the following properties: `operationid` - *(string)* ID of the operation; `inventory_mode` - *(string)* Inventory mode. Required for ”Set host inventory mode” operations.

Action operation command

The operation command object contains data about the command that will be run by the operation.

Property	Type	Description
command	string	Command to run. Required when type IN (0,1,2,3)
type (required)	integer	Type of operation command. Possible values: 0 - custom script; 1 - IPMI; 2 - SSH; 3 - Telnet; 4 - global script.
authtype	integer	Authentication method used for SSH commands. Possible values: 0 - password; 1 - public key. Required for SSH commands.
execute_on	integer	Target on which the custom script operation command will be executed. Possible values: 0 - Zabbix agent; 1 - Zabbix server; 2 - Zabbix server (proxy). Required for custom script commands.
password	string	Password used for SSH commands with password authentication and Telnet commands.
port	string	Port number used for SSH and Telnet commands.
privatekey	string	Name of the private key file used for SSH commands with public key authentication. Required for SSH commands with public key authentication.
publickey	string	Name of the public key file used for SSH commands with public key authentication. Required for SSH commands with public key authentication.
scriptid	string	ID of the script used for global script commands. Required for global script commands.

Property	Type	Description
username	string	User name used for authentication. Required for SSH and Telnet commands.

Action operation message

The operation message object contains data about the message that will be sent by the operation.

Property	Type	Description
default_msg	integer	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.
mediatypeid	string	ID of the media type that will be used to send the message.
message	string	Operation message text.
subject	string	Operation message subject.

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.

Property	Type	Description
opconditionid	string	*(readonly)* ID of the action operation condition
conditiontype (required)	integer	Type of condition. Possible values: 14 - event acknowledged.
value (required)	string	Value to compare with.
operationid	string	*(readonly)* ID of the operation.
operator	integer	Condition operator. Possible values: 0 - *(default)* =.

The following operators and values are supported for each operation condition type.

Condition	Condition name	Supported operators	Expected value
14	Event acknowledged	=	Whether the event is acknowledged. Possible values: 0 - not acknowledged; 1 - acknowledged.

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 actions and internal actions. It has the following properties.

Property	Type	Description
operationid	string	*(readonly)* ID of the action operation.

Property	Type	Description
operationtype (required)	integer	Type of operation. Possible values for trigger actions: 0 - send message; 1 - remote command; 11 - notify all involved. Possible values for internal actions: 0 - send message; 11 - notify all involved.
actionid	string	ID of the action that the recovery operation belongs to.
opcommand	object	Object containing the data about the command run by the recovery operation. The operation command object is described in detail above.
opcommand_grp	array	Required for remote command operations. Host groups to run remote commands on. Each object has the following properties: `opcommand_grpid` - *(string, readonly)* ID of the object; `operationid` - *(string)* ID of the operation; `groupid` - *(string)* ID of the host group.
opcommand_hst	array	Required for remote command operations if `opcommand_hst` is not set. Host to run remote commands on. Each object has the following properties: `opcommand_hstid` - *(string, readonly)* ID of the object; `operationid` - *(string)* ID of the operation; `hostid` - *(string)* ID of the host; if set to 0 the command will be run on the current host.
opmessage	object	Required for remote command operations if `opcommand_grp` is not set. Object containing the data about the message sent by the recovery operation. The operation message object is described in detail above.
opmessage_grp	array	Required for message operations. User groups to send messages to. Each object has the following properties: `operationid` - *(string)* ID of the operation; `usrgrpid` - *(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)* ID of the operation; `userid` - *(string)* ID of the user. Required for message operations if `opmessage_grp` is not set.

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 actions. It has the following

properties.

Property	Type	Description
operationid	string	*(readonly)* ID of the action operation.
operationtype (required)	integer	Type of operation. Possible values for trigger actions: 0 - send message; 1 - remote command; 12 - notify all involved.
opcommand	object	Object containing the data about the command run by the update operation. The operation command object is described in detail above.
opcommand_grp	array	Required for remote command operations. Host groups to run remote commands on. Each object has the following properties: `groupid` - *(string)* ID of the host group.
opcommand_hst	array	Required for remote command operations if `opcommand_hst` is not set. Host to run remote commands on. Each object has the following properties: `hostid` - *(string)* ID of the host; if set to 0 the command will be run on the current host.
opmessage	object	Required for remote command operations if `opcommand_grp` is not set. Object containing the data about the message sent by the update operation. The operation message object is described in detail above.
opmessage_grp	array	User groups to send messages to. Each object has the following properties: `usrgrpid` - *(string)* ID of the user group. Required only for `send message` operations if `opmessage_usr` is not set. Is ignored for `send update message` operations.
opmessage_usr	array	Users to send messages to. Each object has the following properties: `userid` - *(string)* ID of the user. Required only for `send message` operations if `opmessage_grp` is not set. Is ignored for `send update message` operations.

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.

Property	Type	Description
conditions (required)	array	Set of filter conditions to use for filtering results.

Property	Type	Description
evaltype (required)	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
eval_formula	string	*(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.
formula	string	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.

Action filter condition

The action filter condition object defines a specific condition that must be checked before running the action operations.

Property	Type	Description
conditionid	string	*(readonly)* ID of the action condition.

Property	Type	Description
conditiontype (required)	integer	Type of condition. Possible values for trigger actions: 0 - host group; 1 - host; 2 - trigger; 3 - trigger name; 4 - trigger severity; 6 - time period; 13 - host template; 15 - application; 16 - problem is suppressed; 25 - event tag; 26 - event tag value. Possible values for discovery actions: 7 - host IP; 8 - discovered service type; 9 - discovered service port; 10 - discovery status; 11 - uptime or downtime duration; 12 - received value; 18 - discovery rule; 19 - discovery check; 20 - proxy; 21 - discovery object. Possible values for autoregistration actions: 20 - proxy; 22 - host name; 24 - host metadata. Possible values for internal actions: 0 - host group; 1 - host; 13 - host template; 15 - application; 23 - event type.
value (required)	string	Value to compare with.
value2	string	Secondary value to compare with. Required for trigger actions when condition type is *26*.
actionid	string	*(readonly)* ID of the action that the condition belongs to.
formulaid	string	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.

Property	Type	Description
operator	integer	Condition operator. Possible values: 0 - *(default)* equals; 1 - does not equal; 2 - contains; 3 - does not contain; 4 - in; 5 - is greater than or equals; 6 - is less than or equals; 7 - not in; 8 - matches; 9 - does not match; 10 - Yes; 11 - No.

Note:
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.

Condition	Condition name	Supported operators	Expected value
0	Host group	equals, does not equal	Host group ID.
1	Host	equals, does not equal	Host ID.
2	Trigger	equals, does not equal	Trigger ID.
3	Trigger name	contains, does not contain	Trigger name.
4	Trigger severity	equals, does not equal, is greater than or equals, is less than or equals	Trigger severity. Refer to the trigger "severity" property for a list of supported trigger severities.
5	Trigger value	equals	Trigger value. Refer to the trigger "value" property for a list of supported trigger values.
6	Time period	in, not in	Time when the event was triggered as a time period.
7	Host IP	equals, does not equal	One or several IP ranges to check separated by commas. Refer to the network discovery configuration section for more information on supported formats of IP ranges.
8	Discovered service type	equals, does not equal	Type of discovered service. The type of service matches the type of the discovery check used to detect the service. Refer to the discovery check "type" property for a list of supported types.
9	Discovered service port	equals, does not equal	One or several port ranges separated by commas.

Condition	Condition name	Supported operators	Expected value
10	Discovery status	equals	Status of a discovered object. Possible values: 0 - host or service up; 1 - host or service down; 2 - host or service discovered; 3 - host or service lost.
11	Uptime or downtime duration	is greater than or equals, is less than or equals	Time indicating how long has the discovered object been in the current status in seconds.
12	Received values	equals, does not equal, is greater than or equals, is less than or equals, contains, does not contain	Value returned when performing a Zabbix agent, SNMPv1, SNMPv2 or SNMPv3 discovery check.
13	Host template	equals, does not equal	Linked template ID.
15	Application	equals, contains, does not contain	Name of the application.
16	Problem is suppressed	Yes, No	No value required: using the ”Yes” operator means that problem must be suppressed, ”No” - not suppressed.
18	Discovery rule	equals, does not equal	ID of the discovery rule.
19	Discovery check	equals, does not equal	ID of the discovery check.
20	Proxy	equals, does not equal	ID of the proxy.
21	Discovery object	equals	Type of object that triggered the discovery event. Possible values: 1 - discovered host; 2 - discovered service.
22	Host name	contains, does not contain, matches, does not match	Host name. Using a regular expression is supported for operators *matches* and *does not match* in autoregistration conditions.
23	Event type	equals	Specific internal event. Possible values: 0 - item in ”not supported” state; 1 - item in ”normal” state; 2 - LLD rule in ”not supported” state; 3 - LLD rule in ”normal” state; 4 - trigger in ”unknown” state; 5 - trigger in ”normal” state.

Condition	Condition name	Supported operators	Expected value
24	Host metadata	contains, does not contain, matches, does not match	Metadata of the auto-registered host. Using a regular expression is supported for operators *matches* and *does not match*.
25	Tag	equals, does not equal, contains, does not contain	Event tag.
26	Tag value	equals, does not equal, contains, does not contain	Event tag value.

action.create

Description

`object action.create(object/array actions)`

This method allows to create new actions.

Parameters

`(object/array)` Actions to create.

Additionally to the standard action properties, the method accepts the following parameters.

Parameter	Type	Description
filter	object	Action filter object for the action.
operations	array	Action operations to create for the action.
recovery_operations	array	Action recovery operations to create for the action.
acknowledge_operations	array	Action update operations to create for the action.

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 a trigger action that will begin once a trigger (with the word ”memory” in its name) from host ”10084” goes into a PROBLEM state. The action will have 4 configured operations. The first and immediate operation will send a message to all users in user group ”7” via media type ”1”. If the event is not resolved in 30 minutes, the second operation will run script ”3” on all hosts in group ”2”. If the event is resolved, a recovery operation will notify all users who received any messages regarding the problem. If the event is updated, an acknowledge/update operation will notify (with a custom subject and message) all users who received any messages regarding the problem.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Trigger action",
        "eventsource": 0,
        "esc_period": "30m",
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
```

```
                "conditiontype": 1,
                "operator": 0,
                "value": "10084"
            },
            {
                "conditiontype": 3,
                "operator": 2,
                "value": "memory"
            }
        ]
    },
    "operations": [
        {
            "operationtype": 0,
            "esc_step_from": 1,
            "esc_step_to": 1,
            "opmessage_grp": [
                {
                    "usrgrpid": "7"
                }
            ],
            "opmessage": {
                "default_msg": 1,
                "mediatypeid": "1"
            }
        },
        {
            "operationtype": 1,
            "esc_step_from": 2,
            "esc_step_to": 2,
            "opconditions": [
                {
                    "conditiontype": 14,
                    "operator": 0,
                    "value": "0"
                }
            ],
            "opcommand_grp": [
                {
                    "groupid": "2"
                }
            ],
            "opcommand": {
                "type": 4,
                "scriptid": "3"
            }
        }
    ],
    "recovery_operations": [
        {
            "operationtype": "11",
            "opmessage": {
                "default_msg": 1
            }
        }
    ],
    "acknowledge_operations": [
        {
            "operationtype": "12",
            "opmessage": {
                "default_msg": 0,
                "message": "Custom update operation message body",
```

```
                    "subject": "Custom update operation message subject"
                }
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "17"
        ]
    },
    "id": 1
}
```

Create a discovery action

Create a discovery action that will link discovered hosts to template ”10001”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Discovery action",
        "eventsource": 1,
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "conditiontype": 21,
                    "value": "1"
                },
                {
                    "conditiontype": 10,
                    "value": "2"
                }
            ]
        },
        "operations": [
            {
                "optemplate": [
                    {
                        "templateid": "10001"
                    }
                ],
                "operationtype": 6
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
```

```
        "actionids": [
            "18"
        ]
    },
    "id": 1
}
```

Using a custom expression filter

Create a trigger action that uses a custom expression - ”A and (B or C)” - for evaluating action conditions. Once a trigger with a severity higher or equal to ”Warning” from host ”10084” or host ”10106” goes into a PROBLEM state, the action will send a message to all users in user group ”7” via media type ”1”. The formula IDs ”A”, ”B” and ”C” have been chosen arbitrarily.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Trigger action",
        "eventsource": 0,
        "esc_period": "15m",
        "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_step_from": 1,
                "esc_step_to": 1,
                "opmessage_grp": [
                    {
                        "usrgrpid": "7"
                    }
                ],
                "opmessage": {
                    "default_msg": 1,
                    "mediatypeid": "1"
                }
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "18"
        ]
    },
    "id": 1
}
```

Create agent autoregistration rule

Create an autoregistration action that adds a host to host group ”2” when the host name contains ”SRV” or metadata contains ”CentOS”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Register Linux servers",
        "eventsource": "2",
        "filter": {
            "evaltype": "2",
            "conditions": [
                {
                    "conditiontype": "22",
                    "operator": "2",
                    "value": "SRV"
                },
                {
                    "conditiontype": "24",
                    "operator": "2",
                    "value": "CentOS"
                }
            ]
        },
        "operations": [
            {
                "operationtype": "4",
                "opgroup": [
                    {
                        "groupid": "2"
                    }
                ]
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            19
        ]
```

```
    },
    "id": 1
}
```

See also

- Action filter
- Action operation
- Script

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.

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:

```
{
    "jsonrpc": "2.0",
    "method": "action.delete",
    "params": [
        "17",
        "18"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
actionids	string/array	Return only actions with the given IDs.
groupids	string/array	Return only actions that use the given host groups in action conditions.
hostids	string/array	Return only actions that use the given hosts in action conditions.
triggerids	string/array	Return only actions that use the given triggers in action conditions.
mediatypeids	string/array	Return only actions that use the given media types to send messages.
usrgrpids	string/array	Return only actions that are configured to send messages to the given user groups.
userids	string/array	Return only actions that are configured to send messages to the given users.
scriptids	string/array	Return only actions that are configured to run the given scripts.
selectFilter	query	Return a filter property with the action condition filter.
selectOperations	query	Return an operations property with action operations.
selectRecoveryOperations	query	Return a recoveryOperations property with action recovery operations.
selectAcknowledgeOperations	query	Return an acknowledgeOperations property with action update operations.
sortfield	string/array	Sort the result by the given properties. Possible values are: `actionid`, `name` and `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in the reference commentary.
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

Retrieve trigger actions

Retrieve all configured trigger actions together with action conditions and operations.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.get",
    "params": {
        "output": "extend",
        "selectOperations": "extend",
        "selectRecoveryOperations": "extend",
        "selectAcknowledgeOperations": "extend",
        "selectFilter": "extend",
        "filter": {
            "eventsource": 0
```

```
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "actionid": "3",
            "name": "Report problems to Zabbix administrators",
            "eventsource": "0",
            "status": "1",
            "esc_period": "1h",
            "pause_suppressed": "1",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [],
                "eval_formula": ""
            },
            "acknowledgeOperations": [
                {
                    "operationid": "31",
                    "operationtype": "12",
                    "evaltype": "0",
                    "opmessage": {
                        "default_msg": "1",
                        "subject": "",
                        "message": "",
                        "mediatypeid": "0"
                    }
                },
                {
                    "operationid": "32",
                    "operationtype": "0",
                    "evaltype": "0",
                    "opmessage": {
                        "default_msg": "0",
                        "subject": "Updated: {TRIGGER.NAME}",
                        "message": "{USER.FULLNAME} updated problem at {EVENT.UPDATE.DATE} {EVENT.UPDATE.T
                        "mediatypeid": "1"
                    },
                    "opmessage_grp": [
                        {
                            "usrgrpid": "7"
                        }
                    ],
                    "opmessage_usr": []
                },
                {
                    "operationid": "33",
                    "operationtype": "1",
                    "evaltype": "0",
                    "opcommand": {
                        "type": "0",
                        "scriptid": "0",
                        "execute_on": "0",
                        "port": "",
                        "authtype": "0",
```

```
                        "username": "",
                        "password": "",
                        "publickey": "",
                        "privatekey": "",
                        "command": "notify.sh"
                    },
                    "opcommand_hst": [
                        {
                            "hostid": "0"
                        }
                    ],
                    "opcommand_grp": []
                }
            ],
            "operations": [
                {
                    "operationid": "3",
                    "actionid": "3",
                    "operationtype": "0",
                    "esc_period": "0",
                    "esc_step_from": "1",
                    "esc_step_to": "1",
                    "evaltype": "0",
                    "opconditions": [],
                    "opmessage": [
                        {
                            "default_msg": "1",
                            "subject": "",
                            "message": "",
                            "mediatypeid" => "0"
                        }
                    ],
                    "opmessage_grp": [
                        {
                            "usrgrpid": "7"
                        }
                    ]
                }
            ],
            "recoveryOperations": [
                {
                    "operationid": "7",
                    "actionid": "3",
                    "operationtype": "11",
                    "evaltype": "0",
                    "opconditions": [],
                    "opmessage": {
                        "default_msg": "0",
                        "subject": "{TRIGGER.STATUS}: {TRIGGER.NAME}",
                        "message": "Trigger: {TRIGGER.NAME}\r\nTrigger status: {TRIGGER.STATUS}\r\nTrigger
                        "mediatypeid": "0"
                    }
                }
            ]
        }
    ],
    "id": 1
}
```


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",
                    "evaltype": "0",
                    "opconditions": [],
                    "optemplate": [
                        {
                            "templateid": "10001"
                        }
                    ]
                },
                {
                    "operationid": "2",
                    "actionid": "2",
                    "operationtype": "4",
                    "esc_period": "0s",
                    "esc_step_from": "1",
                    "esc_step_to": "1",
                    "evaltype": "0",
                    "opconditions": [],
                    "opgroup": [
                        {
                            "groupid": "2"
                        }
                    ]
                }
            ]
        }
    ],
    "id": 1
}
```


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.

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.

Parameter	Type	Description
filter	object	Action filter object to replace the current filter.
operations	array	Action operations to replace existing operations.
recovery_operations	array	Action recovery operations to replace existing recovery operations.
acknowledge_operations	array	Action update operations to replace existing update operations.

Return values

`(object)` Returns an object containing the IDs of the updated actions under the `actionids` property.

Examples

Disable action

Disable an action, that is, set its status to "1".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.update",
    "params": {
        "actionid": "2",
        "status": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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

Note:
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.

Property	Type	Description
alertid	string	ID of the alert.

Property	Type	Description
actionid	string	ID of the action that generated the alert.
alerttype	integer	Alert type. Possible values: 0 - message; 1 - remote command.
clock	timestamp	Time when the alert was generated.
error	string	Error text if there are problems sending a message or running a command.
esc_step	integer	Action escalation step during which the alert was generated.
eventid	string	ID of the event that triggered the action.
mediatypeid	string	ID of the media type that was used to send the message.
message	text	Message text. Used for message alerts.
retries	integer	Number of times Zabbix tried to send the message.
sendto	string	Address, user name or other identifier of the recipient. Used for message alerts.
status	integer	Status indicating whether the action operation has been executed successfully. 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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
alertids	string/array	Return only alerts with the given IDs.
actionids	string/array	Return only alerts generated by the given actions.
eventids	string/array	Return only alerts generated by the given events.
groupids	string/array	Return only alerts generated by objects from the given host groups.
hostids	string/array	Return only alerts generated by objects from the given hosts.
mediatypeids	string/array	Return only message alerts that used the given media types.
objectids	string/array	Return only alerts generated by the given objects
userids	string/array	Return only message alerts that were sent to the given users.

Parameter	Type	Description
eventobject	integer	Return only alerts generated by events related to objects of the given type. See event "object" for a list of supported object types. Default: 0 - trigger.
eventsource	integer	Return only alerts generated by events of the given type. See event "source" for a list of supported event types. Default: 0 - trigger events.
time_from	timestamp	Return only alerts that have been generated after the given time.
time_till	timestamp	Return only alerts that have been generated before the given time.
selectHosts	query	Return a hosts property with data of hosts that triggered the action operation.
selectMediatypes	query	Return a mediatypes property with an array of the media types that were used for the message alert.
selectUsers	query	Return a users property with an array of the users that the message was addressed to. See `user.get` for restrictions based on user type.
sortfield	string/array	Sort the result by the given properties. Possible values are: `alertid`, `clock`, `eventid`, `mediatypeid`, `sendto` and `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Note:
In Zabbix 5.0.46, *Admin* and *User* type users may retrieve "message" (0) type alert data only about their own user.

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:

```
{
    "jsonrpc": "2.0",
    "method": "alert.get",
    "params": {
        "output": "extend",
        "actionids": "3"
```

```
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "alertid": "1",
            "actionid": "3",
            "eventid": "21243",
            "userid": "1",
            "clock": "1362128008",
            "mediatypeid": "1",
            "sendto": "support@company.com",
            "subject": "PROBLEM: Zabbix agent on Linux server is unreachable for 5 minutes: ",
            "message": "Trigger: Zabbix agent on Linux server is unreachable for 5 minutes: \nTrigger stat
            "status": "0",
            "retries": "3",
            "error": "",
            "esc_step": "1",
            "alerttype": "0",
            "p_eventid": "0",
            "acknowledgeid": "0"
        }
    ],
    "id": 1
}
```

See also

- Host
- Media type
- User

Source

CAlert::get() in *ui/include/classes/api/services/CAlert.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.

Parameters

> **Attention:**
> This method is available to unauthenticated users and must be called without the `auth` parameter in the JSON-RPC request.

`(array)` The method accepts an empty array.

Return values

(`string`) Returns the version of the Zabbix API.

> **Note:**
> 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*.

Application

This class is designed to work with applications.

Object references:

- Application

Available methods:

- application.create - creating new applications
- application.delete - deleting applications
- application.get - retrieving application
- application.massadd - updating application
- application.update - adding items to applications

> Application object

The following objects are directly related to the `application` API.

Application

The application object has the following properties.

Property	Type	Description
applicationid	string	*(readonly)* ID of the application.
hostid (required)	string	ID of the host that the application belongs to. Cannot be updated.
name (required)	string	Name of the application

Property	Type	Description
flags	integer	*(readonly)* Origin of the application. Possible values: 0 - a plain application; 4 - a discovered application.
templateids	array	*(readonly)* IDs of the parent template applications.

application.create

Description

`object application.create(object/array applications)`

This method allows to create new applications.

Parameters

`(object/array)` Applications to create.

The method accepts applications with the standard application properties.

Return values

`(object)` Returns an object containing the IDs of the created applications under the `applicationids` property. The order of the returned IDs matches the order of the passed applications.

Examples

Creating an application

Create an application to store SNMP items.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "application.create",
    "params": {
        "name": "SNMP Items",
        "hostid": "10050"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "applicationids": [
            "356"
        ]
    },
    "id": 1
}
```

Source

CApplication::create() in *ui/include/classes/api/services/CApplication.php*.

application.delete

Description

`object application.delete(array applicationIds)`

This method allows to delete applications.

Parameters

`(array)` IDs of the applications to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted applications under the `applicationids` property.

Examples

Deleting multiple applications

Delete two applications.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "application.delete",
    "params": [
        "356",
        "358"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "applicationids": [
            "356",
            "358"
        ]
    },
    "id": 1
}
```

Source

CApplication::delete() in *ui/include/classes/api/services/CApplication.php*.

application.get

Description

`integer/array application.get(object parameters)`

The method allows to retrieve applications according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
applicationids	string/array	Return only applications with the given IDs.
groupids	string/array	Return only applications that belong to hosts from the given host groups.
hostids	string/array	Return only applications that belong to the given hosts.
inherited	boolean	If set to `true` return only applications inherited from a template.
itemids	string/array	Return only applications that contain the given items.
templated	boolean	If set to `true` return only applications that belong to templates.
templateids	string/array	Return only applications that belong to the given templates.
selectHost	query	Return a host property with the host that the application belongs to.
selectItems	query	Return an items property with the items contained in the application.

Parameter	Type	Description
selectDiscoveryRule	query	Return a discoveryRule property with the LLD rule that created the application.
selectApplicationDiscovery	query	Return an `applicationDiscovery` property with the application discovery object.
sortfield	string/array	Sort the result by the given properties. Possible values are: `applicationid` and `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 applications from a host

Retrieve all applications from a host sorted by name.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "application.get",
    "params": {
        "output": "extend",
        "hostids": "10001",
        "sortfield": "name"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "applicationid": "13",
            "hostid": "10001",
            "name": "CPU",
            "templateids": []
        },
        {
            "applicationid": "5",
            "hostid": "10001",
            "name": "Filesystems",
            "templateids": []
        },
```

```
        {
            "applicationid": "21",
            "hostid": "10001",
            "name": "General",
            "templateids": []
        },
        {
            "applicationid": "15",
            "hostid": "10001",
            "name": "Memory",
            "templateids": []
        },
    ],
    "id": 1
}
```

See also

- Host
- Item

Source

CApplication::get() in *ui/include/classes/api/services/CApplication.php*.

application.massadd

Description

`object application.massadd(object parameters)`

This method allows to simultaneously add multiple items to the given applications.

Parameters

`(object)` Parameters containing the IDs of the applications to update and the items to add to the applications.

The method accepts the following parameters.

Parameter	Type	Description
applications (required)	array/object	Applications to be updated. The applications must have the `applicationid` property defined.
items	array/object	Items to add to the given applications. The items must have the `itemid` property defined.

Return values

`(object)` Returns an object containing the IDs of the updated applications under the `applicationids` property.

Examples

Adding items to multiple applications

Add the given items to two applications.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "application.massadd",
    "params": {
        "applications": [
            {
                "applicationid": "247"
            },
            {
```

```
                "applicationid": "246"
            }
        ],
        "items": [
            {
                "itemid": "22800"
            },
            {
                "itemid": "22801"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "applicationids": [
            "247",
            "246"
        ]
    },
    "id": 1
}
```

See also

- Item

Source

CApplication::massAdd() in *ui/include/classes/api/services/CApplication.php*.

application.update

Description

`object application.update(object/array applications)`

This method allows to update existing applications.

Parameters

`(object/array)` Application properties to be updated.

The `applicationid` property must be defined for each application, 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 applications under the `applicationids` property.

Examples

Changing the name of an application

Change the name of the application to ”Processes and performance”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "application.update",
    "params": {
        "applicationid": "13",
        "name": "Processes and performance"
    },
```

```
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "applicationids": [
            "13"
        ]
    },
    "id": 1
}
```

Source

CApplication::update() in *ui/include/classes/api/services/CApplication.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 `auditlog` API.

Audit log

The audit log object contains information about user actions. It has the following properties.

Property	Type	Description
auditid	string	*(readonly)* ID of audit log entry.
userid	string	Audit log entry author userid.
action	integer	Audit log entry action. Possible values are: 0 - Add; 1 - Update; 2 - Delete; 3 - Login; 4 - Logout; 5 - Enable; 6 - Disable; 7 - Execute.
clock	timestamp	Audit log entry creation timestamp.

Property	Type	Description
resourcetype	integer	Audit log entry resource type. Possible values are: 0 - User; 2 - Configuration of Zabbix; 3 - Media type; 4 - Host; 5 - Action; 6 - Graph; 7 - Graph element; 11 - User group; 12 - Application; 13 - Trigger; 14 - Host group; 15 - Item; 16 - Image; 17 - Value map; 18 - Service; 19 - Map; 20 - Screen; 22 - Web scenario; 23 - Discovery rule; 24 - Slide show; 25 - Script; 26 - Proxy; 27 - Maintenance; 28 - Regular expression; 29 - Macro; 30 - Template; 31 - Trigger prototype; 32 - Icon mapping; 33 - Dashboard; 34 - Event correlation; 35 - Graph prototype; 36 - Item prototype; 37 - Host prototype; 38 - Autoregistration; 39 - Module.
note	string	Audit log entry short description.
ip	string	Audit log entry author IP address.
resourceid	string	Audit log entry resource identifier.
resourcename	string	Audit log entry resource human readable name.

Audit log details

Property	Type	Description
table_name	string	Database table name.
field_name	string	Database table field name.
oldvalue	string	Database table field old value.
newvalue	string	Database table field new value.

auditlog.get

Description

`integer/array auditlog.get(object parameters)`

The method allows to retrieve audit log records according to the given parameters.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
auditids	string/array	Return only audit log with the given IDs.
userids	string/array	Return only audit log that were created by the given users.
time_from	timestamp	Returns only audit log entries that have been created after or at the given time.
time_till	timestamp	Returns only audit log entries that have been created before or at the given time.
selectDetails	query	Returns audit log entries with per field changes as details property. Available only for entries with action *"1 - Update"*, for actions of other types returns empty array.
sortfield	string/array	Sort the result by the given properties. Possible values are: `auditid`, `userid`, `clock`.
filter	object	Return only 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. Additionally supports filtering by details property fields: `table_name`, `field_name`.
search	object	Case insensitive sub-string search in content of fields: `note`, `ip`, `resourcename`, `oldvalue`, `newvalue`.
countOutput	boolean	These parameters being common for all `get` methods are described in the reference commentary.
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
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

Retrieve audit log

Retrieve two latest audit log records.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "auditlog.get",
    "params": {
        "output": "extend",
        "sortfield": "clock",
        "sortorder": "DESC",
        "limit": 2
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "auditid": "189",
            "userid": "1",
            "clock": "1580913141",
            "action": "3",
            "resourcetype": "0",
            "note": "",
            "ip": "127.0.0.1",
            "resourceid": "0",
            "resourcename": ""
        },
        {
            "auditid": "188",
            "userid": "1",
            "clock": "1580903029",
            "action": "3",
            "resourcetype": "0",
            "note": "",
            "ip": "127.0.0.1",
            "resourceid": "0",
            "resourcename": ""
        }
    ],
    "id": 2
}
```


Retrieve audit log records having substring ”test” in oldvalue field.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "auditlog.get",
    "params": {
        "output": ["auditid", "resourcename"],
        "search": {
            "newvalue": "test"
        },
        "selectDetails": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "auditid": "5",
            "resourcename": "Mattermost2",
            "details": [
                {
                    "table_name": "media_type",
                    "field_name": "event_menu_url",
                    "oldvalue": "http://test",
                    "newvalue": "http://test{EVENT.TAGS.__test}"
                }
            ]
        },
```

```
        {
            "auditid": "7",
            "resourcename": "Email",
            "details": [
                {
                    "table_name": "media_type",
                    "field_name": "name",
                    "oldvalue": "Email",
                    "newvalue": "Email test"
                }
            ]
        }
    ],
    "id": 20
}
```


See also

- Audit log object

Source

CAuditLog::get() in *ui/include/classes/api/services/CAuditLog.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.

Property	Type	Description
tls_accept	integer	Type of allowed incoming connections for autoregistration. Possible values: 1 - allow insecure connections; 2 - allow TLS with PSK. 3 - allow both insecure and TLS with PSK connections.
tls_psk_identity	string	*(writeonly)* PSK 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.
tls_psk	string	*(writeonly)* PSK value string (an even number of hexadecimal characters).

autoregistration.get

Description

`object autoregistration.get(object parameters)`

The method allows to retrieve autoregistration object according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports only one parameter.

Parameter	Type	Description
output	query	This parameter being common for all `get` methods described in the reference commentary.

Return values

`(object)` Returns autoregistration object.

Examples

Request:

```
{
   "jsonrpc": "2.0",
    "method": "autoregistration.get",
    "params": {
        "output": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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.

Parameters

`(object)` Autoregistration properties to be updated.

Return values

`(boolean )` Returns boolean true as result on successful update.

Examples

Request:

```
{
    "jsonrpc": "2.0",
    "method": "autoregistration.update",
    "params": {
        "tls_accept": "3",
        "tls_psk_identity": "PSK 001",
        "tls_psk": "11111595725ac58dd977beef14b97461a7c1045b9a1c923453302c5473193478"
```

```
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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.

Parameters

`(object)` Parameters defining the objects to be exported and the format to use.

Parameter	Type	Description
format (required)	string	Format in which the data must be exported. Possible values: `json` - JSON; `xml` - XML.
options (required)	object	Objects to be exported. The `options` object has the following parameters: `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; `screens` - `(array)` IDs of screens to export; `templates` - `(array)` IDs of templates to export; `valueMaps` - `(array)` IDs of value maps to export.

Return values

`(string)` Returns a serialized string containing the requested configuration data.

Examples

Exporting a template

Export the configuration of template "10571" as an XML string.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "configuration.export",
    "params": {
        "options": {
            "templates": [
                "10571"
            ]
        },
        "format": "xml"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<zabbix_export><version>5.0</version><date>2020
    "id": 1
}
```

Source

CConfiguration::export() in *ui/include/classes/api/services/CConfiguration.php*.

configuration.import

Description

`boolean configuration.import(object parameters)`

This method allows to import configuration data from a serialized string.

Parameters

`(object)` Parameters containing the data to import and rules how the data should be handled.

Parameter	Type	Description
format (required)	string	Format of the serialized string. Possible values: `json` - JSON; `xml` - XML.
source (required)	string	Serialized string containing the configuration data.
rules (required)	object	Rules on how new and existing objects should be imported. The `rules` parameter is described in detail in the table below.

Note:
If no rules are given, the configuration will not be updated.

The `rules` object supports the following parameters.

Parameter	Type	Description
applications	object	Rules on how to import applications. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new applications will be created; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, applications not present in the imported data will be deleted from the database; default: `false`.
discoveryRules	object	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`.
graphs	object	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`.
groups	object	Rules on how to import host groups. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new host groups will be created; default: `false`.
hosts	object	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`.
httptests	object	Rules on how to import web scenarios. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new web scenarios will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing web scenarios will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, web scenarios not present in the imported data will be deleted from the database; default: `false`.
images	object	Rules on how to import images. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new images will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing images will be updated; default: `false`.

Parameter	Type	Description
items	object	Rules on how to import items. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new items will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing items will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, items not present in the imported data will be deleted from the database; default: `false`.
maps	object	Rules on how to import maps. Supported parameters: `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`.
mediaTypes	object	Rules on how to import media types. Supported parameters: `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`.
screens	object	Rules on how to import screens. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new screens will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing screens will be updated; default: `false`.
templateLinkage	object	Rules on how to import template links. Supported parameters: `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`.
templates	object	Rules on how to import templates. Supported parameters: `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`.
templateScreens	object	Rules on how to import template screens. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new template screens will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing template screens will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, template screens not present in the imported data will be deleted from the database; default: `false`.

Parameter	Type	Description
triggers	object	Rules on how to import triggers. Supported parameters: `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`.
valueMaps	object	Rules on how to import value maps. 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`.

Return values

(`boolean`) Returns `true` if importing has been successful.

Examples

Importing a template

Import the template configuration contained in the XML string. If any items or triggers in the XML string are missing, they will be deleted from the database, and everything else will be left unchanged.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "configuration.import",
    "params": {
        "format": "xml",
        "rules": {
            "applications": {
                "createMissing": true
            },
            "templates": {
                "createMissing": true,
                "updateExisting": true
            },
            "items": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "triggers": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "valueMaps": {
                "createMissing": true,
                "updateExisting": false
            }
        },
        "source": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<zabbix_export><version>5.0</version><date>
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CConfiguration::import() 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.

Property	Type	Description
correlationid	string	*(readonly)* ID of the correlation.
name (required)	string	Name of the correlation.
description	string	Description of the correlation.
status	integer	Whether the correlation is enabled or disabled. Possible values are: 0 - *(default)* enabled; 1 - disabled.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Correlation operation

The correlation operation object defines an operation that will be performed when a correlation is executed. It has the following properties.

Property	Type	Description
type (required)	integer	Type of operation. Possible values: 0 - close old events; 1 - close new event.

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.

Property	Type	Description
evaltype (required)	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
conditions (required)	array	Set of filter conditions to use for filtering results.
eval_formula	string	*(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.
formula	string	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.

Correlation filter condition

The correlation filter condition object defines a specific condition that must be checked before running the correlation operations.

Property	Type	Description
type (required)	integer	Type of condition. Possible values: 0 - old event tag; 1 - new event tag; 2 - new event host group; 3 - event tag pair; 4 - old event tag value; 5 - new event tag value.
tag	string	Event tag (old or new). Required when type of condition is: 0, 1, 4, 5.
groupid	string	Host group ID. Required when type of condition is: 2.
oldtag	string	Old event tag. Required when type of condition is: 3.
newtag	string	New event tag. Required when type of condition is: 3.
value	string	Event tag (old or new) value. Required when type of condition is: 4, 5.
formulaid	string	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.
operator	integer	Condition operator. Required when type of condition is: 2, 4, 5.

Note:
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.

Condition	Condition name	Supported operators	Expected value
2	Host group	=, <>	Host group ID.
4	Old event tag value	=, <>, like, not like	string
5	New event tag value	=, <>, like, not like	string

correlation.create

Description

`object correlation.create(object/array correlations)`

This method allows to create new correlations.

Parameters

`(object/array)` Correlations to create.

Additionally to the standard correlation properties, the method accepts the following parameters.

Parameter	Type	Description
operations (required)	array	Correlation operations to create for the correlation.
filter (required)	object	Correlation filter object for the correlation.

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:

```
{
    "jsonrpc": "2.0",
    "method": "correlation.create",
    "params": {
        "name": "new event tag correlation",
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "type": 1,
                    "tag": "ok"
                }
            ]
        },
        "operations": [
            {
                "type": 0
            }
        ]
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "id": 1
}
```

Response:

```
{
    "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:

```
{
    "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"
                },
                {
                    "type": 2,
                    "operator": 1,
                    "formulaid": "B"
                }
            ]
        },
        "operations": [
            {
                "type": 1
            }
        ]
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "2"
        ]
    },
    "id": 1
}
```

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.

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:

```
{
    "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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
correlationids	string/array	Return only correlations with the given IDs.
selectFilter	query	Return a filter property with the correlation conditions.
selectOperations	query	Return an operations property with the correlation operations.
sortfield	string/array	Sort the result by the given properties. Possible values are: `correlationid`, `name` and `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in the reference commentary.
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

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.

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.

Parameter	Type	Description
filter	object	Correlation filter object to replace the current filter.
operations	array	Correlation operations to replace existing operations.

Return values

`(object)` Returns an object containing the IDs of the updated correlations under the `correlationids` property.

Examples

Disable correlation

Request:

```
{
    "jsonrpc": "2.0",
    "method": "correlation.update",
    "params": {
        "correlationid": "1",
        "status": "1"
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1"
        ]
    },
    "id": 1
}
```

Replace conditions, but keep the evaluation method

Request:

```
{
    "jsonrpc": "2.0",
    "method": "correlation.update",
    "params": {
        "correlationid": "1",
        "filter": {
            "conditions": [
                {
                    "type": 3,
                    "oldtag": "error",
                    "newtag": "ok"
                }
            ]
        }
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "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 widget
- Dashboard widget field
- Dashboard user group
- Dashboard user

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:

Property	Type	Description
dashboardid	string	*(readonly)* ID of the dashboard.
name (required)	string	Name of the dashboard.
userid	string	Dashboard owner user ID.
private	integer	Type of dashboard sharing. Possible values: 0 - public dashboard; 1 - *(default)* private dashboard.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Dashboard widget

The dashboard widget object has the following properties:

Property	Type	Description
widgetid	string	*(readonly)* ID of the dashboard widget.
type (required)	string	Type of the dashboard widget. Possible values: actionlog - Action log; clock - Clock; dataover - Data overview; discovery - Discovery status; favgraphs - Favorite graphs; favmaps - Favorite maps; favscreens - Favorite screens; graph - Graph (classic); graphprototype - Graph prototype; hostavail - Host availability; map - Map; navtree - Map Navigation Tree; plaintext - Plain text; problemhosts - Problem hosts; problems - Problems; problemsbysv - Problems by severity; svggraph - Graph; systeminfo - System information; trigover - Trigger overview; url - URL; web - Web monitoring;
name	string	Custom widget name.
x	integer	A horizontal position from the left side of the dashboard. Valid values range from 0 to 23.

Property	Type	Description
y	integer	A vertical position from the top of the dashboard. Valid values range from 0 to 62.
width	integer	The widget width. Valid values range from 1 to 24.
height	integer	The widget height. Valid values range from 2 to 32.
view_mode	integer	The widget view mode. Possible values: 0 - (default) default widget view; 1 - with hidden header;
fields	array	Array of the dashboard widget field objects.

Dashboard widget field

The dashboard widget field object has the following properties:

Property	Type	Description
type (required)	integer	Type of the widget field. Possible values: 0 - Integer; 1 - String; 2 - Host group; 3 - Host; 4 - Item; 5 - Item prototype; 6 - Graph; 7 - Graph prototype; 8 - Map.
name	string	Widget field name.
value (required)	mixed	Widget field value depending of type.

Dashboard user group

List of dashboard permissions based on user groups. It has the following properties:

Property	Type	Description
usrgrpid (required)	string	User group ID.
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write.

Dashboard user

List of dashboard permissions based on users. It has the following properties:

Property	Type	Description
userid (required)	string	User ID.

Property	Type	Description
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write.

dashboard.create

Description

`object dashboard.create(object/array dashboards)`

This method allows to create new dashboards.

Parameters

`(object/array)` Dashboards to create.

Additionally to the standard dashboard properties, the method accepts the following parameters.

Parameter	Type	Description
widgets	array	Dashboard widgets to be created for the dashboard.
users	array	Dashboard user shares to be created on the dashboard.
userGroups	array	Dashboard user group shares to be created on the dashboard.

Return values

`(object)` Returns an object containing the IDs of the created dashboards under the `dashboardids` 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).

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "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 widget
- Dashboard widget field
- Dashboard user
- Dashboatd 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.

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:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.delete",
    "params": [
        "2",
        "3"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
dashboardids	string/array	Return only dashboards with the given IDs.
selectWidgets	query	Return a widgets property with the dashboard widgets that are used in the dashboard.
selectUsers	query	Return a users property with users that the dashboard is shared with.
selectUserGroups	query	Return a userGroups property with user groups that the dashboard is shared with.
sortfield	string/array	Sort the result by the given properties. Possible value is: `dashboardid`.
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 a dashboard by ID

Retrieve all data about dashboards "1" and "2".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.get",
    "params": {
        "output": "extend",
        "selectWidgets": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
        "dashboardids": [
            "1",
            "2"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dashboardid": "1",
            "name": "Dashboard",
            "userid": "1",
            "private": "0",
            "users": [],
            "userGroups": [],
            "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": "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": "2",
                "type": "favscreens",
                "name": "",
                "x": "4",
                "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": []
                }
            ]
        },
        {
            "dashboardid": "2",
            "name": "My dashboard",
            "userid": "1",
            "private": "1",
            "users": [
                {
                    "userid": "4",
                    "permission": "3"
                }
            ],
            "userGroups": [
                {
                    "usrgrpid": "7",
                    "permission": "2"
                }
            ],
            "widgets": [
                {
                    "widgetid": "10",
                    "type": "problems",
                    "name": "",
                    "x": "0",
                    "y": "0",
                    "width": "12",
                    "height": "5",
                    "view_mode": 0,
                    "fields": [
                        {
                            "type": "2",
                            "name": "groupids",
                            "value": "4"
                        }
                    ]
                }
            ]
        }
    ],
    "id": 1
}
```


See also

- 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.

Parameters

`(object/array)` Dashboard properties to be updated.

The `dashboardid` property must be defined for each dashboard, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard dashboard properties, the method accepts the following parameters.

Parameter	Type	Description
widgets	array	Dashboard widgets to replace existing dashboard widgets. Dashboard widgets are updated by `widgetid` property. Widgets without `widgetid` property will be created.
users	array	Dashboard user shares to replace the existing elements.
userGroups	array	Dashboard user group shares to replace the existing elements.

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:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.update",
    "params": {
        "dashboardid": "2",
        "name": "SQL server status"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 1
}
```

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 widget
- Dashboard widget field
- Dashboard user
- Dashboard user group

Source

CDashboard::update() in *ui/include/classes/api/services/CDashboard.php*.

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

Note:
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.

Property	Type	Description
dhostid	string	ID of the discovered host.
druleid	string	ID of the discovery rule that detected the host.
lastdown	timestamp	Time when the discovered host last went down.
lastup	timestamp	Time when the discovered host last went up.

Property	Type	Description
status	integer	Whether the discovered host is up or down. A host is up if it has at least one active discovered service. 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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
dhostids	string/array	Return only discovered hosts with the given IDs.
druleids	string/array	Return only discovered hosts that have been created by the given discovery rules.
dserviceids	string/array	Return only discovered hosts that are running the given services.
selectDRules	query	Return a drules property with an array of the discovery rules that detected the host.
selectDServices	query	Return a dservices property with the discovered services running on the host. Supports `count`.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectDServices` - results will be sorted by `dserviceid`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `dhostid` and `druleid`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
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

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": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
        },
        {
            "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

> **Note:**
> Discovered services 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.

Property	Type	Description
dserviceid	string	ID of the discovered service.
dcheckid	string	ID of the discovery check used to detect the service.
dhostid	string	ID of the discovered host running the service.
dns	string	DNS of the host running the service.
ip	string	IP address of the host running the service.
lastdown	timestamp	Time when the discovered service last went down.
lastup	timestamp	Time when the discovered service last went up.
port	integer	Service port number.
status	integer	Status of the service. Possible values: 0 - service up; 1 - service down.
value	string	Value returned by the service when performing a Zabbix agent, SNMPv1, SNMPv2 or SNMPv3 discovery check.

dservice.get

Description

`integer/array dservice.get(object parameters)`

The method allows to retrieve discovered services according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
dserviceids	string/array	Return only discovered services with the given IDs.
dhostids	string/array	Return only discovered services that belong to the given discovered hosts.
dcheckids	string/array	Return only discovered services that have been detected by the given discovery checks.
druleids	string/array	Return only discovered services that have been detected by the given discovery rules.
selectDRules	query	Return a drules property with an array of the discovery rules that detected the service.

Parameter	Type	Description
selectDHosts	query	Return a dhosts property with an array the discovered hosts that the service belongs to.
selectHosts	query	Return a hosts property with the hosts with the same IP address and proxy as the service. Supports `count`.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectHosts` - result will be sorted by `hostid`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `dserviceid`, `dhostid` and `ip`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
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

Retrieve services discovered on a host

Retrieve all discovered services detected on discovered host "11".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dservice.get",
    "params": {
        "output": "extend",
        "dhostids": "11"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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
- Host

Source

CDService::get() in *ui/include/classes/api/services/CDService.php*.

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.

Property	Type	Description
dcheckid	string	*(readonly)* ID of the discovery check.
druleid	string	*(readonly)* ID of the discovery rule that the check belongs to.
key_	string	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.
ports	string	One or several port ranges to check separated by commas. Used for all checks except for ICMP. Default: 0.
snmp_community	string	SNMP community. Required for SNMPv1 and SNMPv2 agent checks.

Property	Type	Description
snmpv3_authpassphrase	string	Authentication passphrase used for SNMPv3 agent checks with security level set to *authNoPriv* or *authPriv*.
snmpv3_authprotocol	integer	Authentication protocol used for SNMPv3 agent checks with security level set to *authNoPriv* or *authPriv*. Possible values: 0 - *(default)* MD5; 1 - SHA.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 checks.
snmpv3_privpassphrase	string	Privacy passphrase used for SNMPv3 agent checks with security level set to *authPriv*.
snmpv3_privprotocol	integer	Privacy protocol used for SNMPv3 agent checks with security level set to *authPriv*. Possible values: 0 - *(default)* DES; 1 - AES.
snmpv3_securitylevel	string	Security level used for SNMPv3 agent checks. Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	Security name used for SNMPv3 agent checks.
type (required)	integer	Type of check. Possible values: 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.
uniq	integer	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. Possible values: 0 - *(default)* do not use this check as a uniqueness criteria; 1 - use this check as a uniqueness criteria.
host_source	integer	Source for host name. Possible values: 1 - *(default)* DNS; 2 - IP; 3 - discovery value of this check.

Property	Type	Description
name_source	integer	Source for visible name. Possible values: 0 - *(default)* not specified; 1 - DNS; 2 - IP; 3 - discovery value of this check.

dcheck.get

Description

`integer/array dcheck.get(object parameters)`

The method allows to retrieve discovery checks according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
dcheckids	string/array	Return only discovery checks with the given IDs.
druleids	string/array	Return only discovery checks that belong to the given discovery rules.
dserviceids	string/array	Return only discovery checks that have detected the given discovered services.
sortfield	string/array	Sort the result by the given properties. Possible values are: `dcheckid` and `druleid`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
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

Retrieve discovery checks for a discovery rule

Retrieve all discovery checks used by discovery rule "6".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dcheck.get",
    "params": {
        "output": "extend",
        "dcheckids": "6"
```

```
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dcheckid": "6",
            "druleid": "4",
            "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"
        }
    ],
    "id": 1
}
```

Source

CDCheck::get() in *ui/include/classes/api/services/CDCheck.php*.

Discovery rule

This class is designed to work with network discovery rules.

> **Note:**
> 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.

Property	Type	Description
druleid	string	*(readonly)* ID of the discovery rule.
iprange (required)	string	One or several IP ranges to check separated by commas. Refer to the network discovery configuration section for more information on supported formats of IP ranges.
name (required)	string	Name of the discovery rule.
delay	string	Execution interval of the discovery rule. Accepts seconds, time unit with suffix and user macro. Default: 1h.
nextcheck	timestamp	*(readonly)* Time when the discovery rule will be executed next.
proxy_hostid	string	ID of the proxy used for discovery.
status	integer	Whether the discovery rule is enabled. Possible values: 0 - *(default)* enabled; 1 - disabled.

Note that for some methods (update, delete) the required/optional parameter combination is different.

drule.create

Description

`object drule.create(object/array discoveryRules)`

This method allows to create new discovery rules.

Parameters

`(object/array)` Discovery rules to create.

Additionally to the standard discovery rule properties, the method accepts the following parameters.

Parameter	Type	Description
dchecks (required)	array	Discovery checks to create for the discovery rule.

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:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "6"
        ]
    },
    "id": 1
}
```

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.

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:

```
{
    "jsonrpc": "2.0",
    "method": "drule.delete",
    "params": [
        "4",
        "6"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
dhostids	string/array	Return only discovery rules that created the given discovered hosts.
druleids	string/array	Return only discovery rules with the given IDs.
dserviceids	string/array	Return only discovery rules that created the given discovered services.
selectDChecks	query	Return a dchecks property with the discovery checks used by the discovery rule. Supports `count`.
selectDHosts	query	Return a dhosts property with the discovered hosts created by the discovery rule. Supports `count`.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectDChecks` - results will be sorted by `dcheckid`; `selectDHosts` - results will be sorted by `dhostsid`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `druleid` and `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
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

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": "8",
                    "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",
            "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 *ui/include/classes/api/services/CDRule.php*.

drule.update

Description

`object drule.update(object/array discoveryRules)`

This method allows to update existing discovery rules.

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.

Parameter	Type	Description
dchecks	array	Discovery checks to replace existing checks.

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:

```
{
    "jsonrpc": "2.0",
    "method": "drule.update",
    "params": {
        "druleid": "6",
        "iprange": "192.168.2.1-255"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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

> **Note:**
> Events are created by the Zabbix server and cannot be modified via the API.

The event object has the following properties.

Property	Type	Description
eventid	string	ID of the event.
source	integer	Type of the event. Possible values: 0 - event created by a trigger; 1 - event created by a discovery rule; 2 - event created by active agent autoregistration; 3 - internal event.
object	integer	Type of object that is related to the event. Possible values for trigger events: 0 - trigger. Possible values for discovery events: 1 - discovered host; 2 - discovered service. Possible values for autoregistration events: 3 - auto-registered host. Possible values for internal events: 0 - trigger; 4 - item; 5 - LLD rule.
objectid	string	ID of the related object.
acknowledged	integer	Whether the event has been acknowledged.
clock	timestamp	Time when the event was created.
ns	integer	Nanoseconds when the event was created.
name	string	Resolved event name.
value	integer	State of the related object. Possible values for trigger events: 0 - OK; 1 - problem. Possible values for discovery events: 0 - host or service up; 1 - host or service down; 2 - host or service discovered; 3 - host or service lost. Possible values for internal events: 0 - ”normal” state; 1 - ”unknown” or ”not supported” state. This parameter is not used for active agent autoregistration events.
severity	integer	Event current severity. Possible values: 0 - not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
r_eventid	string	Recovery event ID
c_eventid	string	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.

Property	Type	Description
correlationid	string	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.
userid	string	User ID if the event was manually closed.
suppressed	integer	Whether the event is suppressed. Possible values: 0 - event is in normal state; 1 - event is suppressed.
opdata	string	Operational data with expanded macros.
urls	array	Active media type URLs.

Event tag

The event tag object has the following properties.

Property	Type	Description
tag	string	Event tag name.
value	string	Event tag value.

Media type URL

The media type URL object has the following properties.

Property	Type	Description
name	string	Media type defined URL name.
url	string	Media type defined URL value.

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 the properties contains an unexpanded macro, both properties will be excluded from results. For supported macros, see *Supported macros*.

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.

Attention:
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.

Parameters

`(object/array)` Parameters containing the IDs of the events and update operations that should be performed.

Parameter	Type	Description
eventids (required)	string/object	IDs of the events to acknowledge.
action (required)	integer	Event update action(s). Possible bitmap values are: 1 - close problem; 2 - acknowledge event; 4 - add message; 8 - change severity; 16 - unacknowledge event. This is bitmask field; any sum of possible bitmap values is acceptable (for example, 6 for acknowledge event and add message).
message	string	Text of the message. **Required**, if action contains 'add message' flag.
severity	integer	New severity for events. **Required**, if action contains 'change severity' flag. Possible values: 0 - not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.

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:

```
{
    "jsonrpc": "2.0",
    "method": "event.acknowledge",
    "params": {
        "eventids": "20427",
        "action": 6,
        "message": "Problem resolved."
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": [
            "20427"
        ]
    },
    "id": 1
}
```

Changing event's severity

Change severity for multiple events and leave a message.

Request:

```
{
    "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:

```
{
    "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.

Attention:
This method may return events of a deleted entity if these events have not been removed by the housekeeper yet.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
eventids	string/array	Return only events with the given IDs.
groupids	string/array	Return only events created by objects that belong to the given host groups.
hostids	string/array	Return only events created by objects that belong to the given hosts.
objectids	string/array	Return only events created by the given objects.
applicationids	string/array	Return only events created by objects that belong to the given applications. Applies only if object is trigger or item.
source	integer	Return only events with the given type. Refer to the event object page for a list of supported event types. Default: 0 - trigger events.

Parameter	Type	Description
object	integer	Return only events created by objects of the given type. Refer to the event object page for a list of supported object types. Default: 0 - trigger.
acknowledged	boolean	If set to `true` return only acknowledged events.
suppressed	boolean	`true` - return only suppressed events; `false` - return events in the normal state.
severities	integer/array	Return only events with given event severities. Applies only if object is trigger.
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.
tags	array of objects	Return only events with given tags. Exact match by tag and case-insensitive search by value and operator. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all events. Possible operator types: 0 - (default) Like; 1 - Equal.
eventid_from	string	Return only events with IDs greater or equal to the given ID.
eventid_till	string	Return only events with IDs less or equal to the given ID.
time_from	timestamp	Return only events that have been created after or at the given time.
time_till	timestamp	Return only events that have been created before or at the given time.
problem_time_from	timestamp	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.
problem_time_till	timestamp	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.
value	integer/array	Return only events with the given values.
selectHosts	query	Return a hosts property with hosts containing the object that created the event. Supported only for events generated by triggers, items or LLD rules.
selectRelatedObject	query	Return a `relatedObject` property with the object that created the event. The type of object returned depends on the event type.
select_alerts	query	Return an alerts property with alerts generated by the event. Alerts are sorted in reverse chronological order.
select_acknowledges	query	Return an `acknowledges` property with event updates. Event updates are sorted in reverse chronological order. The event update object has the following properties: `acknowledgeid` - `(string)` acknowledgment'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)` update action that was performed see event.acknowledge; `old_severity` - `(integer)` event severity before this update action; `new_severity` - `(integer)` event severity after this update action; `alias` - `(string)` alias of the user that updated the event; `name` - `(string)` name of the user that updated the event; `surname` - `(string)` surname of the user that updated the event. Supports `count`.
selectTags	query	Return a tags property with event tags.

Parameter	Type	Description
selectSuppressionData	query	Return a `suppression_data` property with the list of maintenances: `maintenanceid` - `(string)` ID of the maintenance; `suppress_until` - `(integer)` time until the event is suppressed.
sortfield	string/array	Sort the result by the given properties. Possible values are: `eventid`, `objectid` 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	
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 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",
        "objectids": "13926",
        "sortfield": ["clock", "eventid"],
        "sortorder": "DESC"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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----[BULK ACKNOWLEDGE]----",
                    "action": "6",
                    "old_severity": "0",
                    "new_severity": "0",
                    "alias": "Admin",
                    "name": "Zabbix",
                    "surname": "Administrator"
                }
            ],
            "suppression_data": [
                {
                    "maintenanceid": "15",
                    "suppress_until": "1472511600"
                }
            ],
            "suppressed": "1",
            "tags": [
                {
                    "tag": "service",
                    "value": "mysqld"
                },
                {
                    "tag": "error",
                    "value": ""
                }
            ]
        },
        {
            "eventid": "9671",
            "source": "0",
            "object": "0",
            "objectid": "13926",
            "clock": "1347970347",
            "value": "0",
            "acknowledged": "0",
            "ns": "0",
            "name": "Unavailable by ICMP ping",
            "severity": "0",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "opdata": "",
            "acknowledges": [],
            "suppression_data": [],
            "suppressed": "0",
            "tags": []
        }
    ],
```

```
    "id": 1
}
```

Retrieving events by time period

Retrieve all events that have been created between October 9 and 19, 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": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "eventid": "20618",
            "source": "0",
            "object": "0",
            "objectid": "14284",
            "clock": "1350477819",
            "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",
            "opdata": "",
            "suppressed": "0"
        },
        {
            "eventid": "20617",
            "source": "0",
            "object": "0",
            "objectid": "14283",
            "clock": "1350477817",
            "value": "0",
            "acknowledged": "0",
            "ns": "0",
            "name": "Zabbix trapper processes more than 75% busy",
            "severity": "0",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "opdata": "",
            "suppressed": "0"
        },
        {
```

```
            "eventid": "20616",
            "source": "0",
            "object": "0",
            "objectid": "14282",
            "clock": "1350477815",
            "value": "1",
            "acknowledged": "0",
            "ns": "0",
            "name": "High ICMP ping loss",
            "severity": "3",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "opdata": "",
            "suppressed": "0"
        }
    ],
    "id": 1
}
```


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 graphs.

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.

Property	Type	Description
graphid	string	*(readonly)* ID of the graph.
height (required)	integer	Height of the graph in pixels.
name (required)	string	Name of the graph
width (required)	integer	Width of the graph in pixels.

Property	Type	Description
flags	integer	*(readonly)* Origin of the graph. Possible values are: 0 - *(default)* a plain graph; 4 - a discovered graph.
graphtype	integer	Graph's layout type. Possible values: 0 - *(default)* normal; 1 - stacked; 2 - pie; 3 - exploded.
percent_left	float	Left percentile. Default: 0.
percent_right	float	Right percentile. Default: 0.
show_3d	integer	Whether to show pie and exploded graphs in 3D. Possible values: 0 - *(default)* show in 2D; 1 - show in 3D.
show_legend	integer	Whether to show the legend on the graph. Possible values: 0 - hide; 1 - *(default)* show.
show_work_period	integer	Whether to show the working time on the graph. Possible values: 0 - hide; 1 - *(default)* show.
show_triggers	integer	Whether to show the trigger line on the graph. Possible values: 0 - hide; 1 - *(default)* show.
templateid	string	*(readonly)* ID of the parent template graph.
yaxismax	float	The fixed maximum value for the Y axis. Starting with Zabbix 5.0.26, if user have no access to specified item, the graph is rendered like ymax_type would be set to '0' (calculated). Default: 100.
yaxismin	float	The fixed minimum value for the Y axis. Starting with Zabbix 5.0.26, if user have no access to specified item, the graph is rendered like ymin_type would be set to '0' (calculated). Default: 0.
ymax_itemid	string	ID of the item that is used as the maximum value for the Y axis.
ymax_type	integer	Maximum value calculation method for the Y axis. Possible values: 0 - *(default)* calculated; 1 - fixed; 2 - item.
ymin_itemid	string	ID of the item that is used as the minimum value for the Y axis.

Property	Type	Description
ymin_type	integer	Minimum value calculation method for the Y axis. Possible values: 0 - *(default)* calculated; 1 - fixed; 2 - item.

Note that for some methods (update, delete) the required/optional parameter combination is different.

graph.create

Description

`object graph.create(object/array graphs)`

This method allows to create new graphs.

Parameters

`(object/array)` Graphs to create.

Additionally to the standard graph properties, the method accepts the following parameters.

Parameter	Type	Description
gitems (required)	array	Graph items to be created for the graph.

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": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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:

```
{
    "jsonrpc": "2.0",
    "method": "graph.delete",
    "params": [
        "652",
        "653"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
graphids	string/array	Return only graphs with the given IDs.
groupids	string/array	Return only graphs that belong to hosts in the given host groups.
templateids	string/array	Return only graph that belong to the given templates.
hostids	string/array	Return only graphs that belong to the given hosts.
itemids	string/array	Return only graphs that contain the given items.
templated	boolean	If set to `true` return only graphs that belong to templates.
inherited	boolean	If set to `true` return only graphs inherited from a template.
expandName	flag	Expand macros in the graph name.
selectGroups	query	Return a groups property with the host groups that the graph belongs to.
selectTemplates	query	Return a templates property with the templates that the graph belongs to.
selectHosts	query	Return a hosts property with the hosts that the graph belongs to.
selectItems	query	Return an items property with the items used in the graph.
selectGraphDiscovery	query	Return a `graphDiscovery` property with the graph discovery object. The graph discovery objects links the graph to a graph prototype from which it was created. It has the following properties: `graphid` - `(string)` ID of the graph; `parent_graphid` - `(string)` ID of the graph prototype from which the graph has been created.
selectGraphItems	query	Return a gitems property with the items used in the graph.
selectDiscoveryRule	query	Return a discoveryRule property with the low-level discovery rule that created the graph.
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 graph belongs to; `hostid` - ID of the host that the graph belongs to.
sortfield	string/array	Sort the result by the given properties. Possible values are: `graphid`, `name` and `graphtype`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
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 graphs from hosts

Retrieve all graphs from host "10107" and sort them by name.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graph.get",
    "params": {
        "output": "extend",
        "hostids": 10107,
        "sortfield": "name"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

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": "1",
            "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": "0",
            "graphtype": "1",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0",
            "percent_right": "0",
            "ymin_type": "1",
            "ymax_type": "1",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "645",
            "name": "Disk space usage /",
            "width": "600",
            "height": "340",
            "yaxismin": "0",
            "yaxismax": "0",
            "templateid": "0",
            "show_work_period": "0",
            "show_triggers": "0",
            "graphtype": "2",
            "show_legend": "1",
            "show_3d": "1",
            "percent_left": "0",
            "percent_right": "0",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "4"
        }
    ],
    "id": 1
}
```


See also

- Discovery rule
- Graph item
- Item
- Host
- Host group
- Template

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.

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.

Parameter	Type	Description
gitems	array	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.

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:

```
{
    "jsonrpc": "2.0",
    "method": "graph.update",
    "params": {
        "graphid": "439",
        "ymax_type": 1,
        "yaxismax": 100
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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 graph items.

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

> **Note:**
> Graph items can only be modified via the `graph` API.

The graph item object has the following properties.

Property	Type	Description
gitemid	string	*(readonly)* ID of the graph item.
color (required)	string	Graph item's draw color as a hexadecimal color code.
itemid (required)	string	ID of the item.
calc_fnc	integer	Value of the item that will be displayed. Possible values: 1 - minimum value; 2 - *(default)* average value; 4 - maximum value; 7 - all values; 9 - last value, used only by pie and exploded graphs.
drawtype	integer	Draw style of the graph item. Possible values: 0 - *(default)* line; 1 - filled region; 2 - bold line; 3 - dot; 4 - dashed line; 5 - gradient line.
graphid	string	ID of the graph that the graph item belongs to.
sortorder	integer	Position of the item in the graph. Default: starts with 0 and increases by one with each entry.
type	integer	Type of graph item. Possible values: 0 - *(default)* simple; 2 - graph sum, used only by pie and exploded graphs.
yaxisside	integer	Side of the graph where the graph item's Y scale will be drawn. Possible values: 0 - *(default)* left side; 1 - right side.

graphitem.get

Description

`integer/array graphitem.get(object parameters)`

The method allows to retrieve graph items according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
graphids	string/array	Return only graph items that belong to the given graphs.
itemids	string/array	Return only graph items with the given item IDs.
type	integer	Return only graph items with the given type. Refer to the graph item object page for a list of supported graph item types.
selectGraphs	query	Return a graphs property with an array of graphs that the item belongs to.
sortfield	string/array	Sort the result by the given properties. Possible values are: `gitemid`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
limit	integer	
output	query	
preservekeys	boolean	
sortorder	string/array	

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 items from a graph

Retrieve all graph items used in a graph with additional information about the item and the host.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graphitem.get",
    "params": {
        "output": "extend",
        "graphids": "387"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "gitemid": "1242",
            "graphid": "387",
            "itemid": "22665",
            "drawtype": "1",
            "sortorder": "1",
            "color": "FF5555",
```

```
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0"
        },
        {
            "gitemid": "1243",
            "graphid": "387",
            "itemid": "22668",
            "drawtype": "1",
            "sortorder": "2",
            "color": "55FF55",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0"
        },
        {
            "gitemid": "1244",
            "graphid": "387",
            "itemid": "22671",
            "drawtype": "1",
            "sortorder": "3",
            "color": "009999",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0"
        }
    ],
    "id": 1
}
```


See also

- Graph

Source

CGraphItem::get() in *ui/include/classes/api/services/CGraphItem.php*.

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.

Property	Type	Description
graphid	string	*(readonly)* ID of the graph prototype.

Property	Type	Description
height (required)	integer	Height of the graph prototype in pixels.
name (required)	string	Name of the graph prototype.
width (required)	integer	Width of the graph prototype in pixels.
graphtype	integer	Graph prototypes's layout type. Possible values: 0 - *(default)* normal; 1 - stacked; 2 - pie; 3 - exploded.
percent_left	float	Left percentile. Default: 0.
percent_right	float	Right percentile. Default: 0.
show_3d	integer	Whether to show discovered pie and exploded graphs in 3D. Possible values: 0 - *(default)* show in 2D; 1 - show in 3D.
show_legend	integer	Whether to show the legend on the discovered graph. Possible values: 0 - hide; 1 - *(default)* show.
show_work_period	integer	Whether to show the working time on the discovered graph. Possible values: 0 - hide; 1 - *(default)* show.
templateid	string	*(readonly)* ID of the parent template graph prototype.
yaxismax	float	The fixed maximum value for the Y axis. Starting with Zabbix 5.0.26, if user have no access to specified item, the graph is rendered like ymax_type would be set to '0' (calculated).
yaxismin	float	The fixed minimum value for the Y axis. Starting with Zabbix 5.0.26, if user have no access to specified item, the graph is rendered like ymin_type would be set to '0' (calculated).
ymax_itemid	string	ID of the item that is used as the maximum value for the Y axis.
ymax_type	integer	Maximum value calculation method for the Y axis. Possible values: 0 - *(default)* calculated; 1 - fixed; 2 - item.
ymin_itemid	string	ID of the item that is used as the minimum value for the Y axis.
ymin_type	integer	Minimum value calculation method for the Y axis. Possible values: 0 - *(default)* calculated; 1 - fixed; 2 - item.

Property	Type	Description
discover	integer	Graph prototype discovery status. Possible values: 0 - *(default)* new graphs will be discovered; 1 - new graphs will not be discovered and existing graphs will be marked as lost.

Note that for some methods (update, delete) the required/optional parameter combination is different.

graphprototype.create

Description

`object graphprototype.create(object/array graphPrototypes)`

This method allows to create new graph prototypes.

Parameters

`(object/array)` Graph prototypes to create.

Additionally to the standard graph prototype properties, the method accepts the following parameters.

Parameter	Type	Description
gitems (required)	array	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.

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:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652"
        ]
    },
    "id": 1
}
```

See also

- Graph item

Source

CGraphPrototype::create() in *ui/include/classes/api/services/CGraphPrototype.php*.

graphprototype.delete

Description

`object graphprototype.delete(array graphPrototypeIds)`

This method allows to delete graph prototypes.

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:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.delete",
    "params": [
        "652",
        "653"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652",
            "653"
        ]
    },
    "id": 1
}
```

Source

CGraphPrototype::delete() in *ui/include/classes/api/services/CGraphPrototype.php*.

graphprototype.get

Description

`integer/array graphprototype.get(object parameters)`

The method allows to retrieve graph prototypes according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
discoveryids	string/array	Return only graph prototypes that belong to the given discovery rules.
graphids	string/array	Return only graph prototypes with the given IDs.
groupids	string/array	Return only graph prototypes that belong to hosts in the given host groups.
hostids	string/array	Return only graph prototypes that belong to the given hosts.
inherited	boolean	If set to `true` return only graph prototypes inherited from a template.
itemids	string/array	Return only graph prototypes that contain the given item prototypes.
templated	boolean	If set to `true` return only graph prototypes that belong to templates.
templateids	string/array	Return only graph prototypes that belong to the given templates.
selectDiscoveryRule	query	Return a discoveryRule property with the LLD rule that the graph prototype belongs to.
selectGraphItems	query	Return a gitems property with the graph items used in the graph prototype.
selectGroups	query	Return a groups property with the host groups that the graph prototype belongs to.
selectHosts	query	Return a hosts property with the hosts that the graph prototype belongs to.
selectItems	query	Return an `items` property with the items and item prototypes used in the graph prototype.
selectTemplates	query	Return a templates property with the templates that the graph prototype belongs to.
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 graph prototype belongs to; `hostid` - ID of the host that the graph prototype belongs to.
sortfield	string/array	Sort the result by the given properties. Possible values are: `graphid`, `name` and `graphtype`.
countOutput	boolean	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	

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:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.get",
    "params": {
        "output": "extend",
        "discoveryids": "27426"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "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
- Graph item
- Item
- Host
- Host group
- Template

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.

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.

Parameter	Type	Description
gitems	array	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.

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"
        ]
    },
    "id": 1
}
```

Source

CGraphPrototype::update() in *ui/include/classes/api/services/CGraphPrototype.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.

Note:
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.

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	float	Received value.

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.

String history

The string 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	string	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.

Property	Type	Description
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.get

Description

`integer/array history.get(object parameters)`

The method allows to retrieve history data according to the given parameters.

Attention:
This method may return historical data of a deleted entity if this data has not been removed by the housekeeper yet.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
history	integer	History object types to return. Possible values: 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text. Default: 3.
hostids	string/array	Return only history from the given hosts.
itemids	string/array	Return only history from the given items.
time_from	timestamp	Return only values that have been received after or at the given time.
time_till	timestamp	Return only values that have been received before or at the given time.
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:

```
{
    "jsonrpc": "2.0",
    "method": "history.get",
    "params": {
        "output": "extend",
        "history": 0,
        "itemids": "23296",
        "sortfield": "clock",
        "sortorder": "DESC",
        "limit": 10
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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.

Property	Type	Description
hostid	string	*(readonly)* ID of the host.
host (required)	string	Technical name of the host.

Property	Type	Description
available	integer	*(readonly)* Availability of Zabbix agent. Possible values are: 0 - *(default)* unknown; 1 - available; 2 - unavailable.
description	text	Description of the host.
disable_until	timestamp	*(readonly)* The next polling time of an unavailable Zabbix agent.
error	string	*(readonly)* Error text if Zabbix agent is unavailable.
errors_from	timestamp	*(readonly)* Time when Zabbix agent became unavailable.
flags	integer	*(readonly)* Origin of the host. Possible values: 0 - a plain host; 4 - a discovered host.
inventory_mode	integer	Host inventory population mode. Possible values are: -1 - *(default)* disabled; 0 - manual; 1 - automatic.
ipmi_authtype	integer	IPMI authentication algorithm. Possible values are: -1 - *(default)* default; 0 - none; 1 - MD2; 2 - MD5 4 - straight; 5 - OEM; 6 - RMCP+.
ipmi_available	integer	*(readonly)* Availability of IPMI agent. Possible values are: 0 - *(default)* unknown; 1 - available; 2 - unavailable.
ipmi_disable_until	timestamp	*(readonly)* The next polling time of an unavailable IPMI agent.
ipmi_error	string	*(readonly)* Error text if IPMI agent is unavailable.
ipmi_errors_from	timestamp	*(readonly)* Time when IPMI agent became unavailable.
ipmi_password	string	IPMI password.
ipmi_privilege	integer	IPMI privilege level. Possible values are: 1 - callback; 2 - *(default)* user; 3 - operator; 4 - admin; 5 - OEM.
ipmi_username	string	IPMI username.
jmx_available	integer	*(readonly)* Availability of JMX agent. Possible values are: 0 - *(default)* unknown; 1 - available; 2 - unavailable.
jmx_disable_until	timestamp	*(readonly)* The next polling time of an unavailable JMX agent.
jmx_error	string	*(readonly)* Error text if JMX agent is unavailable.
jmx_errors_from	timestamp	*(readonly)* Time when JMX agent became unavailable.
maintenance_from	timestamp	*(readonly)* Starting time of the effective maintenance.

Property	Type	Description
maintenance_status	integer	*(readonly)* Effective maintenance status. Possible values are: 0 - *(default)* no maintenance; 1 - maintenance in effect.
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.
snmp_available	integer	*(readonly)* Availability of SNMP agent. Possible values are: 0 - *(default)* unknown; 1 - available; 2 - unavailable.
snmp_disable_until	timestamp	*(readonly)* The next polling time of an unavailable SNMP agent.
snmp_error	string	*(readonly)* Error text if SNMP agent is unavailable.
snmp_errors_from	timestamp	*(readonly)* Time when SNMP agent became unavailable.
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. This is a bitmask field; any sum of possible bitmap values is acceptable (for example, 6 for PSK and certificate).
tls_issuer	string	Certificate issuer.
tls_subject	string	Certificate subject.
tls_psk_identity	string	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	The preshared key, at least 32 hex digits. Required if either `tls_connect` or `tls_accept` has PSK enabled.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Host inventory

The host inventory object has the following properties.

Note:
Each property has it's own unique ID number, which is used to associate host inventory fields with items.

ID	Property	Type	Description	Maximum length
4	alias	string	Alias.	128 characters
11	asset_tag	string	Asset tag.	64 characters
28	chassis	string	Chassis.	64 characters
23	contact	string	Contact person.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
32	contract_number	string	Contract number.	64 characters
47	date_hw_decomm	string	HW decommissioning date.	64 characters
46	date_hw_expiry	string	HW maintenance expiry date.	64 characters
45	date_hw_install	string	HW installation date.	64 characters
44	date_hw_purchase	string	HW purchase date.	64 characters
34	deployment_status	string	Deployment status.	64 characters
14	hardware	string	Hardware.	255 characters
15	hardware_full	string	Detailed hardware.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
39	host_netmask	string	Host subnet mask.	39 characters
38	host_networks	string	Host networks.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
40	host_router	string	Host router.	39 characters
30	hw_arch	string	HW architecture.	32 characters
33	installer_name	string	Installer name.	64 characters
24	location	string	Location.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
25	location_lat	string	Location latitude.	16 characters
26	location_lon	string	Location longitude.	16 characters
12	macaddress_a	string	MAC address A.	64 characters
13	macaddress_b	string	MAC address B.	64 characters
29	model	string	Model.	64 characters
3	name	string	Name.	128 characters
27	notes	string	Notes.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
41	oob_ip	string	OOB IP address.	39 characters
42	oob_netmask	string	OOB host subnet mask.	39 characters
43	oob_router	string	OOB router.	39 characters
5	os	string	OS name.	128 characters
6	os_full	string	Detailed OS name.	255 characters
7	os_short	string	Short OS name.	128 characters
61	poc_1_cell	string	Primary POC mobile number.	64 characters
58	poc_1_email	string	Primary email.	128 characters
57	poc_1_name	string	Primary POC name.	128 characters
63	poc_1_notes	string	Primary POC notes.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
59	poc_1_phone_a	string	Primary POC phone A.	64 characters
60	poc_1_phone_b	string	Primary POC phone B.	64 characters
62	poc_1_screen	string	Primary POC screen name.	64 characters
68	poc_2_cell	string	Secondary POC mobile number.	64 characters
65	poc_2_email	string	Secondary POC email.	128 characters
64	poc_2_name	string	Secondary POC name.	128 characters
70	poc_2_notes	string	Secondary POC notes.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
66	poc_2_phone_a	string	Secondary POC phone A.	64 characters
67	poc_2_phone_b	string	Secondary POC phone B.	64 characters
69	poc_2_screen	string	Secondary POC screen name.	64 characters
8	serialno_a	string	Serial number A.	64 characters
9	serialno_b	string	Serial number B.	64 characters

ID	Property	Type	Description	Maximum length
48	site_address_a	string	Site address A.	128 characters
49	site_address_b	string	Site address B.	128 characters
50	site_address_c	string	Site address C.	128 characters
51	site_city	string	Site city.	128 characters
53	site_country	string	Site country.	64 characters
56	site_notes	string	Site notes.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
55	site_rack	string	Site rack location.	128 characters
52	site_state	string	Site state.	64 characters
54	site_zip	string	Site ZIP/postal code.	64 characters
16	software	string	Software.	255 characters
18	software_app_a	string	Software application A.	64 characters
19	software_app_b	string	Software application B.	64 characters
20	software_app_c	string	Software application C.	64 characters
21	software_app_d	string	Software application D.	64 characters
22	software_app_e	string	Software application E.	64 characters
17	software_full	string	Software details.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
10	tag	string	Tag.	64 characters
1	type	string	Type.	64 characters
2	type_full	string	Type details.	64 characters
35	url_a	string	URL A.	255 characters
36	url_b	string	URL B.	255 characters
37	url_c	string	URL C.	255 characters
31	vendor	string	Vendor.	64 characters

Host tag

The host tag object has the following properties.

Property	Type	Description
tag (required)	string	Host tag name.
value	string	Host tag value.

host.create

Description

`object host.create(object/array hosts)`

This method allows to create new hosts.

Parameters

`(object/array)` Hosts to create.

Additionally to the standard host properties, the method accepts the following parameters.

Parameter	Type	Description
groups (required)	object/array	Host groups to add the host to. The host groups must have the `groupid` property defined.
interfaces (required)	object/array	Interfaces to be created for the host.
tags	object/array	Host tags.
templates	object/array	Templates to be linked to the host. The templates must have the `templateid` property defined.

Parameter	Type	Description
macros	object/array	User macros to be created for the host.
inventory	object	Host inventory properties.

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",
                "dns": "",
                "port": "10050"
            }
        ],
        "groups": [
            {
                "groupid": "50"
            }
        ],
        "tags": [
            {
                "tag": "Host name",
                "value": "Linux server"
            }
        ],
        "templates": [
            {
                "templateid": "20045"
            }
        ],
        "macros": [
            {
                "macro": "{$USER_ID}",
                "value": "123321"
            },
            {
                "macro": "{$USER_LOCATION}",
                "value": "0:0:0",
                "description": "latitude, longitude and altitude coordinates"
            }
        ],
        "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": [
            {
                "type": 2,
                "main": 1,
                "useip": 1,
                "ip": "127.0.0.1",
                "dns": "",
                "port": "161",
                "details": {
                    "version": 3,
                    "bulk": 0,
                    "securityname": "mysecurityname",
                    "contextname": "",
                    "securitylevel": 1
                }
            }
        ],
        "groups": [
            {
                "groupid": "4"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10658"
        ]
    },
    "id": 1
}
```

Creating a host with PSK encryption

Create a host called "PSK host" with PSK encryption configured. Note that the host has to be pre-configured to use PSK.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "PSK host",
        "interfaces": [
            {
                "type": 1,
                "ip": "192.168.3.1",
                "dns": "",
                "port": "10050",
                "useip": 1,
                "main": 1
            }
        ],
        "groups": [
            {
                "groupid": "2"
            }
        ],
        "tls_accept": 2,
        "tls_connect": 2,
        "tls_psk_identity": "PSK 001",
        "tls_psk": "1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10590"
        ]
    },
    "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.

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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
groupids	string/array	Return only hosts that belong to the given groups.
applicationids	string/array	Return only hosts that have the given applications.
dserviceids	string/array	Return only hosts that are related to the given discovered services.
graphids	string/array	Return only hosts that have the given graphs.
hostids	string/array	Return only hosts with the given host IDs.
httptestids	string/array	Return only hosts that have the given web checks.
interfaceids	string/array	Return only hosts that use the given interfaces.
itemids	string/array	Return only hosts that have the given items.
maintenanceids	string/array	Return only hosts that are affected by the given maintenances.
monitored_hosts	flag	Return only monitored hosts.
proxy_hosts	flag	Return only proxies.

Parameter	Type	Description
proxyids	string/array	Return only hosts that are monitored by the given proxies.
templated_hosts	flag	Return both hosts and templates.
templateids	string/array	Return only hosts that are linked to the given templates.
triggerids	string/array	Return only hosts that have the given triggers.
with_items	flag	Return only hosts that have items. Overrides the `with_monitored_items` and `with_simple_graph_items` parameters.
with_item_prototypes	flag	Return only hosts that have item prototypes. Overrides the `with_simple_graph_item_prototypes` parameter.
with_simple_graph_item_prototypes	flag	Return only hosts that have item prototypes, which are enabled for creation and have numeric type of information.
with_applications	flag	Return only hosts that have applications.
with_graphs	flag	Return only hosts that have graphs.
with_graph_prototypes	flag	Return only hosts that have graph prototypes.
with_httptests	flag	Return only hosts that have web checks. Overrides the `with_monitored_httptests` parameter.
with_monitored_httptests	flag	Return only hosts that have enabled web checks.
with_monitored_items	flag	Return only hosts that have enabled items. Overrides the `with_simple_graph_items` parameter.
with_monitored_triggers	flag	Return only hosts that have enabled triggers. All of the items used in the trigger must also be enabled.
with_simple_graph_items	flag	Return only hosts that have items with numeric type of information.
with_triggers	flag	Return only hosts that have triggers. Overrides the `with_monitored_triggers` parameter.
withProblemsSuppressed	boolean	Return hosts that have suppressed problems. Possible values: `null` - (default) all hosts; `true` - only hosts with suppressed problems; `false` - only hosts with unsuppressed problems.
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.
severities	integer/array	Return hosts that have only problems with given severities. Applies only if problem object is trigger.
tags	array/object	Return only hosts 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 hosts. Possible operator values: 0 - (default) Contains; 1 - Equals.
inheritedTags	boolean	Return hosts that have given `tags` also in all of their linked templates. Default: Possible values: `true` - linked templates must also have given `tags`; `false` - (default) linked template tags are ignored.
selectApplications	query	Return an applications property with host applications. Supports `count`.

Parameter	Type	Description
selectDiscoveries	query	Return a discoveries property with host low-level discovery rules. Supports `count`.
selectDiscoveryRule	query	Return a discoveryRule property with the low-level discovery rule that created the host (from host prototype in VMware monitoring).
selectGraphs	query	Return a graphs property with host graphs. Supports `count`.
selectGroups	query	Return a groups property with host groups data that the host belongs to.
selectHostDiscovery	query	Return a `hostDiscovery` property with host discovery object data. 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: `host` - *(string)* host of the host prototype; `hostid` - *(string)* ID of the discovered host or host prototype; `parent_hostid` - *(string)* ID of the host prototype from which the host has been created; `parent_itemid` - *(string)* ID of the LLD rule that created the discovered host; `lastcheck` - *(timestamp)* time when the host was last discovered; `ts_delete` - *(timestamp)* time when a host that is no longer discovered will be deleted.
selectHttpTests	query	Return an httpTests property with host web scenarios. Supports `count`.
selectInterfaces	query	Return an interfaces property with host interfaces. Supports `count`.
selectInventory	query	Return an inventory property with host inventory data.
selectItems	query	Return an items property with host items. Supports `count`.
selectMacros	query	Return a macros property with host macros.
selectParentTemplates	query	Return a parentTemplates property with templates that the host is linked to. Supports `count`.
selectScreens	query	Return a screens property with host screens. Supports `count`.
selectTags	query	Return a tags property with host tags.
selectInheritedTags	query	Return an inheritedTags property with tags that are on all templates which are linked to host.
selectTriggers	query	Return a triggers property with host triggers. Supports `count`.
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. Allows filtering by interface properties. Doesn't work for `text` fields.

Parameter	Type	Description
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectParentTemplates` - results will be sorted by `host`; `selectInterfaces`; `selectItems` - sorted by `name`; `selectDiscoveries` - sorted by `name`; `selectTriggers` - sorted by `description`; `selectGraphs` - sorted by `name`; `selectApplications` - sorted by `name`; `selectScreens` - sorted by `name`.
search	object	Return results that match the given pattern (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 "%…%"` search. Allows searching by interface properties. Works only for `string` and `text` fields.
searchInventory	object	Return only hosts that have inventory data matching the given wildcard search. This parameter is affected by the same additional parameters as `search`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `hostid`, `host`, `name`, `status`.
countOutput	boolean	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	
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 data by name

Retrieve all data about two hosts named ”Zabbix server” and ”Linux server”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "filter": {
            "host": [
                "Zabbix server",
                "Linux server"
            ]
        }
```

```
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "maintenances": [],
            "hostid": "10160",
            "proxy_hostid": "0",
            "host": "Zabbix server",
            "status": "0",
            "disable_until": "0",
            "error": "",
            "available": "0",
            "errors_from": "0",
            "lastaccess": "0",
            "ipmi_authtype": "-1",
            "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": "Zabbix server",
            "description": "The Zabbix monitoring server.",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": ""
        },
        {
            "maintenances": [],
            "hostid": "10167",
            "proxy_hostid": "0",
            "host": "Linux server",
            "status": "0",
            "disable_until": "0",
            "error": "",
            "available": "0",
            "errors_from": "0",
            "lastaccess": "0",
            "ipmi_authtype": "-1",
            "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": "Linux server",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": ""
        }
    ],
    "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"],
        "selectGroups": "extend",
        "filter": {
            "host": [
                "Zabbix server"
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "parentTemplates": [
                {
                    "name": "Template OS Linux",
                    "templateid": "10001"
                },
                {
                    "name": "Template App Zabbix Server",
                    "templateid": "10047"
                }
            ]
        }
    ],
    "id": 1
}
```

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:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectTags": "extend",
        "evaltype": 0,
        "tags": [
            {
                "tag": "Host name",
                "value": "Linux server",
                "operator": 1
            }
        ]
    },
    "auth": "7f9e00124c75e8f25facd5c093f3e9a0",
    "id": 1
}
```

Response:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "tags": [{"tag": "A", "value": "1", "operator": "0"}],
        "inheritedTags": true
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10623",
            "name": "PC room 1"
        },
        {
            "hostid": "10601",
            "name": "Office"
        }
    ],
    "id": 1
}
```

Searching host with tags and template tags

Retrieve a host with tags and all tags that are linked to parent templates.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "hostids": 10502,
        "selectTags": ["tag", "value"],
        "selectInheritedTags": ["tag", "value"]
    },
```

```
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "severities": 5
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10160",
            "name": "Zabbix server"
        }
    ],
    "id": 1
}
```

Retrieve hosts that have ”Average” and ”High” problems.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
```

```
        "output": ["name"],
        "severities": [3, 4]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "20170",
            "name": "Database"
        },
        {
            "hostid": "20183",
            "name": "workstation"
        }
    ],
    "id": 1
}
```

See also

- Host group
- Template
- User macro
- Host interface

Source

CHost::get() in *ui/include/classes/api/services/CHost.php*.

host.massadd

Description

`object host.massadd(object parameters)`

This method allows to simultaneously add multiple related objects to all the given hosts.

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.

Parameter	Type	Description
hosts (required)	object/array	Hosts to be updated. The hosts must have the `hostid` property defined.
groups	object/array	Host groups to add to the given hosts. The host groups must have the `groupid` property defined.
interfaces	object/array	Host interfaces to be created for the given hosts.
macros	object/array	User macros to be created for the given hosts.
templates	object/array	Templates to link to the given hosts. 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": "{$TEST1}",
                "value": "MACROTEST1"
            },
            {
                "macro": "{$TEST2}",
                "value": "MACROTEST2",
                "description": "Test description"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

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.

Parameters

(object) Parameters containing the IDs of the hosts to update and the objects that should be removed.

Parameter	Type	Description
hostids (required)	string/array	IDs of the hosts to be updated.
groupids	string/array	Host groups to remove the given hosts from.
interfaces	object/array	Host interfaces to remove from the given hosts. The host interface object must have the ip, dns and port properties defined.
macros	string/array	User macros to delete from the given hosts.
templateids	string/array	Templates to unlink from the given hosts.
templateids_clear	string/array	Templates to unlink and clear from the given hosts.

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
- User macro
- Host interface

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.

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.

Parameter	Type	Description
hosts (required)	object/array	Hosts to be updated. The hosts must have the `hostid` property defined.
groups	object/array	Host groups to replace the current host groups the hosts belong to. The host groups must have the `groupid` property defined.
interfaces	object/array	Host interfaces to replace the current host interfaces on the given hosts.
inventory	object	Host inventory properties. Host inventory mode cannot be updated using the `inventory` parameter, use `inventory_mode` instead.
macros	object/array	User macros to replace the current user macros on the given hosts.
templates	object/array	Templates to replace the currently linked templates on the given hosts. The templates must have the `templateid` property defined.
templates_clear	object/array	Templates to unlink and clear from the given hosts. 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

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.

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.

Parameter	Type	Description
groups	object/array	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.
interfaces	object/array	Host interfaces to replace the current host interfaces. All interfaces that are not listed in the request will be removed.
tags	object/array	Host tags to replace the current host tags. All tags that are not listed in the request will be removed.
inventory	object	Host inventory properties.
macros	object/array	User macros to replace the current user macros. All macros that are not listed in the request will be removed.
templates	object/array	Templates to replace the currently linked templates. All templates that are not listed in the request will be only unlinked. The templates must have the `templateid` property defined.
templates_clear	object/array	Templates to unlink and clear from the host. The templates must have the `templateid` property defined.

Note:
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
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10126"
        ]
    },
    "id": 1
}
```

Unlinking templates

Unlink and clear two templates from host.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10126",
        "templates_clear": [
            {
                "templateid": "10124"
            },
            {
                "templateid": "10125"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10126"
        ]
    },
```

```
    "id": 1
}
```

Updating host macros

Replace all host macros with two new ones.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10126",
        "macros": [
            {
                "macro": "{$PASS}",
                "value": "password"
            },
            {
                "macro": "{$DISC}",
                "value": "sda",
                "description": "Updated description"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10126"
        ]
    },
    "id": 1
}
```

Updating host inventory

Change inventory mode and add location

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10387",
        "inventory_mode": 0,
        "inventory": {
            "location": "Latvia, Riga"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
```

```
            "10387"
        ]
    },
    "id": 1
}
```

Updating host tags

Replace all host tags with a new one.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10387",
        "tags": {
            "tag": "OS",
            "value": "CentOS 7"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10387"
        ]
    },
    "id": 1
}
```

Updating host encryption

Update the host ”10590” to use PSK encryption only for connections from host to Zabbix server, and change the PSK identity and PSK key. Note that the host has to be pre-configured to use PSK.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10590",
        "tls_connect": 1,
        "tls_accept": 2,
        "tls_psk_identity": "PSK 002",
        "tls_psk": "e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10590"
        ]
    },
```

```
    "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.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.

Property	Type	Description
groupid	string	*(readonly)* ID of the host group.
name (required)	string	Name of the host group.
flags	integer	*(readonly)* Origin of the host group. Possible values: 0 - a plain host group; 4 - a discovered host group.
internal	integer	*(readonly)* Whether the group is used internally by the system. An internal group cannot be deleted. Possible values: 0 - *(default)* not internal; 1 - internal.

Note that for some methods (update, delete) the required/optional parameter combination is different.

hostgroup.create

Description

`object hostgroup.create(object/array hostGroups)`

This method allows to create new host groups.

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:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.create",
    "params": {
        "name": "Linux servers"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107819"
        ]
    },
    "id": 1
}
```

Source

CHostGroup::create() in *ui/include/classes/api/services/CHostGroup.php*.

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.

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:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.delete",
    "params": [
        "107824",
        "107825"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107824",
            "107825"
        ]
    },
    "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.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
graphids	string/array	Return only host groups that contain hosts or templates with the given graphs.
groupids	string/array	Return only host groups with the given host group IDs.
hostids	string/array	Return only host groups that contain the given hosts.
maintenanceids	string/array	Return only host groups that are affected by the given maintenances.
monitored_hosts	flag	Return only host groups that contain monitored hosts.
real_hosts	flag	Return only host groups that contain hosts.
templated_hosts	flag	Return only host groups that contain templates.
templateids	string/array	Return only host groups that contain the given templates.
triggerids	string/array	Return only host groups that contain hosts or templates with the given triggers.
with_applications	flag	Return only host groups that contain hosts with applications.
with_graphs	flag	Return only host groups that contain hosts with graphs.
with_graph_prototypes	flag	Return only host groups that contain hosts with graph prototypes.
with_hosts_and_templates	flag	Return only host groups that contain hosts *or* templates.

Parameter	Type	Description
with_httptests	flag	Return only host groups that contain hosts with web checks. Overrides the `with_monitored_httptests` parameter.
with_items	flag	Return only host groups that contain hosts or templates with items. Overrides the `with_monitored_items` and `with_simple_graph_items` parameters.
with_item_prototypes	flag	Return only host groups that contain hosts with item prototypes. Overrides the `with_simple_graph_item_prototypes` parameter.
with_simple_graph_item_prototypes	flag	Return only host groups that contain hosts with item prototypes, which are enabled for creation and have numeric type of information.
with_monitored_httptests	flag	Return only host groups that contain hosts with enabled web checks.
with_monitored_items	flag	Return only host groups that contain hosts or templates with enabled items. Overrides the `with_simple_graph_items` parameter.
with_monitored_triggers	flag	Return only host groups that contain hosts with enabled triggers. All of the items used in the trigger must also be enabled.
with_simple_graph_items	flag	Return only host groups that contain hosts with numeric items.
with_triggers	flag	Return only host groups that contain hosts with triggers. Overrides the `with_monitored_triggers` parameter.
selectDiscoveryRule	query	Return a discoveryRule property with the LLD rule that created the host group.
selectGroupDiscovery	query	Return a `groupDiscovery` property with the host group discovery object. The host group discovery object links a discovered host group to a host group prototype and has the following properties: `groupid` - `(string)` ID of the discovered host group; `lastcheck` - `(timestamp)` time when the host group was last discovered; `name` - `(string)` name of the host group prototype; `parent_group_prototypeid` - `(string)` ID of the host group prototype from which the host group has been created; `ts_delete` - `(timestamp)` time when a host group that is no longer discovered will be deleted.
selectHosts	query	Return a hosts property with the hosts that belong to the host group. Supports `count`.
selectTemplates	query	Return a templates property with the templates that belong to the host group. Supports `count`.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectHosts` - results will be sorted by `host`; `selectTemplates` - results will be sorted by `host`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `groupid`, `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	

Parameter	Type	Description
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 data by name

Retrieve all data about two host groups named ”Zabbix servers” and ”Linux servers”.

Request:

```
{
    "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
- Template

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.

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.

Parameter	Type	Description
groups (required)	object/array	Host groups to be updated. The host groups must have the `groupid` property defined.
hosts	object/array	Hosts to add to all host groups. The hosts must have the `hostid` property defined.
templates	object/array	Templates to add to all host groups. The templates must have the `templateid` property defined.

Return values

`(object)` 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
- Template

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.

Parameters

`(object)` Parameters containing the IDs of the host groups to update and the objects that should be removed.

Parameter	Type	Description
groupids (required)	string/array	IDs of the host groups to be updated.
hostids	string/array	Hosts to remove from all host groups.
templateids	string/array	Templates to remove from all host groups.

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:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.massremove",
    "params": {
        "groupids": [
            "5",
            "6"
        ],
        "hostids": [
            "30050",
            "30001"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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 simultaneously replace or remove related objects for multiple host groups.

Parameters

`(object)` Parameters containing the IDs of the host groups to update and the objects that should be updated.

Parameter	Type	Description
groups (required)	object/array	Host groups to be updated. The host groups must have the `groupid` property defined.
hosts	object/array	Hosts to replace the current hosts on the given host groups. The hosts must have the `hostid` property defined.
templates	object/array	Templates to replace the current templates on the given host groups. The templates must have the `templateid` property defined.

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 the host group with ID.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.massupdate",
    "params": {
        "groups": [
            {
                "groupid": "6"
            }
        ],
        "hosts": [
            {
                "hostid": "30050"
            }
        ]
    },
    "auth": "f223adf833b2bf2ff38574a67bba6372",
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "6",
        ]
    },
    "id": 1
}
```

See also

- hostgroup.update
- hostgroup.massadd
- Host
- Template

Source

CHostGroup::massUpdate() in *ui/include/classes/api/services/CHostGroup.php*.

hostgroup.update

Description

`object hostgroup.update(object/array hostGroups)`

This method allows to update existing hosts groups.

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:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.update",
    "params": {
        "groupid": "7",
        "name": "Linux hosts"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "7"
```

```
        ]
    },
    "id": 1
}
```

Source

CHostGroup::update() in *ui/include/classes/api/services/CHostGroup.php*.

Host interface

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

The following objects are directly related to the `hostinterface` API.

Host interface

The host interface object has the following properties.

Attention:
Note that both IP and DNS are required. If you do not want to use DNS, set it to an empty string.

Property	Type	Description
interfaceid	string	*(readonly)* ID of the interface.
dns (required)	string	DNS name used by the interface. Can be empty if the connection is made via IP.
hostid (required)	string	ID of the host the interface belongs to.
ip (required)	string	IP address used by the interface. Can be empty if the connection is made via DNS.
main (required)	integer	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.
port (required)	string	Port number used by the interface. Can contain user macros.

Property	Type	Description
type (required)	integer	Interface type. Possible values are: 1 - agent; 2 - SNMP; 3 - IPMI; 4 - JMX.
useip (required)	integer	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.
details	array	Additional object for interface. **Required** if interface 'type' is SNMP.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Details tag

The details object has the following properties.

Property	Type	Description
version (required)	integer	SNMP interface version. Possible values are: 1 - SNMPv1; 2 - SNMPv2c; 3 - SNMPv3
bulk	integer	Whether to use bulk SNMP requests. Possible values are: 0 - don't use bulk requests; 1 - (default) - use bulk requests.
community	string	SNMP community (required). Used only by SNMPv1 and SNMPv2 interfaces.
securityname	string	SNMPv3 security name. Used only by SNMPv3 interfaces.
securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 interfaces. Possible values are: 0 - (default) - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
authpassphrase	string	SNMPv3 authentication passphrase. Used only by SNMPv3 interfaces.
privpassphrase	string	SNMPv3 privacy passphrase. Used only by SNMPv3 interfaces.
authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 interfaces. Possible values are: 0 - (default) - MD5; 1 - SHA.
privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 interfaces. Possible values are: 0 - (default) - DES; 1 - AES.
contextname	string	SNMPv3 context name. Used only by SNMPv3 interfaces.

hostinterface.create

Description

`object hostinterface.create(object/array hostInterfaces)`

This method allows to create new host interfaces.

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"
        ]
    },
    "id": 1
}
```

See also

- hostinterface.massadd
- host.massadd

Source

CHostInterface::create() in *ui/include/classes/api/services/CHostInterface.php*.

hostinterface.delete

Description

`object hostinterface.delete(array hostInterfaceIds)`

This method allows to delete host interfaces.

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

Description

`integer/array hostinterface.get(object parameters)`

The method allows to retrieve host interfaces according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
hostids	string/array	Return only host interfaces used by the given hosts.
interfaceids	string/array	Return only host interfaces with the given IDs.
itemids	string/array	Return only host interfaces used by the given items.
triggerids	string/array	Return only host interfaces used by items in the given triggers.
selectItems	query	Return an items property with the items that use the interface. Supports `count`.
selectHosts	query	Return a hosts property with an array of hosts that use the interface.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectItems`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `interfaceid`, `dns`, `ip`.
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

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": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "interfaceid": "30050",
            "hostid": "30057",
            "main": "1",
            "type": "1",
            "useip": "1",
            "ip": "127.0.0.1",
            "dns": "",
            "port": "10050",
            "details": []
        },
        {
            "interfaceid": "30067",
            "hostid": "30057",
            "main": "0",
            "type": "1",
            "useip": "0",
            "ip": "",
            "dns": "localhost",
            "port": "10050",
            "details": []
        },
        {
            "interfaceid": "30068",
            "hostid": "30057",
            "main": "1",
            "type": "2",
            "useip": "1",
            "ip": "127.0.0.1",
            "dns": "",
            "port": "161",
            "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

Description

`object hostinterface.massadd(object parameters)`

This method allows to simultaneously add host interfaces to multiple hosts.

Parameters

`(object)` Parameters containing the host interfaces to be created on the given hosts.

The method accepts the following parameters.

Parameter	Type	Description
hosts (required)	object/array	Hosts to be updated. The hosts must have the `hostid` property defined.
interfaces (required)	object/array	Host interfaces to create on the given hosts.

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:

```
{
    "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:

```
{
    "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

Description

`object hostinterface.massremove(object parameters)`

This method allows to remove host interfaces from the given hosts.

Parameters

`(object)` Parameters containing the IDs of the hosts to be updated and the interfaces to be removed.

Parameter	Type	Description
hostids (required)	string/array	IDs of the hosts to be updated.
interfaces (required)	object/array	Host interfaces to remove from the given hosts. The host interface object must have the ip, dns and port properties defined

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:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.massremove",
    "params": {
        "hostids": [
            "30050",
            "30052"
        ],
        "interfaces": {
            "dns": "",
            "ip": "127.0.0.1",
            "port": "161"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30069",
            "30070"
        ]
    },
    "id": 1
}
```

See also

- hostinterface.delete
- host.massremove

Source

CHostInterface::massRemove() in *ui/include/classes/api/services/CHostInterface.php*.

hostinterface.replacehostinterfaces

Description

`object hostinterface.replacehostinterfaces(object parameters)`

This method allows to replace all host interfaces on a given host.

Parameters

`(object)` Parameters containing the ID of the host to be updated and the new host interfaces.

Parameter	Type	Description
hostid (required)	string	ID of the host to be updated.
interfaces (required)	object/array	Host interfaces to replace the current host interfaces with.

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:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.replacehostinterfaces",
    "params": {
        "hostid": "30052",
        "interfaces": {
            "dns": "",
            "ip": "127.0.0.1",
            "main": 1,
            "port": "10050",
            "type": 1,
            "useip": 1
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30081"
        ]
    },
    "id": 1
}
```

See also

- host.update
- host.massupdate

Source

CHostInterface::replaceHostInterfaces() in *ui/include/classes/api/services/CHostInterface.php*.

hostinterface.update

Description

`object hostinterface.update(object/array hostInterfaces)`

This method allows to update existing host interfaces.

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:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.update",
    "params": {
        "interfaceid": "30048",
        "port": "30050"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30048"
        ]
    },
    "id": 1
}
```

Source

CHostInterface::update() in *ui/include/classes/api/services/CHostInterface.php*.

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.

Property	Type	Description
hostid	string	*(readonly)* ID of the host prototype.
host (required)	string	Technical name of the host prototype.
name	string	Visible name of the host prototype. Default: `host` property value.
status	integer	Status of the host prototype. Possible values are: 0 - *(default)* monitored host; 1 - unmonitored host.
inventory_mode	integer	Host inventory population mode. Possible values are: -1 - *(default)* disabled; 0 - manual; 1 - automatic.
templateid	string	*(readonly)* ID of the parent template host prototype.
discover	integer	Host prototype discovery status. Possible values: 0 - *(default)* new hosts will be discovered; 1 - new hosts will not be discovered and existing hosts will be marked as lost.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Group link

The group link object links a host prototype with a host group and has the following properties.

Property	Type	Description
group_prototypeid	string	*(readonly)* ID of the group link.
groupid (required)	string	ID of the host group.
hostid	string	*(readonly)* ID of the host prototype
templateid	string	*(readonly)* ID of the parent template group link.

Group prototype

The group prototype object defines a group that will be created for a discovered host and has the following properties.

Property	Type	Description
group_prototypeid	string	*(readonly)* ID of the group prototype.
name (required)	string	Name of the group prototype.
hostid	string	*(readonly)* ID of the host prototype
templateid	string	*(readonly)* ID of the parent template group prototype.

hostprototype.create

Description

`object hostprototype.create(object/array hostPrototypes)`

This method allows to create new host prototypes.

Parameters

`(object/array)` Host prototypes to create.

Additionally to the standard host prototype properties, the method accepts the following parameters.

Parameter	Type	Description
groupLinks (required)	array	Group links to be created for the host prototype.
ruleid (required)	string	ID of the LLD rule that the host prototype belongs to.
groupPrototypes	array	Group prototypes to be created for the host prototype.
macros	object/array	User macros to be created for the host prototype.
templates	object/array	Templates to be linked to the host prototype. 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}". Link it to host group "2".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.create",
    "params": {
        "host": "{#VM.NAME}",
        "ruleid": "23542",
        "groupLinks": [
            {
```

```
                "groupid": "2"
            }
        ],
        "groupPrototypes": [
            {
                "name": "{#HV.NAME}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10103"
        ]
    },
    "id": 1
}
```

See also

- Group link
- Group prototype
- User macro

Source

CHostPrototype::create() in *ui/include/classes/api/services/CHostPrototype.php*.

hostprototype.delete

Description

`object hostprototype.delete(array hostPrototypeIds)`

This method allows to delete host prototypes.

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::delete() 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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
hostids	string/array	Return only host prototypes with the given IDs.
discoveryids	string/array	Return only host prototype that belong to the given LLD rules.
inherited	boolean	If set to `true` return only items inherited from a template.
selectDiscoveryRule	query	Return a discoveryRule property with the LLD rule that the host prototype belongs to.
selectGroupLinks	query	Return a groupLinks property with the group links of the host prototype.
selectGroupPrototypes	query	Return a groupPrototypes property with the group prototypes of the host prototype.
selectMacros	query	Return a macros property with host prototype macros.
selectParentHost	query	Return a parentHost property with the host that the host prototype belongs to.
selectTemplates	query	Return a templates property with the templates linked to the host prototype. Supports `count`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `hostid`, `host`, `name` and `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail on the Generic Zabbix API information 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 host prototypes from an LLD rule

Retrieve all host prototypes and their group links and group prototypes from an LLD rule.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.get",
    "params": {
        "output": "extend",
        "selectGroupLinks": "extend",
        "selectGroupPrototypes": "extend",
        "discoveryids": "23554"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10092",
            "host": "{#HV.UUID}",
            "status": "0",
            "name": "{#HV.NAME}",
            "templateid": "0",
            "discover": "0",
            "groupLinks": [
                {
                    "group_prototypeid": "4",
                    "hostid": "10092",
                    "groupid": "7",
                    "templateid": "0"
                }
            ],
            "groupPrototypes": [
                {
                    "group_prototypeid": "7",
                    "hostid": "10092",
                    "name": "{#CLUSTER.NAME}",
                    "templateid": "0"
                }
            ]
        }
    ],
    "id": 1
}
```

See also

- 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.

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.

Parameter	Type	Description
groupLinks	array	Group links to replace the current group links on the host prototype.
groupPrototypes	array	Group prototypes to replace the existing group prototypes on the host prototype.
macros	object/array	User macros to replace the current user macros. All macros that are not listed in the request will be removed.
templates	object/array	Templates to replace the currently linked templates. The templates must have the `templateid` property defined.

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:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.update",
    "params": {
        "hostid": "10092",
        "status": 1
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10092"
        ]
    },
    "id": 1
}
```

See also

- Group link
- Group prototype
- User macro

Source

CHostPrototype::update() in *ui/include/classes/api/services/CHostPrototype.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.

Property	Type	Description
iconmapid	string	*(readonly)* ID of the icon map.
default_iconid (required)	string	ID of the default icon.
name (required)	string	Name of the icon map.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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.

Property	Type	Description
iconmappingid	string	*(readonly)* ID of the icon map.
iconid (required)	string	ID of the icon used by the icon mapping.
expression (required)	string	Expression to match the inventory field against.
inventory_link (required)	integer	ID of the host inventory field. Refer to the host inventory object for a list of supported inventory fields.
iconmapid	string	*(readonly)* ID of the icon map that the icon mapping belongs to.
sortorder	integer	*(readonly)* Position of the icon mapping in the icon map.

iconmap.create

Description

`object iconmap.create(object/array iconMaps)`

This method allows to create new icon maps.

Parameters

`(object/array)` Icon maps to create.

Additionally to the standard icon map properties, the method accepts the following parameters.

Parameter	Type	Description
mappings (required)	array	Icon mappings to be created for the icon map.

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:

```
{
    "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": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "iconmapids": [
            "2"
        ]
    },
    "id": 1
}
```

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.

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:

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.delete",
    "params": [
        "2",
        "5"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "iconmapids": [
            "2",
            "5"
        ]
    },
    "id": 1
}
```

Source

CIconMap::delete() in *ui/include/classes/api/services/CIconMap.php*.

iconmap.get

Description

`integer/array iconmap.get(object parameters)`

The method allows to retrieve icon maps according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
iconmapids	string/array	Return only icon maps with the given IDs.
sysmapids	string/array	Return only icon maps that are used in the given maps.
selectMappings	query	Return a mappings property with the icon mappings used.

Parameter	Type	Description
sortfield	string/array	Sort the result by the given properties. Possible values are: `iconmapid` and `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
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

Retrieve an icon map

Retrieve all data about icon map "3".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.get",
    "params": {
        "iconmapids": "3",
        "output": "extend",
        "selectMappings": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

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

`object iconmap.update(object/array iconMaps)`

This method allows to update existing icon maps.

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.

Parameter	Type	Description
mappings	array	Icon mappings to replace the existing icon mappings.

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:


```
{
    "jsonrpc": "2.0",
    "method": "iconmap.update",
    "params": {
        "iconmapid": "1",
        "name": "OS icons"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```


Response:


```
{
    "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.

Property	Type	Description
imageid	string	*(readonly)* ID of the image.
name (required)	string	Name of the image.
imagetype	integer	Type of image. Possible values: 1 - *(default)* icon; 2 - background image.

Note that for some methods (update, delete) the required/optional parameter combination is different.

image.create

Description

`object image.create(object/array images)`

This method allows to create new images.

Parameters

`(object/array)` Images to create.

Additionally to the standard image properties, the method accepts the following parameters.

Parameter	Type	Description
name (required)	string	Name of the image.
imagetype (required)	integer	Type of image. Possible values: 1 - *(default)* icon; 2 - background image.
image (required)	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:

```
{
    "jsonrpc": "2.0",
    "method": "image.create",
    "params": {
        "imagetype": 1,
        "name": "Cloud_(24)",
        "image": "iVBORw0KGgoAAAANSUhEUgAAABgAAAANCAYAAACzbK7QAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAACmAAAApgB
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "imageids": [
            "188"
        ]
    },
    "id": 1
}
```

Source

CImage::create() in *ui/include/classes/api/services/CImage.php*.

image.delete

Description

`object image.delete(array imageIds)`

This method allows to delete images.

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
}
```

Source

CImage::delete() in *ui/include/classes/api/services/CImage.php*.

image.get

Description

`integer/array image.get(object parameters)`

The method allows to retrieve images according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
imageids	string/array	Return only images with the given IDs.
sysmapids	string/array	Return images that are used on the given maps.
select_image	flag	Return an `image` property with the Base64 encoded image.
sortfield	string/array	Sort the result by the given properties. Possible values are: `imageid` and `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	

Parameter	Type	Description
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

Retrieve an image

Retrieve all data for image with ID "2".

Request:

```
{
    "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": "iVBORw0KGgoAAAANSUhEUgAAABgAAAANCAYAAACzbK7QAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAACmAAA
        }
    ],
    "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.

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.

Parameter	Type	Description
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 updated images under the imageids property.

Examples

Rename image

Rename image to "Cloud icon".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "image.update",
    "params": {
        "imageid": "2",
        "name": "Cloud icon"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "imageids": [
            "2"
        ]
    },
    "id": 1
}
```

Source

CImage::update() in *ui/include/classes/api/services/CImage.php*.

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

Note:
Web items cannot be directly created, updated or deleted via the Zabbix API.

The item object has the following properties.

Property	Type	Description
itemid	string	*(readonly)* ID of the item.
delay (required)	string	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. `{$FLEX_INTERVAL}/{$FLEX_PERIOD}`). Optional for Zabbix trapper or dependent items.
hostid (required)	string	ID of the host or template that the item belongs to. For update operations this field is *readonly*.
interfaceid (required)	string	ID of the item's host interface. Used only for host items. Not required for Zabbix agent (active), Zabbix internal, Zabbix trapper, Zabbix aggregate, calculated, dependent and database monitor items.
key_ (required)	string	Item key.
name (required)	string	Name of the item.
type (required)	integer	Type of the item. Possible values: 0 - Zabbix agent; 2 - Zabbix trapper; 3 - Simple check; 5 - Zabbix internal; 7 - Zabbix agent (active); 8 - Zabbix aggregate; 9 - Web item; 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
url (required)	string	URL string, required only for HTTP agent item type. Supports user macros, {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}.
value_type (required)	integer	Type of information of the item. Possible values: 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text.

Property	Type	Description
allow_traps	integer	HTTP agent item 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.
authtype	integer	Used only by SSH agent items or HTTP agent items. SSH agent authentication method possible values: 0 - *(default)* password; 1 - public key. HTTP agent authentication method possible values: 0 - *(default)* none 1 - basic 2 - NTLM 3 - Kerberos
description	string	Description of the item.
error	string	*(readonly)* Error text if there are problems updating the item value.
flags	integer	*(readonly)* Origin of the item. Possible values: 0 - a plain item; 4 - a discovered item.
follow_redirects	integer	HTTP agent item field. Follow response redirects while pooling data. 0 - Do not follow redirects. 1 - *(default)* Follow redirects.
headers	object	HTTP agent item field. Object with HTTP(S) request headers, where header name is used as key and header value as value. Example: { "User-Agent": "Zabbix" }
history	string	A time unit of how long the history data should be stored. Also accepts user macro. Default: 90d.
http_proxy	string	HTTP agent item field. HTTP(S) proxy connection string.
inventory_link	integer	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. Default: 0.
ipmi_sensor	string	IPMI sensor. Used only by IPMI items.
jmx_endpoint	string	JMX agent custom connection string. Default value: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi
lastclock	timestamp	*(readonly)* Time when the item value was last updated. This property will only return a value for the period configured in ZBX_HISTORY_PERIOD.
lastns	integer	*(readonly)* Nanoseconds when the item value was last updated. This property will only return a value for the period configured in ZBX_HISTORY_PERIOD.
lastvalue	string	*(readonly)* Last value of the item. This property will only return a value for the period configured in ZBX_HISTORY_PERIOD.
logtimefmt	string	Format of the time in log entries. Used only by log items.

Property	Type	Description
master_itemid	integer	Master item ID. Recursion up to 3 dependent items and maximum count of dependent items equal to 29999 are allowed. Required by dependent items.
output_format	integer	HTTP agent item field. Should response be converted to JSON. 0 - *(default)* Store raw. 1 - Convert to JSON.
params	string	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.
password	string	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.
post_type	integer	HTTP agent item field. Type of post data body stored in posts property. 0 - *(default)* Raw data. 2 - JSON data. 3 - XML data.
posts	string	HTTP agent item field. HTTP(S) request body data. Used with post_type.
prevvalue	string	*(readonly)* Previous value of the item. This property will only return a value for the period configured in ZBX_HISTORY_PERIOD.
privatekey	string	Name of the private key file.
publickey	string	Name of the public key file.
query_fields	array	HTTP agent item field. Query parameters. Array of objects with ‘key’:’value’ pairs, where value can be empty string.
request_method	integer	HTTP agent item field. Type of request method. 0 - *(default)* GET 1 - POST 2 - PUT 3 - HEAD
retrieve_mode	integer	HTTP agent item field. What part of response should be stored. 0 - *(default)* Body. 1 - Headers. 2 - Both body and headers will be stored. For request_method HEAD only 1 is allowed value.
snmp_oid	string	SNMP OID.
ssl_cert_file	string	HTTP agent item field. Public SSL Key file path.
ssl_key_file	string	HTTP agent item field. Private SSL Key file path.
ssl_key_password	string	HTTP agent item field. Password for SSL Key file.
state	integer	*(readonly)* State of the item. Possible values: 0 - *(default)* normal; 1 - not supported.
status	integer	Status of the item. Possible values: 0 - *(default)* enabled item; 1 - disabled item.

Property	Type	Description
status_codes	string	HTTP agent item field. Ranges of required HTTP status codes separated by commas. Also supports user macros as part of comma separated list. Example: 200,200-{$M},{$M},200-400
templateid	string	(readonly) ID of the parent template item. *Hint*: Use the `hostid` property to specify the template that the item belongs to.
timeout	string	HTTP agent item field. Item data polling request timeout. Support user macros. default: 3s maximum value: 60s
trapper_hosts	string	Allowed hosts. Used by trapper items or HTTP agent items.
trends	string	A time unit of how long the trends data should be stored. Also accepts user macro. Default: 365d.
units	string	Value units.
username	string	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.
valuemapid	string	ID of the associated value map.
verify_host	integer	HTTP agent item field. Whether to validate that the host name for the connection matches the one in the host's certificate. 0 - *(default)* Do not validate. 1 - Validate.
verify_peer	integer	HTTP agent item field. Whether to validate that the host's certificate is authentic. 0 - *(default)* Do not validate. 1 - Validate.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Item preprocessing

The item preprocessing object has the following properties.

Property	Type	Description
type (required)	integer	The preprocessing option type. Possible values: 1 - Custom multiplier; 2 - Right trim; 3 - Left trim; 4 - Trim; 5 - Regular expression matching; 6 - Boolean to decimal; 7 - Octal to decimal; 8 - Hexadecimal to decimal; 9 - Simple change; 10 - Change per second; 11 - XML XPath; 12 - JSONPath; 13 - In range; 14 - Matches regular expression; 15 - Does not match regular expression; 16 - Check for error in JSON; 17 - Check for error in XML; 18 - Check for error using regular expression; 19 - Discard unchanged; 20 - Discard unchanged with heartbeat; 21 - JavaScript; 22 - Prometheus pattern; 23 - Prometheus to JSON; 24 - CSV to JSON; 25 - Replace.
params (required)	string	Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n) character.
error_handler (required)	integer	Action type used in case of preprocessing step failure. Possible values: 0 - Error message is set by Zabbix server; 1 - Discard value; 2 - Set custom value; 3 - Set custom error message.
error_handler_params (required)	string	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.

The following parameters and error handlers are supported for each preprocessing type.

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
1	Custom multiplier	number[1, 6]			0, 1, 2, 3
2	Right trim	list of characters[2]			
3	Left trim	list of characters[2]			
4	Trim	list of characters[2]			
5	Regular expression	pattern[3]	output[2]		0, 1, 2, 3

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
6	Boolean to decimal				0, 1, 2, 3
7	Octal to decimal				0, 1, 2, 3
8	Hexadecimal to decimal				0, 1, 2, 3
9	Simple change				0, 1, 2, 3
10	Change per second				0, 1, 2, 3
11	XML XPath	path[4]			0, 1, 2, 3
12	JSONPath	path[4]			0, 1, 2, 3
13	In range	min[1, 6]	max[1, 6]		0, 1, 2, 3
14	Matches regular expression	pattern[3]			0, 1, 2, 3
15	Does not match regular expression	pattern[3]			0, 1, 2, 3
16	Check for error in JSON	path[4]			0, 1, 2, 3
17	Check for error in XML	path[4]			0, 1, 2, 3
18	Check for error using regular expression	pattern[3]	output[2]		0, 1, 2, 3
19	Discard unchanged				

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
20	Discard unchanged with heartbeat	seconds[5, 6]			
21	JavaScript	script[2]			
22	Prometheus pattern	pattern[6, 7]	output[6, 8]		0, 1, 2, 3
23	Prometheus to JSON	pattern[6, 7]			0, 1, 2, 3
24	CSV to JSON	character[2]	character[2]	0,1	0, 1, 2, 3
25	Replace	search string[2]	replacement[2]		

[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>`.

item.create

Description

`object item.create(object/array items)`

This method allows to create new items.

> **Note:**
> Web items cannot be created via the Zabbix API.

Parameters

`(object/array)` Items to create.

Additionally to the standard item properties, the method accepts the following parameters.

Parameter	Type	Description
applications	array	IDs of the applications to add the item to.
preprocessing	array	Item preprocessing options.

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 to monitor free disk space on host with ID ”30074” and add it to two applications.

Request:

```
{
    "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",
        "applications": [
            "609",
            "610"
        ],
        "delay": "30s"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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:

```
{
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24759"
        ]
    },
    "id": 1
}
```

Creating an item with preprocessing

Create an item using custom multiplier.

Request:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44210"
        ]
    },
    "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:

```
{
    "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
}
```

Create HTTP agent item

Create POST request method item with JSON response preprocessing.

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"
            }
        ]
    },
    "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.

> **Note:**
> Web items cannot be deleted via the Zabbix API.

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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
itemids	string/array	Return only items with the given IDs.
groupids	string/array	Return only items that belong to the hosts from the given groups.
templateids	string/array	Return only items that belong to the given templates.
hostids	string/array	Return only items that belong to the given hosts.
proxyids	string/array	Return only items that are monitored by the given proxies.
interfaceids	string/array	Return only items that use the given host interfaces.
graphids	string/array	Return only items that are used in the given graphs.
triggerids	string/array	Return only items that are used in the given triggers.
applicationids	string/array	Return only items that belong to the given applications.
webitems	flag	Include web items in the result.
inherited	boolean	If set to `true` return only items inherited from a template.
templated	boolean	If set to `true` return only items that belong to templates.
monitored	boolean	If set to `true` return only enabled items that belong to monitored hosts.
group	string	Return only items that belong to a group with the given name.
host	string	Return only items that belong to a host with the given name.
application	string	Return only items that belong to an application with the given name.
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. Supports `count`.
selectGraphs	query	Return a graphs property with the graphs that contain the item. Supports `count`.
selectApplications	query	Return an applications property with the applications that the item belongs to.
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.

Parameter	Type	Description
selectPreprocessing	query	Return a preprocessing property with item preprocessing options. It has the following properties: `type` - (`string`) The preprocessing option type: 1 - Custom multiplier; 2 - Right trim; 3 - Left trim; 4 - Trim; 5 - Regular expression matching; 6 - Boolean to decimal; 7 - Octal to decimal; 8 - Hexadecimal to decimal; 9 - Simple change; 10 - Change per second; 11 - XML XPath; 12 - JSONPath; 13 - In range; 14 - Matches regular expression; 15 - Does not match regular expression; 16 - Check for error in JSON; 17 - Check for error in XML; 18 - Check for error using regular expression; 19 - Discard unchanged; 20 - Discard unchanged with heartbeat; 21 - JavaScript; 22 - Prometheus pattern; 23 - Prometheus to JSON; 24 - CSV to JSON; 25 - Replace. `params` - (`string`) Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n)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.
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 item belongs to.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectGraphs` - results will be sorted by `name`; `selectTriggers` - results will be sorted by `description`.
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.
editable	boolean	
excludeSearch	boolean	
limit	integer	

Parameter	Type	Description
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

Finding items by key

Retrieve all items from host with ID "10084" that have the word "system" in the key and sort them by name.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "hostids": "10084",
        "search": {
            "key_": "system"
        },
        "sortfield": "name"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23298",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "Context switches per second",
            "key_": "system.cpu.switches",
            "delay": "1m",
            "history": "7d",
            "trends": "365d",
            "lastvalue": "2552",
            "lastclock": "1351090998",
            "prevvalue": "2641",
            "state": "0",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "sps",
            "error": "",
            "logtimefmt": "",
            "templateid": "22680",
            "valuemapid": "0",
```

```
        "params": "",
        "ipmi_sensor": "",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "lastns": "564054253",
        "flags": "0",
        "interfaceid": "1",
        "description": "",
        "inventory_link": "0",
        "lifetime": "0s",
        "evaltype": "0",
        "jmx_endpoint": "",
        "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"
    },
    {
        "itemid": "23299",
        "type": "0",
        "snmp_oid": "",
        "hostid": "10084",
        "name": "CPU $2 time",
        "key_": "system.cpu.util[,idle]",
        "delay": "1m",
        "history": "7d",
        "trends": "365d",
        "lastvalue": "86.031879",
        "lastclock": "1351090999",
        "prevvalue": "85.306944",
        "state": "0",
        "status": "0",
        "value_type": "0",
        "trapper_hosts": "",
        "units": "%",
        "error": "",
        "logtimefmt": "",
        "templateid": "17354",
        "valuemapid": "0",
        "params": "",
        "ipmi_sensor": "",
        "authtype": "0",
        "username": "",
        "password": "",
```

```
            "publickey": "",
            "privatekey": "",
            "lastns": "564256864",
            "flags": "0",
            "interfaceid": "1",
            "description": "The time the CPU has spent doing nothing.",
            "inventory_link": "0",
            "lifetime": "0s",
            "evaltype": "0",
            "jmx_endpoint": "",
            "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"
        },
        {
            "itemid": "23300",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "CPU $2 time",
            "key_": "system.cpu.util[,interrupt]",
            "history": "7d",
            "trends": "365d",
            "lastvalue": "0.008389",
            "lastclock": "1351091000",
            "prevvalue": "0.000000",
            "state": "0",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "%",
            "error": "",
            "logtimefmt": "",
            "templateid": "22671",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "lastns": "564661387",
            "flags": "0",
            "interfaceid": "1",
            "description": "The amount of time the CPU has been servicing hardware interrupts.",
```

```
            "inventory_link": "0",
            "lifetime": "0s",
            "evaltype": "0",
            "jmx_endpoint": "",
            "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"
        }
    ],
    "id": 1
}
```


Finding dependent items by key

Retrieve all dependent items from host with ID ”10116” that have the word ”apache” in the key.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "hostids": "10116",
        "search": {
            "key_": "apache"
        },
        "filter": {
            "type": "18"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "25550",
            "type": "18",
            "snmp_oid": "",
            "hostid": "10116",
            "name": "Days",
            "key_": "apache.status.uptime.days",
            "delay": "",
            "history": "90d",
```

```
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "error": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "state": "0",
            "evaltype": "0",
            "master_itemid": "25545",
            "jmx_endpoint": "",
            "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",
            "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": "",
```

```
            "units": "",
            "formula": "",
            "error": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "state": "0",
            "evaltype": "0",
            "master_itemid": "25545",
            "jmx_endpoint": "",
            "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",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        }
    ],
    "id": 1
}
```


Find HTTP agent item

Find HTTP agent item with post body type XML for specific host id.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "hostids": "10255",
        "filter": {
            "type": "19",
            "post_type": "3"
        }
```

```
    },
    "id": 3,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

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": "",
            "formula": "",
            "error": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "state": "0",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "localhost",
            "query_fields": [
                {
                    "mode": "xml"
                }
            ],
            "posts": "<body>\r\n<![CDATA[{$MACRO}<foo></bar>]]>\r\n</body>",
            "status_codes": "200",
            "follow_redirects": "0",
            "post_type": "3",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "1",
            "request_method": "3",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
```

```
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        }
    ],
    "id": 3
}
```

Retrieving items with preprocessing rules

Reatrieve all items and their preprocessing rules from host with ID "10254".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": ["itemid", "name", "key_"],
        "selectPreprocessing": "extend",
        "hostids": "10254"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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

- Application
- Discovery rule
- Graph
- Host
- Host interface
- Trigger

Source

CItem::get() in *ui/include/classes/api/services/CItem.php*.

item.update

Description

`object item.update(object/array items)`

This method allows to update existing items.

> **Note:**
> Web items cannot be updated via the Zabbix API.

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.

Parameter	Type	Description
applications	array	IDs of the applications to replace the current applications.
preprocessing	array	Item preprocessing options to replace the current preprocessing options.

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"
        ]
    },
    "id": 1
}
```

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
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23856"
        ]
    },
    "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.

Property	Type	Description
itemid	string	*(readonly)* ID of the item prototype.
delay (required)	string	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. `{$FLEX_INTERVAL}/{$FLEX_PERIOD}`). Optional for Zabbix trapper or dependent item.
hostid (required)	string	ID of the host that the item prototype belongs to. For update operations this field is *readonly*.

Property	Type	Description
ruleid (required)	string	ID of the LLD rule that the item belongs to. For update operations this field is *readonly*.
interfaceid (required)	string	ID of the item prototype's host interface. Used only for host item prototypes. Not required for Zabbix agent (active), Zabbix internal, Zabbix trapper, Zabbix aggregate, calculated, dependent and database monitor item prototypes.
key_ (required)	string	Item prototype key.
name (required)	string	Name of the item prototype.
type (required)	integer	Type of the item prototype. Possible values: 0 - Zabbix agent; 2 - Zabbix trapper; 3 - simple check; 5 - Zabbix internal; 7 - Zabbix agent (active); 8 - Zabbix aggregate; 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;
url (required)	string	URL string required only for HTTP agent item prototypes. Supports LLD macros, user macros, {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}.
value_type (required)	integer	Type of information of the item prototype. Possible values: 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text.
allow_traps	integer	HTTP agent item prototype 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.
authtype	integer	Used only by SSH agent item prototypes or HTTP agent item prototypes. SSH agent authentication method possible values: 0 - *(default)* password; 1 - public key. HTTP agent authentication method possible values: 0 - *(default)* none 1 - basic 2 - NTLM 3 - Kerberos

Property	Type	Description
description	string	Description of the item prototype.
follow_redirects	integer	HTTP agent item prototype field. Follow response redirects while pooling data. 0 - Do not follow redirects. 1 - *(default)* Follow redirects.
headers	object	HTTP agent item prototype field. Object with HTTP(S) request headers, where header name is used as key and header value as value. Example: { "User-Agent": "Zabbix" }
history	string	A time unit of how long the history data should be stored. Also accepts user macro and LLD macro. Default: 90d.
http_proxy	string	HTTP agent item prototype field. HTTP(S) proxy connection string.
ipmi_sensor	string	IPMI sensor. Used only by IPMI item prototypes.
jmx_endpoint	string	JMX agent custom connection string. Default value: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi
logtimefmt	string	Format of the time in log entries. Used only by log item prototypes.
master_itemid	integer	Master item ID. Recursion up to 3 dependent items and item prototypes and maximum count of dependent items and item prototypes equal to 29999 are allowed. Required by Dependent items.
output_format	integer	HTTP agent item prototype field. Should response be converted to JSON. 0 - *(default)* Store raw. 1 - Convert to JSON.
params	string	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.
password	string	Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent item prototypes.
post_type	integer	HTTP agent item prototype field. Type of post data body stored in posts property. 0 - *(default)* Raw data. 2 - JSON data. 3 - XML data.
posts	string	HTTP agent item prototype field. HTTP(S) request body data. Used with post_type.
privatekey	string	Name of the private key file.
publickey	string	Name of the public key file.
query_fields	array	HTTP agent item prototype field. Query parameters. Array of objects with 'key':'value' pairs, where value can be empty string.
request_method	integer	HTTP agent item prototype field. Type of request method. 0 - *(default)* GET 1 - POST 2 - PUT 3 - HEAD

Property	Type	Description
retrieve_mode	integer	HTTP agent item prototype field. What part of response should be stored. 0 - *(default)* Body. 1 - Headers. 2 - Both body and headers will be stored. For request_method HEAD only 1 is allowed value.
snmp_oid	string	SNMP OID.
ssl_cert_file	string	HTTP agent item prototype field. Public SSL Key file path.
ssl_key_file	string	HTTP agent item prototype field. Private SSL Key file path.
ssl_key_password	string	HTTP agent item prototype field. Password for SSL Key file.
status	integer	Status of the item prototype. Possible values: 0 - *(default)* enabled item prototype; 1 - disabled item prototype; 3 - unsupported item prototype.
status_codes	string	HTTP agent item prototype field. Ranges of required HTTP status codes separated by commas. Also supports user macros or LLD macros as part of comma separated list. Example: 200,200-{$M},{$M},200-400
templateid	string	(readonly) ID of the parent template item prototype.
timeout	string	HTTP agent item prototype field. Item data polling request timeout. Support user macros and LLD macros. default: 3s maximum value: 60s
trapper_hosts	string	Allowed hosts. Used by trapper item prototypes or HTTP item prototypes.
trends	string	A time unit of how long the trends data should be stored. Also accepts user macro and LLD macro. Default: 365d.
units	string	Value units.
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent item prototypes. Required by SSH and Telnet item prototypes.
valuemapid	string	ID of the associated value map.
verify_host	integer	HTTP agent item prototype field. Whether to validate that the host name for the connection matches the one in the host's certificate. 0 - *(default)* Do not validate. 1 - Validate.
verify_peer	integer	HTTP agent item prototype field. Whether to validate that the host's certificate is authentic. 0 - *(default)* Do not validate. 1 - Validate.
discover	integer	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.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Item prototype preprocessing

The item prototype preprocessing object has the following properties.

Property	Type	Description
type (required)	integer	The preprocessing option type. Possible values: 1 - Custom multiplier; 2 - Right trim; 3 - Left trim; 4 - Trim; 5 - Regular expression matching; 6 - Boolean to decimal; 7 - Octal to decimal; 8 - Hexadecimal to decimal; 9 - Simple change; 10 - Change per second; 11 - XML XPath; 12 - JSONPath; 13 - In range; 14 - Matches regular expression; 15 - Does not match regular expression; 16 - Check for error in JSON; 17 - Check for error in XML; 18 - Check for error using regular expression; 19 - Discard unchanged; 20 - Discard unchanged with heartbeat; 21 - JavaScript; 22 - Prometheus pattern; 23 - Prometheus to JSON; 24 - CSV to JSON; 25 - Replace.
params (required)	string	Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n) character.
error_handler (required)	integer	Action type used in case of preprocessing step failure. Possible values: 0 - Error message is set by Zabbix server; 1 - Discard value; 2 - Set custom value; 3 - Set custom error message.
error_handler_params (required)	string	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.

The following parameters and error handlers are supported for each preprocessing type.

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
1	Custom multiplier	number[1, 6]			0, 1, 2, 3
2	Right trim	list of characters[2]			
3	Left trim	list of characters[2]			
4	Trim	list of characters[2]			

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
5	Regular expression	pattern[3]	output[2]		0, 1, 2, 3
6	Boolean to decimal				0, 1, 2, 3
7	Octal to decimal				0, 1, 2, 3
8	Hexadecimal to decimal				0, 1, 2, 3
9	Simple change				0, 1, 2, 3
10	Change per second				0, 1, 2, 3
11	XML XPath	path[4]			0, 1, 2, 3
12	JSONPath	path[4]			0, 1, 2, 3
13	In range	min[1, 6]	max[1, 6]		0, 1, 2, 3
14	Matches regular expression	pattern[3]			0, 1, 2, 3
15	Does not match regular expression	pattern[3]			0, 1, 2, 3
16	Check for error in JSON	path[4]			0, 1, 2, 3
17	Check for error in XML	path[4]			0, 1, 2, 3
18	Check for error using regular expression	pattern[3]	output[2]		0, 1, 2, 3

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
19	Discard unchanged				
20	Discard unchanged with heartbeat	seconds[5, 6]			
21	JavaScript	script[2]			
22	Prometheus pattern	pattern[6, 7]	output[6, 8]		0, 1, 2, 3
23	Prometheus to JSON	pattern[6, 7]			0, 1, 2, 3
24	CSV to JSON	character[2]	character[2]	0,1	0, 1, 2, 3
25	Replace	search string[2]	replacement[2]		

[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>`.

itemprototype.create

Description

`object itemprototype.create(object/array itemPrototypes)`

This method allows to create new item prototypes.

Parameters

`(object/array)` Item prototype to create.

Additionally to the standard item prototype properties, the method accepts the following parameters.

Parameter	Type	Description
ruleid (required)	string	ID of the LLD rule that the item belongs to.
applications	array	IDs of applications to be assigned to the discovered items.
applicationPrototypes	array	Names of application prototypes to be assigned to the item prototype.
preprocessing	array	Item prototype preprocessing options.

Return values

`(object)` Returns an object containing the IDs of the created item prototypes under the `itemids` 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 disk 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": {
        "itemids": [
            "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": ""
            },
            {
                "type": "1",
                "params": "8",
                "error_handler": "2",
                "error_handler_params": "10"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "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:

```
{
    "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:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "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
}
```

Source

CItemPrototype::create() in *ui/include/classes/api/services/CItemPrototype.php*.

itemprototype.delete

Description

`object itemprototype.delete(array itemPrototypeIds)`

This method allows to delete item prototypes.

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
}
```

Source

CItemPrototype::delete() in *ui/include/classes/api/services/CItemPrototype.php*.

itemprototype.get

Description

`integer/array itemprototype.get(object parameters)`

The method allows to retrieve item prototypes according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
discoveryids	string/array	Return only item prototypes that belong to the given LLD rules.
graphids	string/array	Return only item prototypes that are used in the given graph prototypes.
hostids	string/array	Return only item prototypes that belong to the given hosts.
inherited	boolean	If set to `true` return only item prototypes inherited from a template.
itemids	string/array	Return only item prototypes with the given IDs.
monitored	boolean	If set to `true` return only enabled item prototypes that belong to monitored hosts.
templated	boolean	If set to `true` return only item prototypes that belong to templates.
templateids	string/array	Return only item prototypes that belong to the given templates.
triggerids	string/array	Return only item prototypes that are used in the given trigger prototypes.
selectApplications	query	Return an applications property with applications that the item prototype belongs to.
selectApplicationPrototypes	query	Return `applicationPrototypes` property with application prototypes linked to item prototype.
selectDiscoveryRule	query	Return a discoveryRule property with the low-level discovery rule that the item prototype belongs to.
selectGraphs	query	Return a graphs property with graph prototypes that the item prototype is used in. Supports `count`.
selectHosts	query	Return a hosts property with an array of hosts that the item prototype belongs to.
selectTriggers	query	Return a triggers property with trigger prototypes that the item prototype is used in. Supports `count`.

Parameter	Type	Description
selectPreprocessing	query	Return a preprocessing property with item preprocessing options. It has the following properties: `type` - (`string`) The preprocessing option type: 1 - Custom multiplier; 2 - Right trim; 3 - Left trim; 4 - Trim; 5 - Regular expression matching; 6 - Boolean to decimal; 7 - Octal to decimal; 8 - Hexadecimal to decimal; 9 - Simple change; 10 - Change per second; 11 - XML XPath; 12 - JSONPath; 13 - In range; 14 - Matches regular expression; 15 - Does not match regular expression; 16 - Check for error in JSON; 17 - Check for error in XML; 18 - Check for error using regular expression; 19 - Discard unchanged; 20 - Discard unchanged with heartbeat; 21 - JavaScript; 22 - Prometheus pattern; 23 - Prometheus to JSON; 24 - CSV to JSON; 25 - Replace. `params` - (`string`) Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n)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.
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 item prototype belongs to.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectGraphs` - results will be sorted by `name`; `selectTriggers` - results will be sorted by `description`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `itemid`, `name`, `key_`, `delay`, `type` and `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	

Parameter	Type	Description
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 item prototypes from an LLD rule

Retrieve all item prototypes from an LLD rule.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.get",
    "params": {
        "output": "extend",
        "discoveryids": "27426"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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",
            "formula": "",
            "error": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
```

```
        "interfaceid": "0",
        "description": "",
        "inventory_link": "0",
        "lifetime": "30d",
        "state": "0",
        "evaltype": "0",
        "jmx_endpoint": "",
        "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",
        "lastclock": "0",
        "lastns": "0",
        "lastvalue": "0",
        "prevvalue": "0",
        "discover": "0"
    },
    {
        "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": "",
        "formula": "",
        "error": "",
        "logtimefmt": "",
        "templateid": "0",
        "valuemapid": "0",
        "params": "",
        "ipmi_sensor": "",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "flags": "0",
        "interfaceid": "0",
        "description": "The processor load is calculated as system CPU load divided by number of CPU c
        "inventory_link": "0",
```

```
            "lifetime": "0",
            "state": "0",
            "evaltype": "0",
            "jmx_endpoint": "",
            "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",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0",
            "discover": "0"
        }
    ],
    "id": 1
}
```


Finding dependent item

Find one Dependent item for item with ID ”25545”.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "filter": {
            "type": "18",
            "master_itemid": "25545"
        },
        "limit": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": "",
            "formula": "",
            "error": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "state": "0",
            "evaltype": "0",
            "master_itemid": "25545",
            "jmx_endpoint": "",
            "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",
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0",
            "discover": "0"
        }
    ],
    "id": 1
}
```


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": "",
            "hostid": "10254",
            "name": "discovered",
            "key_": "item[{#INAME}]",
            "delay": "{#IUPDATE}",
            "history": "90d",
            "trends": "30d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "formula": "",
            "error": "",
            "logtimefmt": "",
            "templateid": "28255",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "2",
            "interfaceid": "2",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
            "state": "0",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "{#IURL}",
            "query_fields": [],
            "posts": "",
            "status_codes": "",
            "follow_redirects": "0",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
```

```
            "retrieve_mode": "0",
            "request_method": "3",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "discover": "0"
        }
    ],
    "id": 17
}
```

See also

- Application
- 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.

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.

Parameter	Type	Description
applications	array	IDs of the applications to replace the current applications.
applicationPrototypes	array	Names of the application prototypes to replace the current application prototypes.
preprocessing	array	Item prototype preprocessing options to replace the current preprocessing options.

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:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid": "27428",
        "interfaceid": "132"
```

```
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "master_itemid": "25570",
        "itemid": "189030"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "189030"
        ]
    },
    "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"
            }
        ]
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "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
- discoveryrule.get - retrieving LLD rules

- discoveryrule.update - updating LLD rules

> LLD rule object

The following objects are directly related to the `discoveryrule` API.

LLD rule

The low-level discovery rule object has the following properties.

Property	Type	Description
itemid	string	*(readonly)* ID of the LLD rule.
delay (required)	string	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.
hostid (required)	string	ID of the host that the LLD rule belongs to.
interfaceid (required)	string	ID of the LLD rule's host interface. Used only for host LLD rules. Not required for Zabbix agent (active), Zabbix internal, Zabbix trapper, dependent and database monitor LLD rules.
key_ (required)	string	LLD rule key.
name (required)	string	Name of the LLD rule.
type (required)	integer	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;
url (required)	string	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}.
allow_traps	integer	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.
authtype	integer	Used only by SSH agent or HTTP agent LLD rules. SSH agent authentication method possible values: 0 - *(default)* password; 1 - public key. HTTP agent authentication method possible values: 0 - *(default)* none 1 - basic 2 - NTLM

Property	Type	Description
description	string	Description of the LLD rule.
error	string	*(readonly)* Error text if there are problems updating the LLD rule value.
follow_redirects	integer	HTTP agent LLD rule field. Follow response redirects while pooling data. 0 - Do not follow redirects. 1 - *(default)* Follow redirects.
headers	object	HTTP agent LLD rule field. Object with HTTP(S) request headers, where header name is used as key and header value as value. Example: { ”User-Agent”: ”Zabbix” }
http_proxy	string	HTTP agent LLD rule field. HTTP(S) proxy connection string.
ipmi_sensor	string	IPMI sensor. Used only by IPMI LLD rules.
jmx_endpoint	string	JMX agent custom connection string. Default value: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi
lifetime	string	Time period after which items that are no longer discovered will be deleted. Accepts seconds, time unit with suffix and user macro. Default: 30d.
master_itemid	integer	Master item ID. Recursion up to 3 dependent items and maximum count of dependent items equal to 999 are allowed. Discovery rule cannot be master item for another discovery rule. Required for Dependent item.
output_format	integer	HTTP agent LLD rule field. Should response be converted to JSON. 0 - *(default)* Store raw. 1 - Convert to JSON.
params	string	Additional parameters depending on the type of the LLD rule: - executed script for SSH and Telnet LLD rules; - SQL query for database monitor LLD rules; - formula for calculated LLD rules.
password	string	Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent LLD rules.
post_type	integer	HTTP agent LLD rule field. Type of post data body stored in posts property. 0 - *(default)* Raw data. 2 - JSON data. 3 - XML data.
posts	string	HTTP agent LLD rule field. HTTP(S) request body data. Used with post_type.
privatekey	string	Name of the private key file.
publickey	string	Name of the public key file.
query_fields	array	HTTP agent LLD rule field. Query parameters. Array of objects with ‘key’:’value’ pairs, where value can be empty string.
request_method	integer	HTTP agent LLD rule field. Type of request method. 0 - *(default)* GET 1 - POST 2 - PUT 3 - HEAD
retrieve_mode	integer	HTTP agent LLD rule field. What part of response should be stored. 0 - *(default)* Body. 1 - Headers. 2 - Both body and headers will be stored. For request_method HEAD only 1 is allowed value.

Property	Type	Description
snmp_oid	string	SNMP OID.
ssl_cert_file	string	HTTP agent LLD rule field. Public SSL Key file path.
ssl_key_file	string	HTTP agent LLD rule field. Private SSL Key file path.
ssl_key_password	string	HTTP agent LLD rule field. Password for SSL Key file.
state	integer	*(readonly)* State of the LLD rule. Possible values: 0 - *(default)* normal; 1 - not supported.
status	integer	Status of the LLD rule. Possible values: 0 - *(default)* enabled LLD rule; 1 - disabled LLD rule.
status_codes	string	HTTP agent LLD rule field. Ranges of required HTTP status codes separated by commas. Also supports user macros as part of comma separated list. Example: 200,200-{$M},{$M},200-400
templateid	string	(readonly) ID of the parent template LLD rule.
timeout	string	HTTP agent LLD rule field. Item data polling request timeout. Support user macros. default: 3s maximum value: 60s
trapper_hosts	string	Allowed hosts. Used by trapper LLD rules or HTTP agent LLD rules.
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent LLD rules. Required by SSH and Telnet LLD rules.
verify_host	integer	HTTP agent LLD rule field. Whether to validate that the host name for the connection matches the one in the host's certificate. 0 - *(default)* Do not validate. 1 - Validate.
verify_peer	integer	HTTP agent LLD rule field. Whether to validate that the host's certificate is authentic. 0 - *(default)* Do not validate. 1 - Validate.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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:

Property	Type	Description
conditions (required)	array	Set of filter conditions to use for filtering results.
evaltype (required)	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
eval_formula	string	*(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.

Property	Type	Description
formula	string	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.

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:

Property	Type	Description
macro (required)	string	LLD macro to perform the check on.
value (required)	string	Value to compare with.
formulaid	string	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.
operator	integer	Condition operator. Possible values: 8 - *(default)* matches regular expression; 9 - does not match regular expression.

Note:
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:

Property	Type	Description
lld_macro (required)	string	LLD macro.
path (required)	string	Selector for value which will be assigned to corresponding macro.

LLD rule preprocessing

The LLD rule preprocessing object has the following properties.

Property	Type	Description
type (required)	integer	The preprocessing option type. Possible values: 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.
params (required)	string	Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n) character.
error_handler (required)	integer	Action type used in case of preprocessing step failure. Possible values: 0 - Error message is set by Zabbix server; 1 - Discard value; 2 - Set custom value; 3 - Set custom error message.
error_handler_params (required)	string	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.

The following parameters and error handlers are supported for each preprocessing type.

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
5	Regular expression	pattern[1]	output[2]		0, 1, 2, 3
11	XML XPath	path[3]			0, 1, 2, 3
12	JSONPath	path[3]			0, 1, 2, 3
15	Does not match regular expression	pattern[1]			0, 1, 2, 3
16	Check for error in JSON	path[3]			0, 1, 2, 3
17	Check for error in XML	path[3]			0, 1, 2, 3

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
20	Discard unchanged with heartbeat	seconds[4, 5, 6]			
23	Prometheus to JSON	pattern[5, 7]			0, 1, 2, 3
24	CSV to JSON	character[2]	character[2]	0,1	0, 1, 2, 3
25	Replace	search string[2]	replacement[2]		

[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:

Property	Type	Description
name (required)	string	Unique override name.
step (required)	integer	Unique order number of the override.
stop	integer	Stop processing next overrides if matches. Possible values: 0 - *(default)* don't stop processing overrides; 1 - stop processing overrides if filter matches.
filter	object	Override filter.
operations	array	Override operations.

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:

Property	Type	Description
evaltype (required)	integer	Override filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
conditions (required)	array	Set of override filter conditions to use for matching the discovered objects.

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:

Property	Type	Description
macro (required)	string	LLD macro to perform the check on.
value (required)	string	Value to compare with.
formulaid	string	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.
operator	integer	Condition operator. Possible values: 8 - *(default)* matches regular expression; 9 - does not match regular expression.

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:

Property	Type	Description
operationobject (required)	integer	Type of discovered object to perform the action. Possible values: 0 - Item prototype; 1 - Trigger prototype; 2 - Graph prototype; 3 - Host prototype.
operator	integer	Override condition operator. Possible values: 0 - *(default)* equals; 1 - does not equal; 2 - contains; 3 - does not contain; 8 - matches; 9 - does not match.
value	string	Pattern to match item, trigger, graph or host prototype name depending on selected object.

Property	Type	Description
opstatus	object	Override operation status object for item, trigger and host prototype objects.
opdiscover	object	Override operation discover status object (all object types).
opperiod	object	Override operation period (update interval) object for item prototype object.
ophistory	object	Override operation history object for item prototype object.
optrends	object	Override operation trends object for item prototype object.
opseverity	object	Override operation severity object for trigger prototype object.
optag	array	Override operation tag object for trigger prototype object.
optemplate	array	Override operation template object for host prototype object.
opinventory	object	Override operation inventory object for host prototype object.

LLD rule override operation status

LLD rule override operation status that is set to discovered object. It has the following properties:

Property	Type	Description
status (required)	integer	Override the status for selected object. 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:

Property	Type	Description
discover (required)	integer	Override the discover status for selected object. Possible values: 0 - Yes, continue discovering the objects; 1 - No, new objects will not be discovered and existing ones will me 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:

Property	Type	Description
delay (required)	string	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.

LLD rule override operation history

LLD rule override operation history value that is set to discovered item. It has the following properties:

Property	Type	Description
history (required)	string	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.

LLD rule override operation trends

LLD rule override operation trends value that is set to discovered item. It has the following properties:

Property	Type	Description
trends (required)	string	Override the 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.

LLD rule override operation severity

LLD rule override operation severity value that is set to discovered trigger. It has the following properties:

Property	Type	Description
severity (required)	integer	Override the severity of trigger prototype. Possible values are: 0 - *(default)* not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.

LLD rule override operation tag

LLD rule override operation tag object contains tag name and values that are set to discovered trigger. It has the following properties:

Property	Type	Description
tag (required)	string	Override the tag name of trigger prototype tags.
value	string	Override the tag value of trigger prototype tags.

LLD rule override operation template

LLD rule override operation template object that is linked to discovered host. It has the following properties:

Property	Type	Description
templateid (required)	string	Override the template of host prototype linked templates.

LLD rule override operation inventory

LLD rule override operation inventory mode value that is set to discovered host. It has the following properties:

Property	Type	Description
inventory_mode (required)	integer	Override the host prototype inventory mode. Possible values are: -1 - disabled; 0 - *(default)* manual; 1 - automatic.

discoveryrule.copy

Description

`object discoveryrule.copy(object parameters)`

This method allows to copy LLD rules with all of the prototypes to the given hosts.

Parameters

(object) Parameters defining the LLD rules to copy and the target hosts.

Parameter	Type	Description
discoveryids	array	IDs of the LLD rules to be copied.
hostids	array	IDs of the hosts to copy the LLD rules to.

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:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.copy",
    "params": {
        "discoveryids": [
            "27426"
        ],
        "hostids": [
            "10196",
            "10197"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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.

Parameters

(object/array) LLD rules to create.

Additionally to the standard LLD rule properties, the method accepts the following parameters.

Parameter	Type	Description
filter	object	LLD rule filter object for the LLD rule.
preprocessing	array	LLD rule 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 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:

```
{
    "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:

```
{
    "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:

```
{
    "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": [
                {
                    "macro": "{#MACRO1}",
                    "value": "@regex1"
                },
                {
                    "macro": "{#MACRO2}",
                    "value": "@regex2"
                },
                {
                    "macro": "{#MACRO3}",
```

```
                    "value": "@regex3"
                }
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

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}",
                "path": "$.path.1"
            },
            {
                "lld_macro": "{#MACRO2}",
                "path": "$.path.2"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "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:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

Using custom query fields and headers

Create LLD rule with custom query fields and headers.

Request:

```
{
    "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",
        "id": 35,
        "auth": "d678e0b85688ce578ff061bd29a20d3b",
    }
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28336"
        ]
    },
    "id": 35
}
```

Creating a LLD rule with preprocessing

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Discovery rule with preprocessing",
        "key_": "lld.with.preprocessing",
        "hostid": "10001",
        "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"
                            }
                        ]
                    }
                ]
            },
```

```
            {
                "name": "Discover PostgreSQL host",
                "step": "2",
                "stop": "1",
                "filter": {
                    "evaltype": "0",
                    "conditions": [
                        {
                            "macro": "{#UNIT.NAME}",
                            "operator": "8",
                            "value": "^postgresql\\.service$"
                        }
                    ]
                },
                "operations": [
                    {
                        "operationobject": "3",
                        "operator": "2",
                        "value": "Database host",
                        "opstatus": {
                            "status": "0"
                        },
                        "optemplate": [
                            {
                                "templateid": "10263"
                            }
                        ]
                    }
                ]
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "30980"
        ]
    },
    "id": 1
}
```

See also

- LLD rule filter
- LLD macro paths
- LLD rule preprocessing

Source

CDiscoveryRule::create() in *ui/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.delete

Description

`object discoveryrule.delete(array lldRuleIds)`

This method allows to delete LLD rules.

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 `ruleids` 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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
itemids	string/array	Return only LLD rules with the given IDs.
groupids	string/array	Return only LLD rules that belong to the hosts from the given groups.
hostids	string/array	Return only LLD rules that belong to the given hosts.
inherited	boolean	If set to `true` return only LLD rules inherited from a template.
interfaceids	string/array	Return only LLD rules use the given host interfaces.
monitored	boolean	If set to `true` return only enabled LLD rules that belong to monitored hosts.
templated	boolean	If set to `true` return only LLD rules that belong to templates.
templateids	string/array	Return only LLD rules that belong to the given templates.
selectFilter	query	Return a filter property with data of the filter used by the LLD rule.

Parameter	Type	Description
selectGraphs	query	Returns a graphs property with graph prototypes that belong to the LLD rule. Supports `count`.
selectHostPrototypes	query	Return a hostPrototypes property with host prototypes that belong to the LLD rule. Supports `count`.
selectHosts	query	Return a hosts property with an array of hosts that the LLD rule belongs to.
selectItems	query	Return an items property with item prototypes that belong to the LLD rule. Supports `count`.
selectTriggers	query	Return a triggers property with trigger prototypes that belong to the LLD rule. Supports `count`.
selectApplicationPrototypes	query	Return an `applicationPrototypes` property with application prototypes that belong to all item prototypes that belong to this LLD rule.
selectLLDMacroPaths	query	Return an lld_macro_paths property with a list of LLD macros and paths to values assigned to each corresponding macro.
selectPreprocessing	query	Return a `preprocessing` property with LLD rule preprocessing options. 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. `params` - `(string)` Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n) 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.
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 LLD rule belongs to.

Parameter	Type	Description
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectItems`; `selectGraphs`; `selectTriggers`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `itemid`, `name`, `key_`, `delay`, `type` and `status`.
countOutput	boolean	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	

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 from host ”10202”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "output": "extend",
        "hostids": "10202"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "27425",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10202",
            "name": "Network interface discovery",
            "key_": "net.if.discovery",
            "delay": "1h",
            "state": "0",
            "status": "0",
            "trapper_hosts": "",
            "error": "",
```

```
            "templateid": "22444",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "interfaceid": "119",
            "description": "Discovery of network interfaces as defined in global regular expression \"Netw
            "lifetime": "30d",
            "jmx_endpoint": "",
            "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"
        },
        {
            "itemid": "27426",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10202",
            "name": "Mounted filesystem discovery",
            "key_": "vfs.fs.discovery",
            "delay": "1h",
            "state": "0",
            "status": "0",
            "trapper_hosts": "",
            "error": "",
            "templateid": "22450",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "interfaceid": "119",
            "description": "Discovery of file systems of different types as defined in global regular expr
            "lifetime": "30d",
            "jmx_endpoint": "",
            "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"
        }
    ],
    "id": 1
}
```

Retrieving filter conditions

Retrieve 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"
                    },
                    {
                        "macro": "{#MACRO2}",
                        "value": "@regex2",
                        "operator": "8",
                        "formulaid": "B"
                    },
                    {
                        "macro": "{#MACRO3}",
                        "value": "@regex3",
                        "operator": "8",
```

```
                        "formulaid": "C"
                    }
                ],
                "eval_formula": "A and B and C"
            }
        }
    ],
    "id": 1
}
```

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:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "28336",
            "type": "19",
            "snmp_oid": "",
            "hostid": "10257",
            "name": "API HTTP agent",
            "key_": "api_discovery_rule",
            "delay": "5s",
            "history": "90d",
            "trends": "0",
            "status": "0",
            "value_type": "4",
            "trapper_hosts": "",
            "units": "",
            "error": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "1",
            "interfaceid": "5",
            "description": "",
            "inventory_link": "0",
            "lifetime": "30d",
```

```
            "state": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "http://127.0.0.1/discoverer.php",
            "query_fields": [
                {
                    "mode": "json"
                },
                {
                    "elements": "2"
                }
            ],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": {
                "X-Type": "api",
                "Authorization": "Bearer mF_A.B5f-2.1JcM"
            },
            "retrieve_mode": "0",
            "request_method": "1",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0"
        }
    ],
    "id": 39
}
```

Retrieve LLD rule with overrides

Retrieve one LLD rule that has various override settings.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "output": ["name"],
        "itemids": "30980",
        "selectOverrides": ["name", "step", "stop", "filter", "operations"]
    },
    "id": 39,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

Response:

```
{
    "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"
                        },
                        "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"
                        },
                        "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.

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:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22450"
        ]
    },
    "id": 1
}
```

Adding LLD macro paths

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "22450",
        "lld_macro_paths": [
            {
                "lld_macro": "{#MACRO1}",
                "path": "$.json.path"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "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.

Property	Type	Description
maintenanceid	string	*(readonly)* ID of the maintenance.
name (required)	string	Name of the maintenance.
active_since (required)	timestamp	Time when the maintenance becomes active.
active_till (required)	timestamp	Time when the maintenance stops being active.
description	string	Description of the maintenance.
maintenance_type	integer	Type of maintenance. Possible values: 0 - *(default)* with data collection; 1 - without data collection.
tags_evaltype	integer	Problem tag evaluation method. Possible values: 0 - *(default)* And/Or; 2 - Or.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Time period

The time period object is used to define periods when the maintenance must come into effect. It has the following properties.

Property	Type	Description
timeperiodid	string	*(readonly)* ID of the maintenance.
day	integer	Day of the month when the maintenance must come into effect. Required only for monthly time periods.
dayofweek	integer	Days of the week when the maintenance must come into effect. Possible bitmap values are: 1 - Monday; 2 - Tuesday; 4 - Wednesday; 8 - Thursday; 16 - Friday; 32 - Saturday; 64 - Sunday. This is a bitmask field; any sum of possible bitmap values is acceptable (for example, 21 for Monday, Wednesday, and Friday). Used for weekly and monthly time periods. Required only for weekly time periods.

Property	Type	Description
every	integer	For daily and weekly periods `every` defines day or week intervals at which the maintenance must come into effect. For monthly periods `every` defines the week of the month when the maintenance must come into effect. Possible values: 1 - first week; 2 - second week; 3 - third week; 4 - fourth week; 5 - last week.
month	integer	Months when the maintenance must come into effect. Possible bitmap values are: 1 - January; 2 - February; 4 - March; 8 - April; 16 - May; 32 - June; 64 - July; 128 - August; 256 - September; 512 - October; 1024 - November; 2048 - December. This is a bitmask field; any sum of possible bitmap values is acceptable (for example, 585 for January, April, July, and October). Required only for monthly time periods.
period	integer	Duration of the maintenance period in seconds. Default: 3600.
start_date	timestamp	Date when the maintenance period must come into effect. Required only for one time periods. Default: current date.
start_time	integer	Time of day when the maintenance starts in seconds. Required for daily, weekly and monthly periods.
timeperiod_type	integer	Type of time period. Possible values: 0 - *(default)* one time only; 2 - daily; 3 - weekly; 4 - monthly.

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.

Property	Type	Description
tag (required)	string	Problem tag name.

Property	Type	Description
operator	integer	Condition operator. Possible values: 0 - Equals; 2 - *(default)* Contains.
value	string	Problem tag value.

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.

Parameters

`(object/array)` Maintenances to create.

Additionally to the standard maintenance properties, the method accepts the following parameters.

Parameter	Type	Description
groupids (required)	array	IDs of the host groups that will undergo maintenance.
hostids (required)	array	IDs of the hosts that will undergo maintenance.
timeperiods (required)	array	Maintenance time periods.
tags	array	Problem tags. Define what problems must be suppressed. If no tags are given, all active maintenance host problems will be suppressed.

Attention:
At least one host or host group must be defined for each maintenance.

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 "2" with problem tags **service:mysqld** 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:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.create",
    "params": {
        "name": "Sunday maintenance",
        "active_since": 1358844540,
        "active_till": 1390466940,
        "tags_evaltype": 0,
        "groupids": [
            "2"
```

```
        ],
        "timeperiods": [
            {
                "timeperiod_type": 3,
                "every": 1,
                "dayofweek": 64,
                "start_time": 64800,
                "period": 3600
            }
        ],
        "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.

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 maintenance periods

Delete two maintenance periods.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.delete",
    "params": [
        "3",
        "1"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3",
            "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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
groupids	string/array	Return only maintenances that are assigned to the given host groups.
hostids	string/array	Return only maintenances that are assigned to the given hosts.
maintenanceids	string/array	Return only maintenances with the given IDs.
selectGroups	query	Return a groups property with host groups assigned to the maintenance.
selectHosts	query	Return a hosts property with hosts assigned to the maintenance.
selectTags	query	Return a tags property with problem tags of the maintenance.
selectTimeperiods	query	Return a timeperiods property with time periods of the maintenance.
sortfield	string/array	Sort the result by the given properties. Possible values are: `maintenanceid`, `name` and `maintenance_type`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	

Parameter	Type	Description
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 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",
        "selectGroups": "extend",
        "selectTimeperiods": "extend",
        "selectTags": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": [
                {
                    "timeperiodid": "4",
                    "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": "mysqld",
                },
                {
                    "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.

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.

Parameter	Type	Description
groupids	array	IDs of the host groups to replace the current groups.
hostids	array	IDs of the hosts to replace the current hosts.
timeperiods	array	Maintenance time periods to replace the current periods.
tags	array	Problem tags.

Attention:
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 "3" with two different ones.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.update",
    "params": {
```

```
        "maintenanceid": "3",
        "hostids": [
            "10085",
            "10084"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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.

Property	Type	Description
sysmapid	string	*(readonly)* ID of the map.
height (required)	integer	Height of the map in pixels.

Property	Type	Description
name (required)	string	Name of the map.
width (required)	integer	Width of the map in pixels.
backgroundid	string	ID of the image used as the background for the map.
expand_macros	integer	Whether to expand macros in labels when configuring the map. Possible values: 0 - *(default)* do not expand macros; 1 - expand macros.
expandproblem	integer	Whether the problem trigger will be displayed for elements with a single problem. Possible values: 0 - always display the number of problems; 1 - *(default)* display the problem trigger if there's only one problem.
grid_align	integer	Whether to enable grid aligning. Possible values: 0 - disable grid aligning; 1 - *(default)* enable grid aligning.
grid_show	integer	Whether to show the grid on the map. Possible values: 0 - do not show the grid; 1 - *(default)* show the grid.
grid_size	integer	Size of the map grid in pixels. Supported values: 20, 40, 50, 75 and 100. Default: 50.
highlight	integer	Whether icon highlighting is enabled. Possible values: 0 - highlighting disabled; 1 - *(default)* highlighting enabled.
iconmapid	string	ID of the icon map used on the map.
label_format	integer	Whether to enable advanced labels. Possible values: 0 - *(default)* disable advanced labels; 1 - enable advanced labels.
label_location	integer	Location of the map element label. Possible values: 0 - *(default)* bottom; 1 - left; 2 - right; 3 - top.
label_string_host	string	Custom label for host elements. Required for maps with custom host label type.
label_string_hostgroup	string	Custom label for host group elements. Required for maps with custom host group label type.
label_string_image	string	Custom label for image elements. Required for maps with custom image label type.
label_string_map	string	Custom label for map elements. Required for maps with custom map label type.

Property	Type	Description
label_string_trigger	string	Custom label for trigger elements. Required for maps with custom trigger label type.
label_type	integer	Map element label type. Possible values: 0 - label; 1 - IP address; 2 - *(default)* element name; 3 - status only; 4 - nothing.
label_type_host	integer	Label type for host elements. Possible values: 0 - label; 1 - IP address; 2 - *(default)* element name; 3 - status only; 4 - nothing; 5 - custom.
label_type_hostgroup	integer	Label type for host group elements. Possible values: 0 - label; 2 - *(default)* element name; 3 - status only; 4 - nothing; 5 - custom.
label_type_image	integer	Label type for host group elements. Possible values: 0 - label; 2 - *(default)* element name; 4 - nothing; 5 - custom.
label_type_map	integer	Label type for map elements. Possible values: 0 - label; 2 - *(default)* element name; 3 - status only; 4 - nothing; 5 - custom.
label_type_trigger	integer	Label type for trigger elements. Possible values: 0 - label; 2 - *(default)* element name; 3 - status only; 4 - nothing; 5 - custom.
markelements	integer	Whether to highlight map elements that have recently changed their status. Possible values: 0 - *(default)* do not highlight elements; 1 - highlight elements.
severity_min	integer	Minimum severity of the triggers that will be displayed on the map. Refer to the trigger ”severity” property for a list of supported trigger severities.

Property	Type	Description
show_unack	integer	How problems should be displayed. 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.
userid	string	Map owner user ID.
private	integer	Type of map sharing. Possible values: 0 - public map; 1 - *(default)* private map.
show_suppressed	integer	Whether suppressed problems are shown. Possible values: 0 - *(default)* hide suppressed problems; 1 - show suppressed problems.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Map element

The map element object defines an object displayed on a map. It has the following properties.

Property	Type	Description
selementid	string	*(readonly)* ID of the map element.
elements (required)	array	Element data object. Required for host, host group, trigger and map type elements.
elementtype (required)	integer	Type of map element. Possible values: 0 - host; 1 - map; 2 - trigger; 3 - host group; 4 - image.
iconid_off (required)	string	ID of the image used to display the element in default state.
areatype	integer	How separate host group hosts should be displayed. Possible values: 0 - *(default)* the host group element will take up the whole map; 1 - the host group element will have a fixed size.
application	string	Name of the application to display problems from. Used only for host and host group map elements.
elementsubtype	integer	How a host group element should be displayed on a map. Possible values: 0 - *(default)* display the host group as a single element; 1 - display each host in the group separately.
height	integer	Height of the fixed size host group element in pixels. Default: 200.
iconid_disabled	string	ID of the image used to display disabled map elements. Unused for image elements.
iconid_maintenance	string	ID of the image used to display map elements in maintenance. Unused for image elements.
iconid_on	string	ID of the image used to display map elements with problems. Unused for image elements.
label	string	Label of the element.

Property	Type	Description
label_location	integer	Location of the map element label. Possible values: -1 - *(default)* default location; 0 - bottom; 1 - left; 2 - right; 3 - top.
permission	integer	Type of permission level. Possible values: -1 - none; 2 - read only; 3 - read-write.
sysmapid	string	*(readonly)* ID of the map that the element belongs to.
urls	array	Map element URLs. The map element URL object is described in detail below.
use_iconmap	integer	Whether icon mapping must be used for host elements. Possible values: 0 - do not use icon mapping; 1 - *(default)* use icon mapping.
viewtype	integer	Host group element placing algorithm. Possible values: 0 - *(default)* grid.
width	integer	Width of the fixed size host group element in pixels. Default: 200.
x	integer	X-coordinates of the element in pixels. Default: 0.
y	integer	Y-coordinates of the element in pixels. Default: 0.

Map element Host

The map element Host object defines one host element.

Property	Type	Description
hostid	string	Host ID

Map element Host group

The map element Host group object defines one host group element.

Property	Type	Description
groupid	string	Host group ID

Map element Map

The map element Map object defines one map element.

Property	Type	Description
sysmapid	string	Map ID

Map element Trigger

The map element Trigger object defines one or more trigger elements.

Property	Type	Description
triggerid	string	Trigger ID

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:

Property	Type	Description
sysmapelementurlid	string	*(readonly)* ID of the map element URL.
name (required)	string	Link caption.
url (required)	string	Link URL.
selementid	string	ID of the map element that the URL belongs to.

Map link

The map link object defines a link between two map elements. It has the following properties.

Property	Type	Description
linkid	string	*(readonly)* ID of the map link.
selementid1 (required)	string	ID of the first map element linked on one end.
selementid2 (required)	string	ID of the first map element linked on the other end.
color	string	Line color as a hexadecimal color code. Default: 000000.
drawtype	integer	Link line draw style. Possible values: 0 - *(default)* line; 2 - bold line; 3 - dotted line; 4 - dashed line.
label	string	Link label.
linktriggers	array	Map link triggers to use as link status indicators. The map link trigger object is described in detail below.
permission	integer	Type of permission level. Possible values: -1 - none; 2 - read only; 3 - read-write.
sysmapid	string	ID of the map the link belongs to.

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:

Property	Type	Description
linktriggerid	string	*(readonly)* ID of the map link trigger.
triggerid (required)	string	ID of the trigger used as a link indicator.

Property	Type	Description
color	string	Indicator color as a hexadecimal color code. Default: DD0000.
drawtype	integer	Indicator draw style. Possible values: 0 - *(default)* line; 2 - bold line; 3 - dotted line; 4 - dashed line.
linkid	string	ID of the map link that the link trigger belongs to.

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:

Property	Type	Description
sysmapurlid	string	*(readonly)* ID of the map URL.
name (required)	string	Link caption.
url (required)	string	Link URL.
elementtype	integer	Type of map element for which the URL will be available. Refer to the map element "type" property for a list of supported types. Default: 0.
sysmapid	string	ID of the map that the URL belongs to.

Map user

List of map permissions based on users. It has the following properties:

Property	Type	Description
sysmapuserid	string	*(readonly)* ID of the map user.
userid (required)	string	User ID.
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write;

Map user group

List of map permissions based on user groups. It has the following properties:

Property	Type	Description
sysmapusrgrpid	string	*(readonly)* ID of the map user group.
usrgrpid (required)	string	User group ID.
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write;

Map shapes

The map shape object defines a geometric shape (with or without text) displayed on a map. It has the following properties:

Property	Type	Description
sysmap_shapeid	string	*(readonly)* ID of the map shape element.
type (required)	integer	Type of map shape element. Possible values: 0 - rectangle; 1 - ellipse. Property is required when new shapes are created.
x	integer	X-coordinates of the shape in pixels. Default: 0.
y	integer	Y-coordinates of the shape in pixels. Default: 0.
width	integer	Width of the shape in pixels. Default: 200.
height	integer	Height of the shape in pixels. Default: 200.
text	string	Text of the shape.
font	integer	Font of the text within shape. Possible values: 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 Default: 9.
font_size	integer	Font size in pixels. Default: 11.
font_color	string	Font color. Default: ’000000’.
text_halign	integer	Horizontal alignment of text. Possible values: 0 - center; 1 - left; 2 - right. Default: 0.

Property	Type	Description
text_valign	integer	Vertical alignment of text. Possible values: 0 - middle; 1 - top; 2 - bottom. Default: 0.
border_type	integer	Type of the border. Possible values: 0 - none; 1 - ——————; 2 - ┅┅; 3 - – – –. Default: 0.
border_width	integer	Width of the border in pixels. Default: 0.
border_color	string	Border color. Default: '000000'.
background_color	string	Background color (fill color). Default: `(empty)`.
zindex	integer	Value used to order all shapes and lines (z-index). Default: 0.

Map lines

The map line object defines a line displayed on a map. It has the following properties:

Property	Type	Description
sysmap_shapeid	string	*(readonly)* ID of the map shape element.
x1	integer	X-coordinates of the line point 1 in pixels. Default: 0.
y1	integer	Y-coordinates of the line point 1 in pixels. Default: 0.
x2	integer	X-coordinates of the line point 2 in pixels. Default: 200.
y2	integer	Y-coordinates of the line point 2 in pixels. Default: 200.
line_type	integer	Line type. Possible values: 0 - none; 1 - ——————; 2 - ┅┅; 3 - – – –. Default: 0.
line_width	integer	Line width in pixels. Default: 0.

Property	Type	Description
line_color	string	Line color. Default: '000000'.
zindex	integer	Value used to order all shapes and lines (z-index). Default: 0.

map.create

Description

`object map.create(object/array maps)`

This method allows to create new maps.

Parameters

`(object/array)` Maps to create.

Additionally to the standard map properties, the method accepts the following parameters.

Parameter	Type	Description
links	array	Map links to be created on the map.
selements	array	Map elements to be created on the map.
urls	array	Map URLs to be created on the map.
users	array	Map user shares to be created on the map.
userGroups	array	Map user group shares to be created on the map.
shapes	array	Map shapes to be created on the map.
lines	array	Map lines to be created on the map.

Note:
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:

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Map",
        "width": 600,
        "height": 600
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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:

```
{
    "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"}
                ],
                "elementtype": 0,
                "iconid_off": "2"
            }
        ],
        "links": [
            {
                "selementid1": "1",
                "selementid2": "2"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ]
    },
    "id": 1
}
```

Create a trigger map

Create a map with trigger element, which contains two triggers.

Request:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "10"
        ]
    },
    "id": 1
}
```

Map sharing

Create a map with two types of sharing (user and user group).

Request:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ]
    },
    "id": 1
}
```

Map shapes

Create a map with map name title.

Request:

```
{
    "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}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "10"
        ]
    },
    "id": 1
}
```

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

Source

CMap::create() in *ui/include/classes/api/services/CMap.php*.

map.delete

Description

`object map.delete(array mapIds)`

This method allows to delete maps.

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:

```
{
    "jsonrpc": "2.0",
    "method": "map.delete",
    "params": [
```

```
        "12",
        "34"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "12",
            "34"
        ]
    },
    "id": 1
}
```

Source

CMap::delete() 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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
sysmapids	string/array	Returns only maps with the given IDs.
userids	string/array	Returns only maps that belong to the given user IDs.
expandUrls	flag	Adds global map URLs to the corresponding map elements and expands macros in all map element URLs.
selectIconMap	query	Returns an iconmap property with the icon map used on the map.
selectLinks	query	Returns a links property with the map links between elements.
selectSelements	query	Returns a selements property with the map elements.
selectUrls	query	Returns a urls property with the map URLs.
selectUsers	query	Returns a users property with users that the map is shared with.
selectUserGroups	query	Returns a userGroups property with user groups that the map is shared with.
selectShapes	query	Returns a shapes property with the map shapes.
selectLines	query	Returns a lines property with the map lines.
sortfield	string/array	Sort the result by the given properties. Possible values are: `name`, `width` and `height`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	

Parameter	Type	Description
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

Retrieve a map

Retrieve all data about map "3".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.get",
    "params": {
        "output": "extend",
        "selectSelements": "extend",
        "selectLinks": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
        "selectShapes": "extend",
        "selectLines": "extend",
        "sysmapids": "3"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "selements": [
                {
                    "selementid": "10",
                    "sysmapid": "3",
                    "elementtype": "4",
                    "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",
                    "application": "",
                    "urls": [],
                    "elements": []
                },
                {
```

```
                "selementid": "11",
                "sysmapid": "3",
                "elementtype": "4",
                "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",
                "application": "",
                "urls": [],
                "elements": []
            },
            {
                "selementid": "12",
                "sysmapid": "3",
                "elementtype": "0",
                "iconid_off": "185",
                "iconid_on": "0",
                "label": "{HOST.NAME}\r\n{HOST.CONN}",
                "label_location": "0",
                "x": "111",
                "y": "61",
                "iconid_disabled": "0",
                "iconid_maintenance": "0",
                "elementsubtype": "0",
                "areatype": "0",
                "width": "200",
                "height": "200",
                "viewtype": "0",
                "use_iconmap": "0",
                "application": "",
                "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": [
            {
                "sysmapusrgrpid": "1",
                "usrgrpid": "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",
                "text_halign":"0",
                "text_valign":"0",
                "border_type":"0",
                "border_width":"0",
                "border_color":"000000",
                "background_color":"",
                "zindex":"0"
            }
        ],
        "lines":[
            {
                "sysmap_shapeid":"2",
                "x1": 30,
                "y1": 10,
                "x2": 100,
                "y2": 50,
                "line_type": 1,
                "line_width": 10,
                "line_color": "009900",
                "zindex":"1"
            }
        ],
        "sysmapid": "3",
        "name": "Local network",
        "width": "400",
        "height": "400",
        "backgroundid": "0",
        "label_type": "2",
        "label_location": "3",
        "highlight": "1",
        "expandproblem": "1",
        "markelements": "0",
        "show_unack": "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": "",
            "iconmapid": "0",
            "expand_macros": "0",
            "severity_min": "0",
            "userid": "1",
            "private": "1",
            "show_suppressed": "1"
        }
    ],
    "id": 1
}
```


See also

- Icon map
- Map element
- Map link
- Map URL
- Map user
- Map user group
- Map shapes
- Map lines

Source

CMap::get() in *ui/include/classes/api/services/CMap.php*.

map.update

Description

`object map.update(object/array maps)`

This method allows to update existing maps.

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.

Parameter	Type	Description
links	array	Map links to replace the existing links.
selements	array	Map elements to replace the existing elements.
urls	array	Map URLs to replace the existing URLs.
users	array	Map user shares to replace the existing elements.
userGroups	array	Map user group shares to replace the existing elements.
shapes	array	Map shapes to replace the existing shapes.
lines	array	Map lines to replace the existing lines.

Note:
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:

```
{
    "jsonrpc": "2.0",
    "method": "map.update",
    "params": {
        "sysmapid": "8",
        "width": 1200,
        "height": 1200
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "8"
        ]
    },
    "id": 1
}
```

Change map owner

Available only for admins and super admins.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.update",
    "params": {
        "sysmapid": "9",
        "userid": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
- mediatype.update - updating 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.

Property	Type	Description
mediatypeid	string	*(readonly)* ID of the media type.
name (required)	string	Name of the media type.
type (required)	integer	Transport used by the media type. Possible values: 0 - email; 1 - script; 2 - SMS; 4 - Webhook.
exec_path	string	For script media types `exec_path` contains the name of the executed script. Required for script media types.
gsm_modem	string	Serial device name of the GSM modem. Required for SMS media types.
passwd	string	Authentication password. Used for email media types.
smtp_email	string	Email address from which notifications will be sent. Required for email media types.
smtp_helo	string	SMTP HELO. Required for email media types.
smtp_server	string	SMTP server. Required for email media types.
smtp_port	integer	SMTP server port to connect to.

Property	Type	Description
smtp_security	integer	SMTP connection security level to use. Possible values: 0 - None; 1 - STARTTLS; 2 - SSL/TLS.
smtp_verify_host	integer	SSL verify host for SMTP. Possible values: 0 - No; 1 - Yes.
smtp_verify_peer	integer	SSL verify peer for SMTP. Possible values: 0 - No; 1 - Yes.
smtp_authentication	integer	SMTP authentication method to use. Possible values: 0 - None; 1 - Normal password.
status	integer	Whether the media type is enabled. Possible values: 0 - *(default)* enabled; 1 - disabled.
username	string	User name. Used for email media types.
exec_params	string	Script parameters. Each parameter ends with a new line feed.
maxsessions	integer	The maximum number of alerts that can be processed in parallel. Possible values for SMS: 1 - *(default)* Possible values for other media types: 0-100
maxattempts	integer	The maximum number of attempts to send an alert. Possible values: 1-100 Default value: 3
attempt_interval	string	The interval between retry attempts. Accepts seconds and time unit with suffix. Possible values: 0-1h Default value: 10s
content_type	integer	Message format. Possible values: 0 - plain text; 1 - *(default)* html.
script	text	Media type webhook script javascript body.

Property	Type	Description
timeout	string	Media type webhook script timeout. Accepts seconds and time unit with suffix. Possible values: 1-60s Default value: 30s
process_tags	integer	Defines should the webhook script response to be interpreted as tags and these tags should be added to associated event. Possible values: 0 - *(default)* Ignore webhook script response. 1 - Process webhook script response as tags.
show_event_menu	integer	Show media type entry in `problem.get` and `event.get` property `urls`. Possible values: 0 - *(default)* Do not add `urls` entry. 1 - Add media type to `urls` property.
event_menu_url	string	Define `url` property of media type entry in `urls` property of `problem.get` and `event.get`.
event_menu_name	string	Define `name` property of media type entry in `urls` property of `problem.get` and `event.get`.
parameters	array	Array of webhook input parameters.
description	text	Media type description.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Webhook parameters

Parameters passed to webhook script when it is called, have the following properties.

Property	Type	Description
name (required)	string	Parameter name.
value	string	Parameter value, supports macros. Supported macros are described on the Supported macros page.

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.

Property	Type	Description
eventsource (required)	integer	Event source. Possible values: 0 - triggers; 1 - discovery; 2 - autoregistration; 3 - internal.
recovery (required)	integer	Operation mode. Possible values: 0 - operations; 1 - recovery operations; 2 - update operations.
subject	string	Message subject.

Property	Type	Description
message	string	Message text.

mediatype.create

Description

`object mediatype.create(object/array mediaTypes)`

This method allows to create new media types.

Parameters

`(object/array)` Media types to create.

Additionally to the standard media type properties, the method accepts the following parameters.

Parameter	Type	Description
parameters	array	Webhook parameters to be created for the media type.
message_templates	array	Message templates to be created for the media type.

Return values

`(object)` 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:

```
{
    "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}",
                "message": "Problem \"{EVENT.NAME}\" on host \"{HOST.NAME}\" started at {EVENT.TIME}."
            },
            {
                "eventsource": "0",
                "recovery": "1",
                "subject": "Resolved in {EVENT.DURATION}: {EVENT.NAME}",
                "message": "Problem \"{EVENT.NAME}\" on host \"{HOST.NAME}\" has been resolved at {EVENT.R
            },
            {
                "eventsource": "0",
                "recovery": "2",
                "subject": "Updated problem in {EVENT.AGE}: {EVENT.NAME}",
                "message": "{USER.FULLNAME} {EVENT.UPDATE.ACTION} problem \"{EVENT.NAME}\" on host \"{HOST
            }
        ]
```

```
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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:

```
{
    "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 = {\r\n    token: null,\r\n    to: null,\r\n    subject: null,\r\n    messa
        "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": [
            "9"
        ]
    },
    "id": 1
}
```

Source

CMediaType::create() in *ui/include/classes/api/services/CMediaType.php*.

mediatype.delete

Description

`object mediatype.delete(array mediaTypeIds)`

This method allows to delete media types.

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

Deleting multiple media types

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
}
```

Source

CMediaType::delete() in *ui/include/classes/api/services/CMediaType.php*.

mediatype.get

Description

`integer/array mediatype.get(object parameters)`

The method allows to retrieve media types according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Note:
Since Zabbix 5.0.44, when requesting user-related information of media types, *Admin* type users may retrieve only data about their own user. For an example, see Retrieving media types as *Admin*.

Parameter	Type	Description
mediatypeids	string/array	Return only media types with the given IDs.
mediaids	string/array	Return only media types used by the given media.
userids	string/array	Return only media types used by the given users.
selectMessageTemplates	query	Return a message_templates property with an array of media type messages. Since Zabbix 5.0.44, this parameter is supported only for *Super admin* type users.
selectUsers	query	Return a users property with the users that use the media type. See user.get for restrictions based on user type.
sortfield	string/array	Sort the result by the given properties. Possible values are: `mediatypeid`.
filter	object	Return only those results that exactly match the given filter. Accepts an object, where the keys are property names, and the values are either a single value or an array of values to match against. Does not support properties of `text` data type. Possible Media type object properties for *Admin* type users (since Zabbix 5.0.44): `mediatypeid`, `name`, `type`, `status`, `maxattempts`.

Parameter	Type	Description
output	query	Media type object properties to be returned. Since Zabbix 5.0.44, *Admin* type users may retrieve only the following properties: `mediatypeid`, `name`, `type`, `status`, `maxattempts`, `description`. For an example, see Retrieving media types as *Admin*. Default: `extend`.
search	object	Return results that match the given pattern (case-insensitive). Accepts an object, 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 "%…%"` search. Supports only properties of `string` and `text` data type. Possible Media type object properties for *Admin* type users (since Zabbix 5.0.44): `name`, `description`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
limit	integer	
preservekeys	boolean	
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 media types

Retrieve all configured media types.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.get",
    "params": {
        "output": "extend",
        "selectMessageTemplates": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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}",
                "message": "Problem started at {EVENT.TIME} on {EVENT.DATE}\r\nProblem name: {EVENT.NA
            },
            {
                "eventsource": "0",
                "recovery": "1",
                "subject": "Resolved: {EVENT.NAME}",
                "message": "Problem has been resolved at {EVENT.RECOVERY.TIME} on {EVENT.RECOVERY.DATE
            },
            {
                "eventsource": "0",
                "recovery": "2",
                "subject": "Updated problem: {EVENT.NAME}",
                "message": "{USER.FULLNAME} {EVENT.UPDATE.ACTION} problem at {EVENT.UPDATE.DATE} {EVEN
            },
            {
                "eventsource": "1",
                "recovery": "0",
                "subject": "Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IPADDRESS}",
                "message": "Discovery rule: {DISCOVERY.RULE.NAME}\r\n\r\nDevice IP: {DISCOVERY.DEVICE.
            },
            {
                "eventsource": "2",
                "recovery": "0",
                "subject": "Autoregistration: {HOST.HOST}",
                "message": "Host name: {HOST.HOST}\r\nHost IP: {HOST.IP}\r\nAgent port: {HOST.PORT}"
            }
        ],
        "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": [
                {
                    "eventsource": "0",
                    "recovery": "0",
                    "subject": "",
                    "message": "{EVENT.SEVERITY}: {EVENT.NAME}\r\nHost: {HOST.NAME}\r\n{EVENT.DATE} {EVENT
                },
                {
                    "eventsource": "0",
                    "recovery": "1",
                    "subject": "",
                    "message": "RESOLVED: {EVENT.NAME}\r\nHost: {HOST.NAME}\r\n{EVENT.DATE} {EVENT.TIME}"
                },
                {
                    "eventsource": "0",
                    "recovery": "2",
                    "subject": "",
                    "message": "{USER.FULLNAME} {EVENT.UPDATE.ACTION} problem at {EVENT.UPDATE.DATE} {EVEN
                },
                {
                    "eventsource": "1",
                    "recovery": "0",
                    "subject": "",
                    "message": "Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IPADDRESS}"
                },
                {
                    "eventsource": "2",
                    "recovery": "0",
                    "subject": "",
                    "message": "Autoregistration: {HOST.HOST}\r\nHost IP: {HOST.IP}\r\nAgent port: {HOST.P
                }
            ],
            "parameters": []
        }
    ],
    "id": 1
}
```


Retrieving media types as *Admin*

As an *Admin* type user, retrieve all media types that are enabled, with users that use these media types. The following example returns two media types:

- email media type with one user (since Zabbix 5.0.44, only *Admin* type user's own user);
- SMS media type with no users.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.get",
    "params": {
        "output": "extend",
        "filter": {
            "status": 0
        },
        "selectUsers": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mediatypeid": "1",
            "type": "0",
            "name": "Email",
            "status": "0",
            "description": "",
            "maxattempts": "3",
            "users": [
                {
                    "userid": "3",
                    "alias": "database-admin",
                    "name": "John",
                    "surname": "Doe",
                    "url": "",
                    "autologin": "0",
                    "autologout": "0",
                    "lang": "en_GB",
                    "type": "2",
                    "theme": "default",
                    "attempt_failed": "0",
                    "attempt_ip": "",
                    "attempt_clock": "0",
                    "rows_per_page": "50"
                }
            ]
        },
        {
            "mediatypeid": "3",
            "type": "2",
            "name": "SMS",
            "status": "0",
            "description": "",
            "maxattempts": "3",
            "users": []
        }
    ],
    "id": 1
}
```

See also

- User

Source

CMediaType::get() in *ui/include/classes/api/services/CMediaType.php*.

mediatype.update

Description

`object mediatype.update(object/array mediaTypes)`

This method allows to update existing media types.

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.

Parameter	Type	Description
parameters	array	Webhook parameters to replace the current webhook parameters.
message_templates	array	Message templates to replace the current message templates.

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
}
```

Source

CMediaType::update() in *ui/include/classes/api/services/CMediaType.php*.

Problem

This class is designed to work with problems.

Object references:

- Problem

Available methods:

- problem.get - retrieving problems

> Problem object

The following objects are directly related to the `problem` API.

Problem

> **Note:**
> Problems are created by the Zabbix server and cannot be modified via the API.

The problem object has the following properties.

Property	Type	Description
eventid	string	ID of the problem event.
source	integer	Type of the problem event. Possible values: 0 - event created by a trigger; 3 - internal event.
object	integer	Type of object that is related to the problem event. Possible values for trigger events: 0 - trigger. Possible values for internal events: 0 - trigger; 4 - item; 5 - LLD rule.
objectid	string	ID of the related object.
clock	timestamp	Time when the problem event was created.
ns	integer	Nanoseconds when the problem event was created.
r_eventid	string	Recovery event ID.
r_clock	timestamp	Time when the recovery event was created.
r_ns	integer	Nanoseconds when the recovery event was created.
correlationid	string	Correlation rule ID if this event was recovered by global correlation rule.
userid	string	User ID if the problem was manually closed.
name	string	Resolved problem name.
acknowledged	integer	Acknowledge state for problem. Possible values: 0 - not acknowledged; 1 - acknowledged.

Property	Type	Description
severity	integer	Problem current severity. Possible values: 0 - not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
suppressed	integer	Whether the problem is suppressed. Possible values: 0 - problem is in normal state; 1 - problem is suppressed.
opdata	string	Operational data with expanded macros.
urls	array	Active media type URLs.

Problem tag

The problem tag object has the following properties.

Property	Type	Description
tag	string	Problem tag name.
value	string	Problem tag value.

Media type URL

The media type URL object has the following properties.

Property	Type	Description
name	string	Media type defined URL name.
url	string	Media type defined URL value.

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 the properties contains an unexpanded macro, both properties will be excluded from results. For supported macros, see *Supported macros*.

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.

Attention:
This method may return problems of a deleted entity if these problems have not been removed by the housekeeper yet.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
eventids	string/array	Return only problems with the given IDs.
groupids	string/array	Return only problems created by objects that belong to the given host groups.
hostids	string/array	Return only problems created by objects that belong to the given hosts.
objectids	string/array	Return only problems created by the given objects.
applicationids	string/array	Return only problems created by objects that belong to the given applications. Applies only if object is trigger or item.
source	integer	Return only problems with the given type. Refer to the problem event object page for a list of supported event types. Default: 0 - problem created by a trigger.
object	integer	Return only problems created by objects of the given type. Refer to the problem event object page for a list of supported object types. Default: 0 - trigger.
acknowledged	boolean	`true` - return acknowledged problems only; `false` - unacknowledged only.
suppressed	boolean	`true` - return only suppressed problems; `false` - return problems in the normal state.
severities	integer/array	Return only problems with given event severities. Applies only if object is trigger.
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.
tags	array of objects	Return only problems with given tags. Exact match by tag and case-insensitive search by value and operator. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all problems. Possible operator types: 0 - (default) Like; 1 - Equal.
recent	boolean	`true` - return PROBLEM and recently RESOLVED problems (depends on Display OK triggers for N seconds) Default: `false` - UNRESOLVED problems only
eventid_from	string	Return only problems with IDs greater or equal to the given ID.
eventid_till	string	Return only problems with IDs less or equal to the given ID.
time_from	timestamp	Return only problems that have been created after or at the given time.
time_till	timestamp	Return only problems that have been created before or at the given time.

Parameter	Type	Description
selectAcknowledges	query	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; Supports `count`.
selectTags	query	Return a tags property with the problem tags. Output format: `[{"tag": "<tag>", "value": "<value>"}, ...]`.
selectSuppressionData	query	Return a `suppression_data` property with the list of maintenances: `maintenanceid` - `(string)` ID of the maintenance; `suppress_until` - `(integer)` time until the problem is suppressed.
sortfield	string/array	Sort the result by the given properties. Possible values are: `eventid`.
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 trigger problem events

Retrieve recent events from trigger "15112."

Request:

```
{
    "jsonrpc": "2.0",
```

```
    "method": "problem.get",
    "params": {
        "output": "extend",
        "selectAcknowledges": "extend",
        "selectTags": "extend",
        "selectSuppressionData": "extend",
        "objectids": "15112",
        "recent": "true",
        "sortfield": ["eventid"],
        "sortorder": "DESC"
    },
    "auth": "67f45d3eb1173338e1b1647c4bdc1916",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "eventid": "1245463",
            "source": "0",
            "object": "0",
            "objectid": "15112",
            "clock": "1472457242",
            "ns": "209442442",
            "r_eventid": "1245468",
            "r_clock": "1472457285",
            "r_ns": "125644870",
            "correlationid": "0",
            "userid": "1",
            "name": "Zabbix agent on localhost is unreachable for 5 minutes",
            "acknowledged": "1",
            "severity": "3",
            "opdata": "",
            "acknowledges": [
                {
                    "acknowledgeid": "14443",
                    "userid": "1",
                    "eventid": "1245463",
                    "clock": "1472457281",
                    "message": "problem solved",
                    "action": "6",
                    "old_severity": "0",
                    "new_severity": "0"
                }
            ],
            "suppression_data": [
                {
                    "maintenanceid": "15",
                    "suppress_until": "1472511600"
                }
            ],
            "suppressed": "1",
            "tags": [
                {
                    "tag": "test tag",
                    "value": "test value"
                }
            ]
        }
    ],
```

```
    "id": 1
}
```

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.

Property	Type	Description
proxyid	string	*(readonly)* ID of the proxy.
host (required)	string	Name of the proxy.
status (required)	integer	Type of proxy. Possible values: 5 - active proxy; 6 - passive proxy.
description	text	Description of the proxy.
lastaccess	timestamp	*(readonly)* Time when the proxy last connected to the server.
tls_connect	integer	Connections to host. Possible values are: 1 - *(default)* No encryption; 2 - PSK; 4 - certificate.

Property	Type	Description
tls_accept	integer	Connections from host. Possible bitmap values are: 1 - *(default)* No encryption; 2 - PSK; 4 - certificate. This is a bitmask field; any sum of possible bitmap values is acceptable (for example, 6 for PSK and certificate).
tls_issuer	string	Certificate issuer.
tls_subject	string	Certificate subject.
tls_psk_identity	string	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	The preshared key, at least 32 hex digits. Required if either `tls_connect` or `tls_accept` has PSK enabled.
proxy_address	string	Comma-delimited IP addresses or DNS names of active Zabbix proxy.
auto_compress	integer	*(readonly)* Indicates if communication between Zabbix server and proxy is compressed. Possible values are: 0 - No compression; 1 - Compression enabled;

Note that for some methods (update, delete) the required/optional parameter combination is different.

Proxy interface

The proxy interface object defines the interface used to connect to a passive proxy. It has the following properties.

Property	Type	Description
interfaceid	string	*(readonly)* ID of the interface.
dns (required)	string	DNS name to connect to. Can be empty if connections are made via IP address.
ip (required)	string	IP address to connect to. Can be empty if connections are made via DNS names.
port (required)	string	Port number to connect to.
useip (required)	integer	Whether the connection should be made via IP address. Possible values are: 0 - connect using DNS name; 1 - connect using IP address.
hostid	string	*(readonly)* ID of the proxy the interface belongs to.

proxy.create

Description

`object proxy.create(object/array proxies)`

This method allows to create new proxies.

Parameters

`(object/array)` Proxies to create.

Additionally to the standard proxy properties, the method accepts the following parameters.

Parameter	Type	Description
hosts	array	Hosts 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 `hostid` property defined.
interface	object	Host interface to be created for the passive proxy. Required for passive proxies.

Return values

`(object)` Returns an object containing the IDs of the created proxies under the `proxyids` 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:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.create",
    "params": {
        "host": "Active proxy",
        "status": "5",
        "hosts": [
            {
                "hostid": "10279"
            }
        ]
    },
    "auth": "ab9638041ec6922cb14b07982b268f47",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10280"
        ]
    },
    "id": 1
}
```

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": "ab9638041ec6922cb14b07982b268f47",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10284"
        ]
    },
    "id": 1
}
```

See also

- Host
- Proxy interface

Source

CProxy::create() in *ui/include/classes/api/services/CProxy.php*.

proxy.delete

Description

`object proxy.delete(array proxies)`

This method allows to delete proxies.

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::delete() 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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
proxyids	string/array	Return only proxies with the given IDs.
selectHosts	query	Return a hosts property with the hosts monitored by the proxy.
selectInterface	query	Return an interface property with the proxy interface used by a passive proxy.
sortfield	string/array	Sort the result by the given properties. Possible values are: `hostid`, `host` and `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
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

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:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "host": "Active proxy",
            "status": "5",
            "lastaccess": "0",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": "",
            "proxy_address": "",
            "auto_compress": "0",
            "proxyid": "30091",
            "interface": []
        },
        {
            "host": "Passive proxy",
            "status": "6",
            "lastaccess": "0",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": "",
            "proxy_address": "",
            "auto_compress": "0",
            "proxyid": "30092",
            "interface": {
                "interfaceid": "30109",
                "hostid": "30092",
                "useip": "1",
                "ip": "127.0.0.1",
                "dns": "",
                "port": "10051"
            }
        }
    ],
    "id": 1
}
```

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.

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.

Parameter	Type	Description
hosts	array	Hosts 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 `hostid` property defined.
interface	object	Host interface to replace the existing interface for the passive proxy.

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:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.update",
    "params": {
        "proxyid": "10293",
        "hosts": [
            "10294",
            "10295"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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:

```
{
    "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
}
```

See also

- Host
- Proxy interface

Source

CProxy::update() in *ui/include/classes/api/services/CProxy.php*.

Screen

This class is designed to work with screen.

Object references:

- Screen
- Screen user
- Screen user group

Available methods:

- screen.create - creating new screen
- screen.delete - deleting screens
- screen.get - retrieving screens
- screen.update - updating screens

> Screen object

The following objects are directly related to the `screen` API.

Screen

The screen object has the following properties.

Property	Type	Description
screenid	string	*(readonly)* ID of the screen.
name (required)	string	Name of the screen.

Property	Type	Description
hsize	integer	Width of the screen. Default: 1
vsize	integer	Height of the screen. Default: 1
userid	string	Screen owner user ID.
private	integer	Type of screen sharing. Possible values: 0 - public screen; 1 - *(default)* private screen.

Screen user

List of screen permissions based on users. It has the following properties:

Property	Type	Description
screenuserid	string	*(readonly)* ID of the screen user.
userid (required)	string	User ID.
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write;

Screen user group

List of screen permissions based on user groups. It has the following properties:

Property	Type	Description
screenusrgrpid	string	*(readonly)* ID of the screen user group.
usrgrpid (required)	string	User group ID.
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write;

screen.create

Description

`object screen.create(object/array screens)`

This method allows to create new screens.

Parameters

`(object/array)` Screens to create.

Additionally to the standard screen properties, the method accepts the following parameters.

Parameter	Type	Description
screenitems	array	Screen items to be created for the screen.
users	array	Screen user shares to be created on the screen.
userGroups	array	Screen user group shares to be created on the screen.

Return values

(object) Returns an object containing the IDs of the created screens under the screenids property. The order of the returned IDs matches the order of the passed screens.

Examples

Creating a screen

Create a screen named "Graphs" with 2 rows and 3 columns and add a graph to the upper-left cell.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screen.create",
    "params": {
        "name": "Graphs",
        "hsize": 3,
        "vsize": 2,
        "screenitems": [
            {
                "resourcetype": 0,
                "resourceid": "612",
                "rowspan": 1,
                "colspan": 1,
                "x": 0,
                "y": 0
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "26"
        ]
    },
    "id": 1
}
```

Screen sharing

Create a screen with two types of sharing (user and user group).

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screen.create",
    "params": {
        "name": "Screen sharing",
        "hsize": 3,
        "vsize": 2,
        "users": [
            {
                "userid": "4",
                "permission": "3"
            }
        ],
        "userGroups": [
            {
                "usrgrpid": "7",
```

```
                "permission": "2"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "83"
        ]
    },
    "id": 1
}
```

See also

- Screen item
- Screen user
- Screen user group

Source

CScreen::create() in *ui/include/classes/api/services/CScreen.php*.

screen.delete

Description

`object screen.delete(array screenIds)`

This method allows to delete screens.

Parameters

`(array)` IDs of the screens to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted screens under the `screenids` property.

Examples

Deleting multiple screens

Delete two screens.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screen.delete",
    "params": [
        "25",
        "26"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
```

```
            "25",
            "26"
        ]
    },
    "id": 1
}
```

Source

CScreen::delete() in *ui/include/classes/api/services/CScreen.php*.

screen.get

Description

`integer/array screen.get(object parameters)`

The method allows to retrieve screens according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
screenids	string/array	Return only screens with the given IDs.
userids	string/array	Return only screens that belong to the given user IDs.
screenitemids	string/array	Return only screens that contain the given screen items.
selectScreenItems	query	Return a screenitems property with the elements that are used in the screen.
selectUsers	query	Return a users property with users that the screen is shared with.
selectUserGroups	query	Return a userGroups property with user groups that the screen is shared with.
sortfield	string/array	Sort the result by the given properties. Possible values are: `screenid` and `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 a screen by ID

Retrieve all data about screen "26" and its screen items.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screen.get",
    "params": {
        "output": "extend",
        "selectScreenItems": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
        "screenids": "26"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "screenitems": [
                {
                    "screenitemid": "67",
                    "screenid": "26",
                    "resourcetype": "0",
                    "resourceid": "612",
                    "width": "320",
                    "height": "200",
                    "x": "0",
                    "y": "0",
                    "colspan": "0",
                    "rowspan": "0",
                    "elements": "25",
                    "valign": "0",
                    "halign": "0",
                    "style": "0",
                    "url": "",
                    "dynamic": "0",
                    "sort_triggers": "0"
                }
            ],
            "users": [
                {
                    "sysmapuserid": "1",
                    "userid": "2",
                    "permission": "2"
                }
            ],
            "userGroups": [
                {
                    "screenusrgrpid": "1",
                    "usrgrpid": "7",
                    "permission": "3"
                }
            ],
            "screenid": "26",
            "name": "CPU Graphs",
            "hsize": "3",
            "vsize": "2",
            "userid": "1",
            "private": "1"
        }
    ],
```

```
    "id": 1
}
```

See also

- Screen item
- Screen user
- Screen user group

Source

CScreen::get() in *ui/include/classes/api/services/CScreen.php*.

screen.update

Description

`object screen.update(object/array screens)`

This method allows to update existing screens.

Parameters

`(object/array)` Screen properties to be updated.

The `screenid` property must be defined for each screen, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard screen properties, the method accepts the following parameters.

Parameter	Type	Description
screenitems	array	Screen items to replace existing screen items. Screen items are updated by coordinates, so each screen item must have the `x` and `y` properties defined.
users	array	Screen user shares to replace the existing elements.
userGroups	array	Screen user group shares to replace the existing elements.

Return values

`(object)` Returns an object containing the IDs of the updated screens under the `screenids` property.

Examples

Renaming a screen

Rename a screen to "CPU Graphs".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screen.update",
    "params": {
        "screenid": "26",
        "name": "CPU Graphs"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "26"
        ]
```

```
    },
    "id": 1
}
```

Change screen owner

Available only for admins and super admins.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screen.update",
    "params": {
        "screenid": "83",
        "userid": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "83"
        ]
    },
    "id": 2
}
```

See also

- Screen item
- screenitem.create
- screenitem.update
- screenitem.updatebyposition
- Screen user
- Screen user group

Source

CScreen::update() in *ui/include/classes/api/services/CScreen.php*.

Screen item

This class is designed to work with screen items.

Object references:

- Screen item

Available methods:

- screenitem.create - creating new screen items
- screenitem.delete - deleting screen items
- screenitem.get - retrieving screen items
- screenitem.update - updating screen items
- screenitem.updatebyposition - updating screen items in a specific screen cell

> Screen item object

The following objects are directly related to the `screenitem` API.

Screen item

The screen item object defines an element displayed on a screen. It has the following properties.

Property	Type	Description
screenitemid	string	*(readonly)* ID of the screen item.
resourcetype (required)	integer	Type of screen item. Possible values: 0 - graph; 1 - simple graph; 2 - map; 3 - plain text; 4 - hosts info; 5 - triggers info; 6 - system information; 7 - clock; 9 - triggers overview; 10 - data overview; 11 - URL; 12 - history of actions; 13 - history of events; 14 - latest host group issues; 15 - problems by severity; 16 - latest host issues; 19 - simple graph prototype; 20 - graph prototype.
screenid (required)	string	ID of the screen that the item belongs to.
application	string	Application or part of application name by which data in screen item can be filtered. Applies to resource types: ”Data overview” and ”Triggers overview”.
colspan	integer	Number of columns the screen item will span across. Default: 1.
dynamic	integer	Whether the screen item is dynamic. Possible values: 0 - *(default)* not dynamic; 1 - dynamic.
elements	integer	Number of lines to display on the screen item. Default: 25.
halign	integer	Specifies how the screen item must be aligned horizontally in the cell. Possible values: 0 - *(default)* center; 1 - left; 2 - right.
height	integer	Height of the screen item in pixels. Default: 200.
max_columns	integer	Specifies the maximum amount of columns a graph prototype or simple graph prototype screen element can have. Default: 3.

Property	Type	Description
resourceid	string	ID of the object displayed on the screen item. Depending on the type of a screen item, the `resourceid` property can reference different objects. Required for data overview, graph, map, plain text, simple graph and trigger overview screen items. Unused by local and server time clocks, history of actions, history of events, hosts info, system information, problems by severity and URL screen items.
rowspan	integer	Number or rows the screen item will span across. Default: 1.
sort_triggers	integer	Order in which actions or triggers must be sorted. Possible values for history of actions screen elements: 3 - time, ascending; 4 - time, descending; 5 - type, ascending; 6 - type, descending; 7 - status, ascending; 8 - status, descending; 9 - retries left, ascending; 10 - retries left, descending; 11 - recipient, ascending; 12 - recipient, descending. Possible values for latest host group issues and latest host issues screen items: 0 - *(default)* last change, descending; 1 - severity, descending; 2 - host, ascending.
style	integer	Screen item display option. Possible values for data overview and triggers overview screen items: 0 - *(default)* display hosts on the left side; 1 - display hosts on the top. Possible values for hosts info and triggers info screen elements: 0 - *(default)* horizontal layout; 1 - vertical layout. Possible values for clock screen items: 0 - *(default)* local time; 1 - server time; 2 - host time. Possible values for plain text screen items: 0 - *(default)* display values as plain text; 1 - display values as HTML.
url	string	URL of the webpage to be displayed in the screen item. Used by URL screen items.
valign	integer	Specifies how the screen item must be aligned vertically in the cell. Possible values: 0 - *(default)* middle; 1 - top; 2 - bottom.
width	integer	Width of the screen item in pixels. Default: 320.
x	integer	X-coordinates of the screen item on the screen, from left to right. Default: 0.

Property	Type	Description
y	integer	Y-coordinates of the screen item on the screen, from top to bottom. Default: 0.

screenitem.create

Description

`object screenitem.create(object/array screenItems)`

This method allows to create new screen items.

Parameters

`(object/array)` Screen items to create.

The method accepts screen items with the standard screen item properties.

Return values

`(object)` Returns an object containing the IDs of the created screen items under the `screenitemids` property. The order of the returned IDs matches the order of the passed screen items.

Examples

Creating a screen item

Create a screen item displaying a graph in the left-upper cell of the screen.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screenitem.create",
    "params": {
        "screenid": 16,
        "resourcetype": 0,
        "resourceid": 612,
        "x": 0,
        "y": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenitemids": [
            "65"
        ]
    },
    "id": 1
}
```

See also

- screen.update

Source

CScreenItem::create() in *ui/include/classes/api/services/CScreenItem.php*.

screenitem.delete

Description

`object screenitem.delete(array screenItemIds)`

This method allows to delete screen items.

Parameters

`(array)` IDs of the screen items to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted screen items under the `screenitemids` property.

Examples

Deleting multiple screen items

Delete two screen items.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screenitem.delete",
    "params": [
        "65",
        "63"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenitemids": [
            "65",
            "63"
        ]
    },
    "id": 1
}
```

See also

- screen.update

Source

CScreenItem::delete() in *ui/include/classes/api/services/CScreenItem.php*.

screenitem.get

Description

`integer/array screenitem.get(object parameters)`

The method allows to retrieve screen items according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
screenitemids	string/array	Return only screen items with the given IDs.
screenids	string/array	Return only screen items that belong to the given screen.
sortfield	string/array	Sort the result by the given properties. Possible values are: `screenitemid` and `screenid`.

Parameter	Type	Description
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page 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 screen items from screen

Retrieve all screen items from the given screen.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screenitem.get",
    "params": {
        "output": "extend",
        "screenids": "3"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "screenitemid": "20",
            "screenid": "3",
            "resourcetype": "0",
            "resourceid": "433",
            "width": "500",
            "height": "120",
            "x": "0",
            "y": "0",
            "colspan": "1",
            "rowspan": "1",
            "elements": "0",
            "valign": "1",
            "halign": "0",
            "style": "0",
            "url": "",
            "dynamic": "0",
            "sort_triggers": "0",
            "application": "",
            "max_columns": "3"
```

```
        },
        {
            "screenitemid": "21",
            "screenid": "3",
            "resourcetype": "0",
            "resourceid": "387",
            "width": "500",
            "height": "100",
            "x": "0",
            "y": "1",
            "colspan": "1",
            "rowspan": "1",
            "elements": "0",
            "valign": "1",
            "halign": "0",
            "style": "0",
            "url": "",
            "dynamic": "0",
            "sort_triggers": "0",
            "application": "",
            "max_columns": "3"
        },
        {
            "screenitemid": "22",
            "screenid": "3",
            "resourcetype": "1",
            "resourceid": "10013",
            "width": "500",
            "height": "148",
            "x": "1",
            "y": "0",
            "colspan": "1",
            "rowspan": "1",
            "elements": "0",
            "valign": "1",
            "halign": "0",
            "style": "0",
            "url": "",
            "dynamic": "0",
            "sort_triggers": "0",
            "application": "",
            "max_columns": "3"
        },
        {
            "screenitemid": "23",
            "screenid": "3",
            "resourcetype": "1",
            "resourceid": "22181",
            "width": "500",
            "height": "184",
            "x": "1",
            "y": "1",
            "colspan": "1",
            "rowspan": "1",
            "elements": "0",
            "valign": "1",
            "halign": "0",
            "style": "0",
            "url": "",
            "dynamic": "0",
            "sort_triggers": "0",
            "application": "",
```

```
            "max_columns": "3"
        }
    ],
    "id": 1
}
```

Source

CScreenItem::get() in *ui/include/classes/api/services/CScreenItem.php*.

screenitem.update

Description

`object screenitem.update(object/array screenItems)`

This method allows to update existing screen items.

Parameters

`(object/array)` Screen item properties to be updated.

The `screenitemid` property must be defined for each screen item, 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 screen items under the `screenitemids` property.

Examples

Setting the size of the screen item

Set the width of the screen item to 500px and height to 300px.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screenitem.update",
    "params": {
        "screenitemid": "20",
        "width": 500,
        "height": 300
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenitemids": [
            "20"
        ]
    },
    "id": 1
}
```

See also

- screenitem.updatebyposition

Source

CScreenItem::update() in *ui/include/classes/api/services/CScreenItem.php*.

screenitem.updatebyposition

Description

`object screenitem.updatebyposition(array screenItems)`

This method allows to update screen items in the given screen cells. If a cell is empty, a new screen item will be created.

Parameters

`(array)` Screen item properties to be updated.

The `x`, `y` and `screenid` properties must be defined for each screen item, 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 and created screen items under the `screenitemids` property.

Examples

Changing a screen items resource ID

Change the resource ID for the screen element located in the upper-left cell of the screen.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screenitem.updatebyposition",
    "params": [
        {
            "screenid": "16",
            "x": 0,
            "y": 0,
            "resourceid": "644"
        }
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenitemids": [
            "66"
        ]
    },
    "id": 1
}
```

See also

- screenitem.update

Source

CScreenItem::update() in *ui/include/classes/api/services/CScreenItem.php*.

Script

This class is designed to work with scripts.

Object references:

- Script

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.

Property	Type	Description
scriptid	string	*(readonly)* ID of the script.
command (required)	string	Command to run.
name (required)	string	Name of the script.
confirmation	string	Confirmation pop up text. The pop up will appear when trying to run the script from the Zabbix frontend.
description	string	Description of the script.
execute_on	integer	Where to run the script. Possible values: 0 - run on Zabbix agent; 1 - run on Zabbix server. 2 - *(default)* run on Zabbix server (proxy).
groupid	string	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.
host_access	integer	Host permissions needed to run the script. Possible values: 2 - *(default)* read; 3 - write.
type	integer	Script type. Possible values: 0 - *(default)* script; 1 - IPMI.
usrgrpid	string	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. Default: 0.

Note that for some methods (update, delete) the required/optional parameter combination is different.

script.create

Description

`object script.create(object/array scripts)`

This method allows to create new scripts.

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 script

Create a 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",
        "host_access": 3,
        "confirmation": "Are you sure you would like to reboot the server?"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "3"
        ]
    },
    "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.

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.

Parameters

`(object)` Parameters containing the ID of the script to run and the ID of the host.

Parameter	Type	Description
hostid (required)	string	ID of the host to run the script on.
scriptid (required)	string	ID of the script to run.

Return values

`(object)` Returns the result of script execution.

Property	Type	Description
response	string	Whether the script was run successfully. Possible value - `success`.
value	string	Script output.

Examples

Run a 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.\n64 bytes from 127.0.0.1: icmp_req=1 tt
    },
    "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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
groupids	string/array	Return only scripts that can be run on the given host groups.
hostids	string/array	Return only scripts that can be run on the given hosts.
scriptids	string/array	Return only scripts with the given IDs.
usrgrpids	string/array	Return only scripts that can be run by users in the given user groups.
selectGroups	query	Return a groups property with host groups that the script can be run on.
selectHosts	query	Return a hosts property with hosts that the script can be run on.
sortfield	string/array	Sort the result by the given properties. Possible values are: `scriptid` and `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
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

Retrieve all scripts

Retrieve all configured scripts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.get",
    "params": {
        "output": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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"
        },
        {
            "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"
        },
        {
            "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"
        }
    ],
    "id": 1
}
```

See also

- Host
- Host group

Source

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.

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.

> **Note:**
> 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:

```
{
    "jsonrpc": "2.0",
    "method": "script.getscriptsbyhosts",
    "params": [
        "30079",
        "30073"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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": "",
            "type": "0",
            "execute_on": "1",
            "hostid": "10001"
        },
        {
            "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",
            "hostid": "10001"
        }
    ],
    "30073": [
        {
            "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": "",
            "type": "0",
            "execute_on": "1",
            "hostid": "10001"
        },
        {
            "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",
```

```
                "hostid": "10001"
            }
        ]
    },
    "id": 1
}
```

Source

CScript::getScriptsByHosts() in *ui/include/classes/api/services/CScript.php*.

script.update

Description

`object script.update(object/array scripts)`

This method allows to update existing scripts.

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.

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": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "1"
        ]
    },
    "id": 1
}
```

Source

CScript::update() in *ui/include/classes/api/services/CScript.php*.

Service

This class is designed to work with services.

Object references:

- Service
- Service time
- Service dependency
- Service alarm

Available methods:

- service.adddependencies - adding dependencies between IT services
- service.addtimes - adding service times
- service.create - creating new IT services
- service.delete - deleting IT services
- service.deletedependencies - deleting dependencies between IT services
- service.deletetimes - deleting service times
- service.get - retrieving IT services
- service.getsla - retrieving availability information about IT services
- service.update - updating IT services

> Service object

The following objects are directly related to the `service` API.

Service

The service object has the following properties.

Property	Type	Description
serviceid	string	*(readonly)* ID of the service.
algorithm (required)	integer	Algorithm used to calculate the state of the service. Possible values: 0 - do not calculate; 1 - problem, if at least one child has a problem; 2 - problem, if all children have problems.
name (required)	string	Name of the service.
showsla (required)	integer	Whether SLA should be calculated. Possible values: 0 - do not calculate; 1 - calculate.
sortorder (required)	integer	Position of the service used for sorting.
goodsla	float	Minimum acceptable SLA value. If the SLA drops lower, the service is considered to be in problem state. Default: 99.9.
status	integer	*(readonly)* Whether the service is in OK or problem state. If the service is in problem state, `status` is equal either to: - the priority of the linked trigger if it is set to 2, ”Warning” or higher (priorities 0, ”Not classified” and 1, ”Information” are ignored); - the highest status of a child service in problem state. If the service is in OK state, `status` is equal to 0.
triggerid	string	Trigger associated with the service. Can only be set for services that don’t have children. Default: 0

Note that for some methods (update, delete) the required/optional parameter combination is different.

Service time

The service time object defines periods, when an service is scheduled to be up or down. It has the following properties.

Property	Type	Description
timeid	string	*(readonly)* ID of the service time.
serviceid (required)	string	ID of the service. Cannot be updated.
ts_from (required)	integer	Time when the service time comes into effect. For onetime downtimes `ts_from` must be set as a Unix timestamp, for other types - as a specific time in a week, in seconds, for example, 90000 for Mon, 1:00 AM.
ts_to (required)	integer	Time when the service time ends. For onetime uptimes `ts_to` must be set as a Unix timestamp, for other types - as a specific time in a week, in seconds, for example, 90000 for Mon, 1:00 AM.
type (required)	integer	Service time type. Possible values: 0 - planned uptime, repeated every week; 1 - planned downtime, repeated every week; 2 - one-time downtime.
note	string	Additional information about the service time.

Service dependency

The service dependency object represents a dependency between services. It has the following properties.

Property	Type	Description
linkid	string	*(readonly)* ID of the service dependency.
servicedownid (required)	string	ID of the service, that a service depends on, that is, the child service. A service can have multiple children.
serviceupid (required)	string	ID of the service, that is dependent on a service, that is, the parent service. A service can have multiple parents forming a directed graph.
soft (required)	integer	Type of dependency between services. Possible values: 0 - hard dependency; 1 - soft dependency. A service can have only one hard-dependent parent. This attribute has no effect on status or SLA calculation and is only used to create a core service tree. Additional parents can be added as soft dependencies forming a graph. A service can not be deleted if it has hard-dependent children.

Service alarm

Note:
Service alarms cannot be directly created, updated or deleted via the Zabbix API.

The service alarm objects represents an service's state change. It has the following properties.

Property	Type	Description
servicealarmid	string	ID of the service alarm.
serviceid	string	ID of the service.

Property	Type	Description
clock	timestamp	Time when the service state change has happened.
value	integer	Status of the service. Refer the service status property for a list of possible values.

service.adddependencies

Description

`object service.adddependencies(object/array serviceDependencies)`

This method allows to create dependencies between services.

Parameters

`(object/array)` Service dependencies to create.

Each service dependency has the following parameters.

Parameter	Type	Description
serviceid	string	ID of the service that depends on a service, that is, the parent service.
dependsOnServiceid	string	ID of the service that a service depends on, that is, the child service.
soft	string	Type of dependency. Refer to the service dependency object page for more information on dependency types.

Return values

`(object)` Returns an object containing the IDs of the affected parent services under the `serviceids` property.

Examples

Creating a hard dependency

Make service "2" a hard-dependent child of service "3".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.adddependencies",
    "params": {
        "serviceid": "3",
        "dependsOnServiceid": "2",
        "soft": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- service.update

Source

CService::addDependencies() in *ui/include/classes/api/services/CService.php*.

service.addtimes

Description

`object service.addtimes(object/array serviceTimes)`

This method allows to create new service times.

Parameters

`(object/array)` Service times to create.

The method accepts service times with the standard service time properties.

Return values

`(object)` Returns an object containing the IDs of the affected services under the `serviceids` property.

Examples

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.addtimes",
    "params": {
        "serviceid": "4",
        "type": 1,
        "ts_from": 165600,
        "ts_to": 201600
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "4"
        ]
    },
    "id": 1
}
```

See also

- service.update

Source

CService::addTimes() in *ui/include/classes/api/services/CService.php*.

service.create

Description

`object service.create(object/array services)`

This method allows to create new services.

Parameters

(object/array) services to create.

Additionally to the standard service properties, the method accepts the following parameters.

Parameter	Type	Description
dependencies	array	Service dependencies. Each service dependency has the following parameters: - dependsOnServiceid - *(string)* ID of an service the service depends on, that is, the child service. - soft - *(integer)* type of service dependency.
parentid	string	ID of a hard-linked parent service.
times	array	Service times 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 an service

Create an service that will be switched to problem state, if at least one child has a problem. SLA calculation will be on and the minimum acceptable SLA is 99.99%.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.create",
    "params": {
        "name": "Server 1",
        "algorithm": 1,
        "showsla": 1,
        "goodsla": 99.99,
        "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.

Services with hard-dependent child services cannot be deleted.

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:

```
{
    "jsonrpc": "2.0",
    "method": "service.delete",
    "params": [
        "4",
        "5"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "4",
            "5"
        ]
    },
    "id": 1
}
```

Source

CService::delete() in *ui/include/classes/api/services/CService.php*.

service.deletedependencies

Description

`object service.deletedependencies(string/array serviceIds)`

This method allows to delete all dependencies from services.

Parameters

`(string/array)` IDs of the services to delete all dependencies from.

Return values

`(object)` Returns an object containing the IDs of the affected services under the `serviceids` property.

Examples

Deleting dependencies from an service

Delete all dependencies from service ”2”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.deletedependencies",
    "params": [
        "2"
```

```
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "2"
        ]
    },
    "id": 1
}
```

See also

- service.update

Source

CService::delete() in *ui/include/classes/api/services/CService.php*.

service.deletetimes

Description

`object service.deletetimes(string/array serviceIds)`

This method allows to delete all service times from services.

Parameters

`(string/array)` IDs of the services to delete all service times from.

Return values

`(object)` Returns an object containing the IDs of the affected services under the `serviceids` property.

Examples

Deleting service times from an service

Delete all service times from service "2".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.deletetimes",
    "params": [
        "2"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "2"
        ]
    },
    "id": 1
}
```

See also

- service.update

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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
serviceids	string/array	Return only services with the given IDs.
parentids	string/array	Return only services with the given hard-dependent parent services.
childids	string/array	Return only services that are hard-dependent on the given child services.
selectParent	query	Return a parent property with the hard-dependent parent service.
selectDependencies	query	Return a dependencies property with child service dependencies.
selectParentDependencies	query	Return a parentDependencies property with parent service dependencies.
selectTimes	query	Return a times property with service times.
selectAlarms	query	Return an alarms property with service alarms.
selectTrigger	query	Return a trigger property with the associated trigger.
sortfield	string/array	Sort the result by the given properties. Possible values are: `name` and `sortorder`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
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 all services

Retrieve all data about all services and their dependencies.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.get",
    "params": {
        "output": "extend",
        "selectDependencies": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "serviceid": "2",
            "name": "Server 1",
            "status": "0",
            "algorithm": "1",
            "triggerid": "0",
            "showsla": "1",
            "goodsla": "99.9000",
            "sortorder": "0",
            "dependencies": []
        },
        {
            "serviceid": "3",
            "name": "Data center 1",
            "status": "0",
            "algorithm": "1",
            "triggerid": "0",
            "showsla": "1",
            "goodsla": "99.9000",
            "sortorder": "0",
            "dependencies": [
                {
                    "linkid": "11",
                    "serviceupid": "3",
                    "servicedownid": "2",
                    "soft": "0",
                    "sortorder": "0",
                    "serviceid": "2"
                },
                {
                    "linkid": "10",
                    "serviceupid": "3",
                    "servicedownid": "5",
                    "soft": "0",
                    "sortorder": "1",
                    "serviceid": "5"
                }
            ]
        },
        {
            "serviceid": "5",
            "name": "Server 2",
            "status": "0",
            "algorithm": "1",
            "triggerid": "0",
            "showsla": "1",
            "goodsla": "99.9900",
```

```
            "sortorder": "1",
            "dependencies": []
        }
    ],
    "id": 1
}
```

Source

CService::get() in *ui/include/classes/api/services/CService.php*.

service.getsla

Description

`object service.getsla(object parameters)`

This method allows to calculate availability information about services.

Parameters

`(object)` Parameters containing the IDs of the services and time intervals to calculate SLA.

Parameter	Type	Description
serviceids	string/array	IDs of services to return availability information for.
intervals	array	Time intervals to return service layer availability information about. Each time interval must have the following parameters: - `from` - *(timestamp)* interval start time; - `to` - *(timestamp)* interval end time.

Return values

`(object)` Returns the following availability information about each service under the corresponding service ID.

Property	Type	Description
status	integer	Current status of the service. Refer to the service object page for more information on service statuses.
problems	array	Triggers that are currently in problem state and are linked either to the service or one of its descendants.
sla	array	SLA data about each time period. Each SLA object has the following properties: - `from` - *(timestamp)* interval start time; - `to` - *(timestamp)* interval end time; - `sla` - *(float)* SLA for the given time interval; - `okTime` - *(integer)* time the service was in OK state, in seconds; - `problemTime` - *(integer)* time the service was in problem state, in seconds; - `downtimeTime` - *(integer)* time the service was in scheduled downtime, in seconds.

Examples

Retrieving availability information for an service

Retrieve availability information about a service during a week.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.getsla",
    "params": {
        "serviceids": "2",
        "intervals": [
            {
                "from": 1352452201,
                "to": 1353057001
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "2": {
            "status": "3",
            "problems": {
                "13904": {
                    "triggerid": "13904",
                    "expression": "{13359}=0",
                    "description": "Service unavailable",
                    "url": "",
                    "status": "0",
                    "value": "1",
                    "priority": "3",
                    "lastchange": "1352967420",
                    "comments": "",
                    "error": "",
                    "templateid": "0",
                    "type": "0",
                    "value_flags": "0",
                    "flags": "0"
                }
            },
            "sla": [
                {
                    "from": 1352452201,
                    "to": 1353057001,
                    "sla": 97.046296296296,
                    "okTime": 586936,
                    "problemTime": 17864,
                    "downtimeTime": 0
                }
            ]
        }
    },
    "id": 1
}
```

See also

- Trigger

Source

CService::getSla() in *ui/include/classes/api/services/CService.php*.

service.update

Description

`object service.update(object/array services)`

This method allows to update existing services.

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.

Parameter	Type	Description
dependencies	array	Service dependencies to replace the current service dependencies. Each service dependency has the following parameters: - `dependsOnServiceid` - *(string)* ID of an service the service depends on, that is, the child service. - `soft` - *(integer)* type of service dependency; refer to the service dependency object page for more information on dependency types.
parentid	string	ID of a hard-linked parent service.
times	array	Service times to replace the current service times.

Return values

`(object)` Returns an object containing the IDs of the updated services under the `serviceids` property.

Examples

Setting the parent of an service

Make service "3" the hard-linked parent of service "5".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.update",
    "params": {
        "serviceid": "5",
        "parentid": "3"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "5"
        ]
    },
    "id": 1
}
```

See also

- service.adddependencies
- service.addtimes
- service.deletedependencies
- service.deletetimes

Source

CService::update() in *ui/include/classes/api/services/CService.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
- 'Check now' request object
- 'Diagnostic information' request object
- Statistic request object
- Statistic result object

Available methods:

- task.create - creating new tasks
- task.get - retrieving tasks

> Task object

The following objects are directly related to the `task` API.

<note note>Task object described in this page is supported since Zabbix version 5.0.5. For proper usage of `task.create` in older Zabbix versions, please see documentation for task.create. :::

The task object has the following properties:

Property	Type	Description
taskid	string	*(readonly)* ID of the task.
type (required)	integer	Type of the task. Possible values: 1 - Diagnostic information; 6 - Check now.
status	integer	*(readonly)* Status of the task. Possible values: 1 - new task; 2 - task in progress; 3 - task is completed; 4 - task is expired.
clock	timestamp	*(readonly)* Time when the task was created.
ttl	integer	*(readonly)* The time in seconds after which task expires.
proxy_hostid	string	ID of the proxy about which diagnostic information statistic is collected. Ignored for 'Check now' tasks.
request (required)	object	Task request object according to the task type: Object of 'Check now' task is described in detail below; Object of 'Diagnostic information' task is described in detail below.
result	object	*(readonly)* Result object of the diagnostic information task. May contain NULL if result is not yet ready. Result object is described in detail below.

'Check now' request object

The 'Check now' task request object has the following properties.

Property	Type	Description
itemid	string	ID of item and low-level discovery rules.

'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.

Property	Type	Description
historycache	object	History cache statistic request. Available on server and proxy.
valuecache	object	Items cache statistic request. Available on server.
preprocessing	object	Preprocessing manager statistic request. Available on server and proxy.
alerting	object	Alert manager statistic request. Available on server.
lld	object	LLD manager statistic request. Available on server.

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.

Property	Type	Description
stats	query	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.
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: { "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.

Diagnostic type	Available fields	Description
historycache	items	Number of cached items.
	values	Number of cached values.
	memory	Shared memory statistics (free space, number of used chunks, number of free chunks, max size of free chunk).
	memory.data	History data cache shared memory statistics.
	memory.index	History index cache shared memory statistics.
valuecache	items	Number of cached items.
	values	Number of cached values.
	memory	Shared memory statistics (free space, number of used chunks, number of free chunks, max size of free chunk).
	mode	Value cache mode.
preprocessing	values	Number of queued values.
	preproc.values	Number of queued values with preprocessing steps.
alerting	alerts	Number of queued alerts.
lld	rules	Number of queued rules.
	values	Number of queued values.

List of sorting fields available for each type of diagnostic information request

Following statistic fields can be used to sort and limit requested information.

Diagnostic type	Available fields	Type
historycache	values	integer
valuecache	values	integer
	request.values	integer
preprocessing	values	integer
alerting	media.alerts	integer
	source.alerts	integer
lld	values	integer

Statistic result object

Statistic result object is retrieved in `result` field of task object.

Property	Type	Description
status	integer	*(readonly)* Status of the task result. Possible values: -1 - error occurred during performing task; 0 - task result is created.
data	string/object	Results according the statistic request object of particular diagnostic information task. Contains error message string if error occurred during performing task.

task.create

Description

`object task.create(object task)`

This method allows to create a new task (such as collect diagnostic data or check items or low-level discovery rules without config reload).

<note note>Starting from Zabbix version 5.0.5, `task.create` accepts array of task objects `object/array tasks`, allowing to create multiple tasks in single request. :::

Parameters

`(object)` A task to create.

The method accepts the following parameters. Please use this format for Zabbix versions before 5.0.5 only.

Parameter	Type	Description
type (required)	integer	Task type. Possible values: 6 - Check now.
itemids (required)	string/array	IDs of items and low-level discovery rules.

Please be aware that starting from Zabbix version 5.0.5, request format has changed:

Parameter	Type	Description
type (required)	integer	Task type. Possible values: 1 - Diagnostic information; 6 - Check now.
request (required)	object	Task request object according to the task type. Correct format of request object is described in Task object section.

Parameter	Type	Description
proxy_hostid	integer	Proxy about which Diagnostic information task will collect data. Ignored for Check now tasks.

Note that 'Check now' tasks can be created for the following types of items/discovery rules:

- Zabbix agent
- SNMPv1/v2/v3 agent
- Simple check
- Internal check
- Aggregate 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` or task objects passed as array.

Examples

Creating a task

Create a task `check now` for two items. One is an item, the other is a low-level discovery rule.

Request for Zabbix versions before 5.0.5:

```
{
    "jsonrpc": "2.0",
    "method": "task.create",
    "params": {
        "type": "6",
        "itemids": [
            "10092",
            "10093"
        ],
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Same request for Zabbix versions starting from 5.0.5:

```
{
    "jsonrpc": "2.0",
    "method": "task.create",
    "params": [
        {
            "type": "6",
            "request": {
                "itemid": "10092"
            }
        },
        {
            "type": "6",
            "request": {
                "itemid": "10093"
            }
```

```
        }
    ],
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response is similar for both example requests:

```
{
    "jsonrpc": "2.0",
    "result": {
        "taskids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

Create a task `diagnostic information` task. Request:

```
{
    "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:

```
{
    "jsonrpc": "2.0",
    "result": {
        "taskids": [
            "3"
        ]
    },
    "id": 2
}
```

See also

- Task
- 'Check 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.

<note note>This method is available since Zabbix version 5.0.5. :::

For for non-Super Admins users, method returns an insufficient permission message.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
taskids	string/array	Return only tasks with the given IDs.
output	query	These parameters being common for all `get` methods are described in detail in the reference commentary.
preservekeys	boolean	

Return values

`(integer/array)` Returns an array of objects.

Examples

Retrieve task by ID

Retrieve all data about task "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
                    }
                }
            },
            "result": {
                "data": {
                    "alerting": {
                        "alerts": 0,
                        "top": {
                            "media.alerts": []
                        },
                        "time": 0.000663
                    },
                    "lld": {
                        "rules": 0,
                        "values": 0,
                        "top": {
                            "values": []
                        },
                        "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.

Property	Type	Description
templateid	string	*(readonly)* ID of the template.
host (required)	string	Technical name of the template.
description	text	Description of the template.
name	string	Visible name of the template. Default: `host` property value.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Template tag

The template tag object has the following properties.

Property	Type	Description
tag (required)	string	Template tag name.
value	string	Template tag value.

template.create

Description

`object template.create(object/array templates)`

This method allows to create new templates.

Parameters

`(object/array)` Templates to create.

Additionally to the standard template properties, the method accepts the following parameters.

Parameter	Type	Description
groups (required)	object/array	Host groups to add the template to. The host groups must have the `groupid` property defined.
tags	object/array	Template tags.
templates	object/array	Templates to be linked to the template. The templates must have the `templateid` property defined.
macros	object/array	User macros to be created for the template.
hosts	object/array	Hosts to link the template to. The hosts must have the `hostid` property defined.

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 it to two hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.create",
    "params": {
        "host": "Linux template",
        "groups": {
            "groupid": 1
        },
        "hosts": [
            {
                "hostid": "10084"
            },
            {
                "hostid": "10090"
            }
        ],
        "tags": [
            {
                "tag": "Host name",
                "value": "{HOST.NAME}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086"
        ]
    },
    "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.

> **Note:**
> 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.

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.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
templateids	string/array	Return only templates with the given template IDs.
groupids	string/array	Return only templates that belong to the given host groups.
parentTemplateids	string/array	Return only templates that are parent to the given templates.
hostids	string/array	Return only templates that are linked to the given hosts/templates.
graphids	string/array	Return only templates that contain the given graphs.
itemids	string/array	Return only templates that contain the given items.
triggerids	string/array	Return only templates that contain the given triggers.
with_items	flag	Return only templates that have items.
with_triggers	flag	Return only templates that have triggers.
with_graphs	flag	Return only templates that have graphs.
with_httptests	flag	Return only templates that have web scenarios.
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.

Parameter	Type	Description
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. Possible operator values: 0 - (default) Contains; 1 - Equals.
selectGroups	query	Return the host groups that the template belongs to in the groups property.
selectTags	query	Return template tags in the tags property.
selectHosts	query	Return the hosts that are linked to the template in the hosts property. Supports `count`.
selectTemplates	query	Return templates to which the template is a child, in the `templates` property. Supports `count`.
selectParentTemplates	query	Return templates to which the template is a parent, in the `parentTemplates` property. Supports `count`.
selectHttpTests	query	Return the web scenarios from the template in the httpTests property. Supports `count`.
selectItems	query	Return items from the template in the items property. Supports `count`.
selectDiscoveries	query	Return low-level discoveries from the template in the `discoveries` property. Supports `count`.
selectTriggers	query	Return triggers from the template in the triggers property. Supports `count`.
selectGraphs	query	Return graphs from the template in the graphs property. Supports `count`.
selectApplications	query	Return applications from the template in the applications property. Supports `count`.
selectMacros	query	Return the macros from the template in the `macros` property..
selectScreens	query	Return screens from the template in the screens property. Supports `count`.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectTemplates` - results will be sorted by `name`; `selectHosts` - sorted by `host`; `selectParentTemplates` - sorted by `host`; `selectItems` - sorted by `name`; `selectDiscoveries` - sorted by `name`; `selectTriggers` - sorted by `description`; `selectGraphs` - sorted by `name`; `selectApplications` - sorted by `name`; `selectScreens` - sorted by `name`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `hostid`, `host`, `name`, `status`.

Parameter	Type	Description
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
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 templates by name

Retrieve all data about two templates named "Template OS Linux" and "Template OS Windows".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": "extend",
        "filter": {
            "host": [
                "Template OS Linux",
                "Template OS Windows"
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "proxy_hostid": "0",
            "host": "Template OS Linux",
            "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": "Template OS Linux",
            "flags": "0",
            "templateid": "10001",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": ""
        },
        {
            "proxy_hostid": "0",
            "host": "Template OS 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": "Template OS Windows",
            "flags": "0",
            "templateid": "10081",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
```

```
            "tls_psk_identity": "",
            "tls_psk": ""
        }
    ],
    "id": 1
}
```

Searching by template tags

Retrieve templates that have tag "Host name" equal to "{HOST.NAME}".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": ["hostid"],
        "selectTags": "extend",
        "evaltype": 0,
        "tags": [
            {
                "tag": "Host name",
                "value": "{HOST.NAME}",
                "operator": 1
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10402",
            "tags": [
                {
                    "tag": "Host name",
                    "value": "{HOST.NAME}"
                }
            ]
        }
    ],
    "id": 1
}
```

See also

- Host 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.

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.

Parameter	Type	Description
templates (required)	object/array	Templates to be updated. The templates must have the `templateid` property defined.
groups	object/array	Host groups to add the given templates to. The host groups must have the `groupid` property defined.
hosts	object/array	Hosts and templates to link the given templates to. The hosts must have the `hostid` property defined.
macros	object/array	User macros to be created for the given templates.
templates_link	object/array	Templates to link to 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

Adding templates to a group

Add two templates to the host group ”2”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.massadd",
    "params": {
        "templates": [
            {
                "templateid": "10085"
            },
            {
                "templateid": "10086"
            }
        ],
        "groups": [
            {
                "groupid": "2"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
```

```
    "id": 1
}
```

Linking a template to hosts

Link template "10073" to two hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.massadd",
    "params": {
        "templates": [
            {
                "templateid": "10073"
            }
        ],
        "hosts": [
            {
                "hostid": "10106"
            },
            {
                "hostid": "10104"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10073"
        ]
    },
    "id": 1
}
```

See also

- template.update
- Host
- Host 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.

Parameters

`(object)` Parameters containing the IDs of the templates to update and the objects that should be removed.

Parameter	Type	Description
templateids (required)	string/array	IDs of the templates to be updated.
groupids	string/array	Host groups to remove the given templates from.
hostids	string/array	Hosts or templates to unlink the given templates from (downstream).
macros	string/array	User macros to delete from the given templates.
templateids_clear	string/array	Templates to unlink and clear from the given templates (upstream).
templateids_link	string/array	Templates to unlink from the given templates (upstream).

Return values

`(object)` 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 template ”10085” from two hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.massremove",
    "params": {
        "templateids": "10085",
        "hostids": [
            "10106",
            "10104"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

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.

Parameters

`(object)` Parameters containing the IDs of the templates to update and the properties that should be updated.

Additionally to the standard template properties, the method accepts the following parameters.

Parameter	Type	Description
templates (required)	object/array	Templates to be updated. The templates must have the `templateid` property defined.
groups	object/array	Host groups to replace the current host groups the templates belong to. The host groups must have the `groupid` property defined.
hosts	object/array	Hosts and templates to replace the ones the templates are currently linked to. Both hosts and templates must use the `hostid` property to pass an ID.
macros	object/array	User macros to replace the current user macros on the given templates.
templates_clear	object/array	Templates to unlink and clear from the given templates. The templates must have the `templateid` property defined.
templates_link	object/array	Templates to replace the currently linked 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

Replacing host groups

Unlink and clear template ”10091” from the given templates.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.massupdate",
    "params": {
        "templates": [
            {
                "templateid": "10085"
            },
            {
                "templateid": "10086"
            }
        ],
        "templates_clear": [
            {
                "templateid": "10091"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
    "id": 1
}
```

See also

- template.update
- template.massadd
- Host 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.

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	Host groups to replace the current host groups the templates belong to. The host groups must have the `groupid` property defined.

Parameter	Type	Description
tags	object/array	Template tags to replace the current template tags.
hosts	object/array	Hosts and templates to replace the ones the templates are currently linked to. Both hosts and templates must use the `hostid` property to pass an ID.
macros	object/array	User macros to replace the current user macros on the given templates.
templates	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:

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": {
        "templateid": "10086",
        "name": "Template OS Linux"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086"
        ]
    },
    "id": 1
}
```

Updating template tags

Replace all template tags with a new one.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": {
        "templateid": "10086",
        "tags": [
            {
                "tag": "Host name",
                "value": "{HOST.NAME}"
            }
```

```
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086"
        ]
    },
    "id": 1
}
```

Source

CTemplate::update() in *ui/include/classes/api/services/CTemplate.php*.

Template screen

This class is designed to work with template screens.

Object references:

- Template screen

Available methods:

- templatescreen.copy - copy template screens
- templatescreen.create - create new template screens
- templatescreen.delete - delete template screens
- templatescreen.get - retrieve template screens
- templatescreen.update - update template screens

> Template screen object

The following objects are directly related to the `templatescreen` API.

Template screen

The template screen object has the following properties.

Property	Type	Description
screenid	string	*(readonly)* ID of the template screen.
name (required)	string	Name of the template screen.
templateid (required)	string	ID of the template that the screen belongs to.
hsize	integer	Width of the template screen. Default: 1
vsize	integer	Height of the template screen. Default: 1

templatescreen.copy

Description

`object templatescreen.copy(object parameters)`

This method allows to copy template screens to the given templates.

Parameters

`(object)` Parameters defining the template screens to copy and the target templates.

Parameter	Type	Description
screenIds (required)	string/array	IDs of template screens to copy.
templateIds (required)	string/array	IDs of templates to copy the screens to.

Return values

`(boolean)` Returns `true` if the copying was successful.

Examples

Copy a template screen

Copy template screen "25" to template "30085".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreen.copy",
    "params": {
        "screenIds": "25",
        "templateIds": "30085"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CTemplateScreen::copy() in *ui/include/classes/api/services/CTemplateScreen.php*.

templatescreen.create

Description

`object templatescreen.create(object/array templateScreens)`

This method allows to create new template screens.

Parameters

`(object/array)` Template screens to create.

Additionally to the standard template screen properties, the method accepts the following parameters.

Parameter	Type	Description
screenitems	array	Template screen items to create on the screen.

Return values

(object) Returns an object containing the IDs of the created template screens under the screenids property. The order of the returned IDs matches the order of the passed template screens.

Examples

Create a template screen

Create a template screen named “Graphs” with 2 rows and 3 columns and add a graph to the upper-left cell.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreen.create",
    "params": {
        "name": "Graphs",
        "templateid": "10047",
        "hsize": 3,
        "vsize": 2,
        "screenitems": [
            {
                "resourcetype": 0,
                "resourceid": "410",
                "x": 0,
                "y": 0
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "45"
        ]
    },
    "id": 1
}
```

See also

- Template screen item

Source

CTemplateScreen::create() in *ui/include/classes/api/services/CTemplateScreen.php*.

templatescreen.delete

Description

`object templatescreen.delete(array templateScreenIds)`

This method allows to delete template screens.

Parameters

(array) IDs of the template screens to delete.

Return values

(object) Returns an object containing the IDs of the deleted template screens under the screenids property.

Examples

Delete multiple template screens

Delete two template screens.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreen.delete",
    "params": [
        "45",
        "46"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "45",
            "46"
        ]
    },
    "id": 1
}
```

Source

CTemplateScreen::delete() in *ui/include/classes/api/services/CTemplateScreen.php*.

templatescreen.get

Description

`integer/array templatescreen.get(object parameters)`

The method allows to retrieve template screens according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
hostids	string/array	Return only template screens that belong to the given hosts.
screenids	string/array	Return only template screens with the given IDs.
screenitemids	string/array	Return only template screens that contain the given screen items.
templateids	string/array	Return only template screens that belong to the given templates.
noInheritance	flag	Do not return inherited template screens.
selectScreenItems	query	Return the screen items that are used in the template screen in the screenitems property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `screenid` and `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	

Parameter	Type	Description
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

Retrieve screens from template

Retrieve all screens from template "10001" and all of the screen items.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreen.get",
    "params":  {
        "output": "extend",
        "selectScreenItems": "extend",
        "templateids": "10001"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "screenid": "3",
            "name": "System performance",
            "hsize": "2",
            "vsize": "2",
            "templateid": "10001",
            "screenitems": [
                {
                    "screenitemid": "20",
                    "screenid": "3",
                    "resourcetype": "0",
                    "resourceid": "433",
                    "width": "500",
                    "height": "120",
                    "x": "0",
                    "y": "0",
                    "colspan": "1",
                    "rowspan": "1",
                    "elements": "0",
                    "valign": "1",
                    "halign": "0",
                    "style": "0",
                    "url": ""
                },
                {
                    "screenitemid": "21",
                    "screenid": "3",
                    "resourcetype": "0",
                    "resourceid": "387",
```

```
                    "width": "500",
                    "height": "100",
                    "x": "0",
                    "y": "1",
                    "colspan": "1",
                    "rowspan": "1",
                    "elements": "0",
                    "valign": "1",
                    "halign": "0",
                    "style": "0",
                    "url": ""
                },
                {
                    "screenitemid": "22",
                    "screenid": "3",
                    "resourcetype": "1",
                    "resourceid": "10013",
                    "width": "500",
                    "height": "148",
                    "x": "1",
                    "y": "0",
                    "colspan": "1",
                    "rowspan": "1",
                    "elements": "0",
                    "valign": "1",
                    "halign": "0",
                    "style": "0",
                    "url": ""
                },
                {
                    "screenitemid": "23",
                    "screenid": "3",
                    "resourcetype": "1",
                    "resourceid": "22181",
                    "width": "500",
                    "height": "184",
                    "x": "1",
                    "y": "1",
                    "colspan": "1",
                    "rowspan": "1",
                    "elements": "0",
                    "valign": "1",
                    "halign": "0",
                    "style": "0",
                    "url": ""
                }
            ]
        }
    ],
    "id": 1
}
```


See also

- Template screen item

Source

CTemplateScreen::get() in *ui/include/classes/api/services/CTemplateScreen.php*.

templatescreen.update

Description

`object templatescreen.update(object/array templateScreens)`

This method allows to update existing template screens.

Parameters

`(object/array)` Template screen properties to be updated.

The `screenid` property must be defined for each template screen, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard template screen properties, the method accepts the following parameters.

Parameter	Type	Description
screenitems	array	Template screen items to replace existing screen items. Screen items are updated by coordinates, so each screen item must have the `x` and `y` properties defined.

Return values

`(object)` Returns an object containing the IDs of the updated template screens under the `screenids` property.

Examples

Rename a template screen

Rename the template screen to ”Performance graphs”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreen.update",
    "params": {
        "screenid": "3",
        "name": "Performance graphs"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "3"
        ]
    },
    "id": 1
}
```

Source

CTemplateScreen::update() in *ui/include/classes/api/services/CTemplateScreen.php*.

Template screen item

This class is designed to work with template screen items.

Object references:

- Template screen item

Available methods:

- templatescreenitem.get - retrieve template screen items

> Template screen item object

The following objects are directly related to the `templatescreenitem` API.

Template screen item

The template screen item object defines an element displayed on a template screen. It has the following properties.

Property	Type	Description
screenitemid	string	*(readonly)* ID of the template screen item.
resourceid (required)	string	ID of the object from the parent template displayed on the template screen item. Depending on the type of screen item, the `resourceid` property can reference different objects. Unused by clock and URL template screen items. *Note: the `resourceid` property always references an object used in the parent template object, even if the screen item itself is inherited on a host or template.*
resourcetype (required)	integer	Type of template screen item. Possible values: 0 - graph; 1 - simple graph; 3 - plain text; 7 - clock; 11 - URL; 19 - simple graph prototype; 20 - graph prototype.
screenid (required)	string	ID of the template screen that the item belongs to.
colspan	integer	Number of columns the template screen item will span across. Default: 1.
elements	integer	Number of lines to display on the template screen item. Default: 25.
halign	integer	Specifies how the template screen item must be aligned horizontally in the cell. Possible values: 0 - *(default)* center; 1 - left; 2 - right.
height	integer	Height of the template screen item in pixels. Default: 200.
max_columns	integer	Specifies the maximum amount of columns a graph prototype or simple graph prototype screen element can have. Default: 3.
rowspan	integer	Number or rows the template screen item will span across. Default: 1.
style	integer	Template screen item display option. Possible values for clock screen items: 0 - *(default)* local time; 1 - server time; 2 - host time. Possible values for plain text screen items: 0 - *(default)* display values as plain text; 1 - display values as HTML.

Property	Type	Description
url	string	URL of the webpage to be displayed in the template screen item. Used by URL template screen items.
valign	integer	Specifies how the template screen item must be aligned vertically in the cell. Possible values: 0 - *(default)* middle; 1 - top; 2 - bottom.
width	integer	Width of the template screen item in pixels. Default: 320.
x	integer	X-coordinates of the template screen item on the screen, from left to right. Default: 0.
y	integer	Y-coordinates of the template screen item on the screen, from top to bottom. Default: 0.

templatescreenitem.get

Description

`integer/array templatescreenitem.get(object parameters)`

The method allows to retrieve template screen items according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
screenids	string/array	Return only template screen items that belong to the given template screens.
screenitemids	string/array	Return only template screen items with the given IDs.
hostids	string/array	Returns an additional `real_resourceid` property for each template screen item, that belongs to a screen from the given hosts or templates. The `real_resourceid` property contains the ID of object displayed on the screen.
sortfield	string/array	Sort the result by the given properties. Possible values are: `screenitemid` and `screenid`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
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

Retrieve template screen items for screen

Return all template screen items from template screen "15".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreenitem.get",
    "params": {
        "output": "extend",
        "screenids": "15"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "screenitemid": "42",
            "screenid": "15",
            "resourcetype": "0",
            "resourceid": "454",
            "width": "500",
            "height": "200",
            "x": "0",
            "y": "0",
            "colspan": "1",
            "rowspan": "1",
            "elements": "0",
            "valign": "1",
            "halign": "0",
            "style": "0",
            "url": "",
            "max_columns": "3"
        },
        {
            "screenitemid": "43",
            "screenid": "15",
            "resourcetype": "0",
            "resourceid": "455",
            "width": "500",
            "height": "270",
            "x": "1",
            "y": "0",
            "colspan": "1",
            "rowspan": "1",
            "elements": "0",
            "valign": "1",
            "halign": "0",
            "style": "0",
            "url": "",
            "max_columns": "3"
        }
    ],
    "id": 1
```

```
}
```

Source

CTemplateScreenItem::get() in *ui/include/classes/api/services/CTemplateScreenItem.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.

Note:
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.

Property	Type	Description
clock	timestamp	Timestamp of an hour for which the value was calculated. E. g. timestamp of 04:00:00 means values calculated for period 04:00:00-04:59:59.
itemid	integer	ID of the related item.
num	integer	Number of values that were available for the hour.
value_min	float	Hourly minimum value.
value_avg	float	Hourly average value.
value_max	float	Hourly maximum value.

Integer trend

The integer trend object has the following properties.

Property	Type	Description
clock	timestamp	Timestamp of an hour for which the value was calculated. E. g. timestamp of 04:00:00 means values calculated for period 04:00:00-04:59:59.
itemid	integer	ID of the related item.
num	integer	Number of values that were available for the hour.
value_min	integer	Hourly minimum value.
value_avg	integer	Hourly average value.
value_max	integer	Hourly maximum value.

trend.get

Description

`integer/array trend.get(object parameters)`

The method allows to retrieve trend data according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
itemids	string/array	Return only trends with the given item IDs.
time_from	timestamp	Return only values that have been collected after or at the given time.
time_till	timestamp	Return only values that have been collected before or at the given time.
countOutput	boolean	Count the number of retrieved objects.
limit	integer	Limit the amount of retrieved objects.
output	query	Set Trend object properties to be returned.

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:

```
{
    "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:

```
{
    "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.adddependencies - adding new trigger dependencies
- trigger.create - creating new triggers
- trigger.delete - deleting triggers
- trigger.deletedependencies - deleting trigger dependencies
- 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.

Property	Type	Description
triggerid	string	*(readonly)* ID of the trigger.
description (required)	string	Name of the trigger.
expression (required)	string	Reduced trigger expression.
opdata	string	Operational data.
comments	string	Additional description of the trigger.
error	string	*(readonly)* Error text if there have been any problems when updating the state of the trigger.
flags	integer	*(readonly)* Origin of the trigger. Possible values are: 0 - *(default)* a plain trigger; 4 - a discovered trigger.
lastchange	timestamp	*(readonly)* Time when the trigger last changed its state.
priority	integer	Severity of the trigger. Possible values are: 0 - *(default)* not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
state	integer	*(readonly)* State of the trigger. Possible values: 0 - *(default)* trigger state is up to date; 1 - current trigger state is unknown.
status	integer	Whether the trigger is enabled or disabled. Possible values are: 0 - *(default)* enabled; 1 - disabled.

Property	Type	Description
templateid	string	*(readonly)* ID of the parent template trigger.
type	integer	Whether the trigger can generate multiple problem events. Possible values are: 0 - *(default)* do not generate multiple events; 1 - generate multiple events.
url	string	URL associated with the trigger.
value	integer	*(readonly)* Whether the trigger is in OK or problem state. Possible values are: 0 - *(default)* OK; 1 - problem.
recovery_mode	integer	OK event generation mode. Possible values are: 0 - *(default)* Expression; 1 - Recovery expression; 2 - None.
recovery_expression	string	Reduced trigger recovery expression.
correlation_mode	integer	OK event closes. Possible values are: 0 - *(default)* All problems; 1 - All problems if tag values match.
correlation_tag	string	Tag for matching.
manual_close	integer	Allow manual close. Possible values are: 0 - *(default)* No; 1 - Yes.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Trigger tag

The trigger tag object has the following properties.

Property	Type	Description
tag (required)	string	Trigger tag name.
value	string	Trigger tag value.

trigger.adddependencies

Description

`object trigger.adddependencies(object/array triggerDependencies)`

This method allows to create new trigger dependencies.

Parameters

`(object/array)` Trigger dependencies to create.

Each trigger dependency has the following parameters:

Parameter	Type	Description
triggerid (required)	string	ID of the dependent trigger.
dependsOnTriggerid (required)	string	ID of the trigger that the trigger depends on.

Return values

(object) Returns an object containing the IDs of the dependent triggers under the triggerids property.

Examples

Add a trigger dependency

Make trigger "14092" dependent on trigger "13565."

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.adddependencies",
    "params": {
        "triggerid": "14092",
        "dependsOnTriggerid": "13565"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "14092"
        ]
    },
    "id": 1
}
```

See also

- trigger.update
- Trigger dependencies

Source

CTrigger::addDependencies() in *ui/include/classes/api/services/CTrigger.php*.

trigger.create

Description

`object trigger.create(object/array triggers)`

This method allows to create new triggers.

Parameters

(object/array) Triggers to create.

Additionally to the standard trigger properties the method accepts the following parameters.

Parameter	Type	Description
dependencies	array	Triggers that the trigger is dependent on. The triggers must have the triggerid property defined.
tags	array	Trigger tags.

Attention:
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:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.create",
    "params": [
        {
            "description": "Processor load is too high on {HOST.NAME}",
            "expression": "{Linux server:system.cpu.load[percpu,avg1].last()}>5",
            "dependencies": [
                {
                    "triggerid": "17367"
                }
            ]
        },
        {
            "description": "Service status",
            "expression": "{Linux server:log[/var/log/system,Service .* has stopped].strlen()}<>0",
            "dependencies": [
                {
                    "triggerid": "17368"
                }
            ],
            "tags": [
                {
                    "tag": "service",
                    "value": "{{ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"\\1\")}"
                },
                {
                    "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.

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:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "12002",
            "12003"
        ]
    },
    "id": 1
}
```

Source

CTrigger::delete() in *ui/include/classes/api/services/CTrigger.php*.

trigger.deletedependencies

Description

`object trigger.deletedependencies(string/array triggers)`

This method allows to delete all trigger dependencies from the given triggers.

Parameters

`(string/array)` Triggers to delete the trigger dependencies from.

Return values

`(object)` Returns an object containing the IDs of the affected triggers under the `triggerids` property.

Examples

Deleting dependencies from multiple triggers

Delete all dependencies from two triggers.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.deleteDependencies",
    "params": [
        {
            "triggerid": "14544"
        },
        {
            "triggerid": "14545"
        }
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "14544",
            "14545"
        ]
    },
    "id": 1
}
```

See also

- trigger.update

Source

CTrigger::deleteDependencies() in *ui/include/classes/api/services/CTrigger.php*.

trigger.get

Description

`integer/array trigger.get(object parameters)`

The method allows to retrieve triggers according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
triggerids	string/array	Return only triggers with the given IDs.
groupids	string/array	Return only triggers that belong to hosts from the given host groups.
templateids	string/array	Return only triggers that belong to the given templates.
hostids	string/array	Return only triggers that belong to the given hosts.
itemids	string/array	Return only triggers that contain the given items.
applicationids	string/array	Return only triggers that contain items from the given applications.
functions	string/array	Return only triggers that use the given functions. Refer to the supported trigger functions page for a list of supported functions.
group	string	Return only triggers that belong to hosts from the host group with the given name.
host	string	Return only triggers that belong to host with the given technical name.
inherited	boolean	If set to `true` return only triggers inherited from a template.
templated	boolean	If set to `true` return only triggers that belong to templates.

Parameter	Type	Description
dependent	boolean	If set to `true` return only triggers that have dependencies. If set to `false` return only triggers that do not have dependencies.
monitored	flag	Return only enabled triggers that belong to monitored hosts and contain only enabled items.
active	flag	Return only enabled triggers that belong to monitored hosts.
maintenance	boolean	If set to `true` return only enabled triggers that belong to hosts in maintenance.
withUnacknowledgedEvents	flag	Return only triggers that have unacknowledged events.
withAcknowledgedEvents	flag	Return only triggers with all events acknowledged.
withLastEventUnacknowledged	flag	Return only triggers with the last event unacknowledged.
skipDependent	flag	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.
lastChangeSince	timestamp	Return only triggers that have changed their state after the given time.
lastChangeTill	timestamp	Return only triggers that have changed their state before the given time.
only_true	flag	Return only triggers that have recently been in a problem state.
min_severity	integer	Return only triggers with severity greater or equal than the given severity.
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.
tags	array of objects	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: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all triggers. Possible operator types: 0 - (default) Like; 1 - Equal.
expandComment	flag	Expand macros in the trigger description.
expandDescription	flag	Expand macros in the name of the trigger.
expandExpression	flag	Expand functions and macros in the trigger expression.
selectGroups	query	Return the host groups that the trigger belongs to in the groups property.
selectHosts	query	Return the hosts that the trigger belongs to in the hosts property.
selectItems	query	Return items contained by the trigger in the items property.
selectFunctions	query	Return functions used in the trigger in the `functions` property. The function objects represents 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.
selectDependencies	query	Return triggers that the trigger depends on in the `dependencies` property.
selectDiscoveryRule	query	Return the low-level discovery rule that created the trigger.
selectLastEvent	query	Return the last significant trigger event in the lastEvent property.
selectTags	query	Return the trigger tags in tags property.
selectTriggerDiscovery	query	Return the trigger discovery object in the `triggerDiscovery` property. The trigger discovery objects links the trigger to a trigger prototype from which it was created. It has the following properties: `parent_triggerid` - (`string`) ID of the trigger prototype from which the trigger has been created.

Parameter	Type	Description
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 belongs to; `hostid` - ID of the host that the trigger belongs to.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectHosts` - results will be sorted by `host`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `triggerid`, `description`, `status`, `priority`, `lastchange` and `hostname`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
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 data by trigger ID

Retrieve all data and the functions used in trigger "14062".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "triggerids": "14062",
        "output": "extend",
        "selectFunctions": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "14062",
            "expression": "{13513}>0",
            "description": "/etc/passwd has been changed on {HOST.NAME}",
```

```
            "url": "",
            "status": "0",
            "value": "0",
            "priority": "2",
            "lastchange": "0",
            "comments": "",
            "error": "",
            "templateid": "10016",
            "type": "0",
            "state": "0",
            "flags": "0",
            "recovery_mode": "0",
            "recovery_expression": "",
            "correlation_mode": "0",
            "correlation_tag": "",
            "manual_close": "0",
            "opdata": "",
            "functions": [
                {
                    "functionid": "13513",
                    "itemid": "24350",
                    "triggerid": "14062",
                    "parameter": "0",
                    "function": "diff"
                }
            ]
        }
    ],
    "id": 1
}
```

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:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "output": [
            "triggerid",
            "description",
            "priority"
        ],
        "filter": {
            "value": 1
        },
        "sortfield": "priority",
        "sortorder": "DESC"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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",
            "priority": "3"
        }
    ],
    "id": 1
}
```

Retrieving a specific trigger with tags

Retrieve a specific trigger with tags.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "output": [
            "triggerid",
            "description"
        ],
        "selectTags": "extend",
        "triggerids": [
            "17578"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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

Source

CTrigger::get() in *ui/include/classes/api/services/CTrigger.php*.

trigger.update

Description

`object trigger.update(object/array triggers)`

This method allows to update existing triggers.

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.

Parameter	Type	Description
dependencies	array	Triggers that the trigger is dependent on. The triggers must have the `triggerid` property defined.
tags	array	Trigger tags.

Attention:
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:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": {
        "triggerid": "13938",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

Replacing triggers tags

Replace tags for trigger.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": {
        "triggerid": "13938",
        "tags": [
            {
                "tag": "service",
                "value": "{{ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"\\1\")}"
            },
            {
                "tag": "error",
                "value": ""
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

See also

- trigger.adddependencies
- trigger.deletedependencies

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.

Property	Type	Description
triggerid	string	*(readonly)* ID of the trigger prototype.
description (required)	string	Name of the trigger prototype.
expression (required)	string	Reduced trigger expression.
opdata	string	Operational data.
comments	string	Additional comments to the trigger prototype.
priority	integer	Severity of the trigger prototype. Possible values: 0 - *(default)* not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
status	integer	Whether the trigger prototype is enabled or disabled. Possible values: 0 - *(default)* enabled; 1 - disabled.
templateid	string	*(readonly)* ID of the parent template trigger prototype.
type	integer	Whether the trigger prototype can generate multiple problem events. Possible values: 0 - *(default)* do not generate multiple events; 1 - generate multiple events.
url	string	URL associated with the trigger prototype.
recovery_mode	integer	OK event generation mode. Possible values are: 0 - *(default)* Expression; 1 - Recovery expression; 2 - None.
recovery_expression	string	Reduced trigger recovery expression.
correlation_mode	integer	OK event closes. Possible values are: 0 - *(default)* All problems; 1 - All problems if tag values match.
correlation_tag	string	Tag for matching.
manual_close	integer	Allow manual close. Possible values are: 0 - *(default)* No; 1 - Yes.
discover	integer	Trigger prototype discovery status. Possible values: 0 - *(default)* new triggers will be discovered; 1 - new triggers will not be discovered and existing triggers will be marked as lost.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Trigger prototype tag

The trigger prototype tag object has the following properties.

Property	Type	Description
tag (required)	string	Trigger prototype tag name.
value	string	Trigger prototype tag value.

triggerprototype.create

Description

`object triggerprototype.create(object/array triggerPrototypes)`

This method allows to create new trigger prototypes.

Parameters

`(object/array)` Trigger prototypes to create.

Additionally to the standard trigger prototype properties the method accepts the following parameters.

Parameter	Type	Description
dependencies	array	Triggers and trigger prototypes that the trigger prototype is dependent on. The triggers must have the `triggerid` property defined.
tags	array	Trigger prototype tags.

Attention:
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:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.create",
    "params": {
        "description": "Free disk space is less than 20% on volume {#FSNAME}",
        "expression": "{Zabbix server:vfs.fs.size[{#FSNAME},pfree].last()}<20",
        "tags": [
            {
                "tag": "volume",
                "value": "{#FSNAME}"
            },
            {
                "tag": "type",
                "value": "{#FSTYPE}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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.

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:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.delete",
    "params": [
        "12002",
        "12003"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "12002",
            "12003"
        ]
    },
    "id": 1
}
```

Source

CTriggerPrototype::delete() in *ui/include/classes/api/services/CTriggerPrototype.php*.

triggerprototype.get

Description

`integer/array triggerprototype.get(object parameters)`

The method allows to retrieve trigger prototypes according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
active	flag	Return only enabled trigger prototypes that belong to monitored hosts.
applicationids	string/array	Return only trigger prototypes that contain items from the given applications.
discoveryids	string/array	Return only trigger prototypes that belong to the given LLD rules.
functions	string/array	Return only triggers that use the given functions. Refer to the supported trigger functions page for a list of supported functions.
group	string	Return only trigger prototypes that belong to hosts from the host groups with the given name.
groupids	string/array	Return only trigger prototypes that belong to hosts from the given host groups.
host	string	Return only trigger prototypes that belong to hosts with the given name.
hostids	string/array	Return only trigger prototypes that belong to the given hosts.
inherited	boolean	If set to `true` return only trigger prototypes inherited from a template.
maintenance	boolean	If set to `true` return only enabled trigger prototypes that belong to hosts in maintenance.
min_severity	integer	Return only trigger prototypes with severity greater or equal than the given severity.
monitored	flag	Return only enabled trigger prototypes that belong to monitored hosts and contain only enabled items.
templated	boolean	If set to `true` return only trigger prototypes that belong to templates.
templateids	string/array	Return only trigger prototypes that belong to the given templates.
triggerids	string/array	Return only trigger prototypes with the given IDs.
expandExpression	flag	Expand functions and macros in the trigger expression.
selectDiscoveryRule	query	Return the LLD rule that the trigger prototype belongs to.
selectFunctions	query	Return functions used in the trigger prototype in the `functions` property. The function objects represents 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.
selectGroups	query	Return the host groups that the trigger prototype belongs to in the 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.
selectDependencies	query	Return trigger prototypes and triggers that the trigger prototype depends on in the `dependencies` property.
selectTags	query	Return the trigger prototype tags in tags 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.

Parameter	Type	Description
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectHosts` - results will be sorted by `host`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `triggerid`, `description`, `status` and `priority`.
countOutput	boolean	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	

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 prototypes from an LLD rule

Retrieve all trigger prototypes and their functions from an LLD rule.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.get",
    "params": {
        "output": "extend",
        "selectFunctions": "extend",
        "discoveryids": "22450"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "13272",
            "expression": "{12598}<20",
            "description": "Free inodes is less than 20% on volume {#FSNAME}",
            "url": "",
            "status": "0",
            "priority": "2",
            "comments": "",
            "templateid": "0",
            "type": "0",
            "flags": "2",
            "recovery_mode": "0",
```

```
            "recovery_expression": "",
            "correlation_mode": "0",
            "correlation_tag": "",
            "manual_close": "0",
            "opdata": "",
            "discover": "0",
            "functions": [
                {
                    "functionid": "12598",
                    "itemid": "22454",
                    "triggerid": "13272",
                    "parameter": "0",
                    "function": "last"
                }
            ]
        },
        {
            "triggerid": "13266",
            "expression": "{13500}<201",
            "description": "Free disk space is less than 20% on volume {#FSNAME}",
            "url": "",
            "status": "0",
            "priority": "2",
            "comments": "",
            "templateid": "0",
            "type": "0",
            "flags": "2",
            "recovery_mode": "0",
            "recovery_expression": "",
            "correlation_mode": "0",
            "correlation_tag": "",
            "manual_close": "0",
            "opdata": "",
            "discover": "0",
            "functions": [
                {
                    "functionid": "13500",
                    "itemid": "22686",
                    "triggerid": "13266",
                    "parameter": "0",
                    "function": "last"
                }
            ]
        }
    ],
    "id": 1
}
```


Retrieving a specific trigger prototype with tags

Request:


```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.get",
    "params": {
        "output": [
            "triggerid",
            "description"
        ],
        "selectTags": "extend",
        "triggerids": [
            "17373"
        ]
```

```
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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
}
```

See also

- Discovery rule
- Item
- Host
- Host group

Source

CTriggerPrototype::get() in *ui/include/classes/api/services/CTriggerPrototype.php*.

triggerprototype.update

Description

`object triggerprototype.update(object/array triggerPrototypes)`

This method allows to update existing trigger prototypes.

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.

Parameter	Type	Description
dependencies	array	Triggers and trigger prototypes that the trigger prototype is dependent on. The triggers must have the `triggerid` property defined.
tags	array	Trigger prototype tags.

Attention:
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:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.update",
    "params": {
        "triggerid": "13938",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

Replacing trigger prototype tags

Replace tags for one trigger prototype.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.update",
    "params": {
        "triggerid": "17373",
        "tags": [
            {
                "tag": "volume",
                "value": "{#FSNAME}"
            },
            {
                "tag": "type",
                "value": "{#FSTYPE}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "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.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.update - updating users

> User object

The following objects are directly related to the `user` API.

User

The user object has the following properties.

Property	Type	Description
userid	string	*(readonly)* ID of the user.
alias *(required)*	string	User alias.
attempt_clock	timestamp	*(readonly)* Time of the last unsuccessful login attempt.
attempt_failed	integer	*(readonly)* Recent failed login attempt count.
attempt_ip	string	*(readonly)* IP address from where the last unsuccessful login attempt came from.
autologin	integer	Whether to enable auto-login. Possible values: 0 - *(default)* auto-login disabled; 1 - auto-login enabled.
autologout	string	User session life time. Accepts seconds and time unit with suffix. If set to 0s, the session will never expire. Default: 15m.
lang	string	Language code of the user's language. Default: `en_GB`.
name	string	Name of the user.
refresh	string	Automatic refresh period. Accepts seconds or time unit with suffix (e.g., 30s, 90s, 1m, 1h). Default: 30s.
rows_per_page	integer	Amount of object rows to show per page. Default: 50.
surname	string	Surname of the user.

Property	Type	Description
theme	string	User's theme. Possible values: `default` - *(default)* system default; `blue-theme` - Blue; `dark-theme` - Dark.
type	integer	Type of the user. Possible values: 1 - *(default)* Zabbix user; 2 - Zabbix admin; 3 - Zabbix super admin.
url	string	URL of the page to redirect the user to after logging in.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Media

The media object has the following properties.

Property	Type	Description
mediatypeid (required)	string	ID of the media type used by the media.
sendto (required)	string/array	Address, user name or other identifier of the recipient. If type of Media type is e-mail, values are represented as array. For other types of Media types, value is represented as a string.
active	integer	Whether the media is enabled. Possible values: 0 - *(default)* enabled; 1 - disabled.
severity	integer	Trigger severities to send notifications about. Possible bitmap values are: 1 - Not classified; 2 - Information; 4 - Warning; 8 - Average; 16 - High; 32 - Disaster. This is a bitmask field; any sum of possible bitmap values is acceptable (for example, 48 for Average, High, and Disaster). Default: 63.
period	string	Time when the notifications can be sent as a time period or user macros separated by a semicolon. Default: 1-7,00:00-24:00

user.checkAuthentication

Description

`object user.checkAuthentication`

This method checks and prolongs user session.

Parameters

The method accepts the following parameters.

Parameter	Type	Description
extend	boolean	Default value: ”true”. Setting it’s value to ”false” allows to check session without extending it’s lifetime. Supported since Zabbix 4.0.
sessionid	string	User session id.

Attention:
Calling **user.checkAuthentication** method prolongs user session by default.

Return values

(object) Returns an object containing information about user.

Examples

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.checkAuthentication",
    "params": {
        "sessionid": "8C8447FF6F61D134CEAC740CCA1BC90D"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userid": "1",
        "alias": "Admin",
        "name": "Zabbix",
        "surname": "Administrator",
        "url": "",
        "autologin": "1",
        "autologout": "0",
        "lang": "ru_RU",
        "refresh": "0",
        "type": "3",
        "theme": "default",
        "attempt_failed": "0",
        "attempt_ip": "127.0.0.1",
        "attempt_clock": "1355919038",
        "rows_per_page": "50",
        "debug_mode": true,
        "userip": "127.0.0.1",
        "sessionid": "8C8447FF6F61D134CEAC740CCA1BC90D",
        "gui_access": "0"
    },
    "id": 1
}
```

Note:
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.

Parameters

`(object/array)` Users to create.

Additionally to the standard user properties, the method accepts the following parameters.

Parameter	Type	Description
passwd *(required)*	string	User's password. Can be omitted if user is added only to groups that have LDAP access.
usrgrps (required)	array	User groups to add the user to. The user groups must have the `usrgrpid` property defined.
user_medias	array	User media to be created.

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:

```
{
    "jsonrpc": "2.0",
    "method": "user.create",
    "params": {
        "alias": "John",
        "passwd": "Doe123",
        "usrgrps": [
            {
                "usrgrpid": "7"
            }
        ],
        "user_medias": [
            {
                "mediatypeid": "1",
                "sendto": [
                    "support@company.com"
                ],
                "active": 0,
                "severity": 63,
                "period": "1-7,00:00-24:00"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "12"
```

```
        ]
    },
    "id": 1
}
```

See also

- Media
- User group

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.

Parameters

`(array)` IDs of users to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted users under the `userids` 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": {
        "userids": [
            "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.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Note:
Since Zabbix 5.0.46, when requesting user media or permissions, *Admin* and *User* type users may retrieve data only about their own user. For an example, see Retrieving users as *Admin*.

Parameter	Type	Description
mediaids	string/array	Return only users that use the given media.
mediatypeids	string/array	Return only users that use the given media types.
userids	string/array	Return only users with the given IDs.
usrgrpids	string/array	Return only users that belong to the given user groups.
getAccess	flag	Adds additional information about user permissions. Adds the following properties for each user: `gui_access` - *(integer)* user's frontend authentication method. Refer to the `gui_access` property of the user group object for a list of possible values. `debug_mode` - *(integer)* indicates whether debug is enabled for the user. Possible values: 0 - debug disabled, 1 - debug enabled. `users_status` - *(integer)* indicates whether the user is disabled. Possible values: 0 - user enabled, 1 - user disabled.
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. See `usergroup.get` for restrictions based on user type.
filter	object	Return only those results that exactly match the given filter. Accepts an object, where the keys are property names, and the values are either a single value or an array of values to match against. Does not support properties of `text` data type. Possible User object properties for *Admin* and *User* type users when requesting data on users in their user group (since Zabbix 5.0.46): `userid`, `alias`, `name`, `surname`.
output	query	User object properties to be returned. Since Zabbix 5.0.46, *Admin* and *User* type users may retrieve only the following properties: - For their own user: `userid`, `alias`, `attempt_clock`, `attempt_failed`, `attempt_ip`, `autologin`, `autologout`, `lang`, `name`, `refresh`, `rows_per_page`, `surname`, `theme`, `type`, `url`. - For users in their user group: `userid`, `alias`, `name`, `surname`. Default: `extend`.
search	object	Return results that match the given pattern (case-insensitive). Accepts an object, 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 "%…%"` search. Supports only properties of `string` and `text` data type. Possible User object properties for *Admin* and *User* type users when requesting data on users in their user group (since Zabbix 5.0.46): `alias`, `name`, `surname`.

Parameter	Type	Description
sortfield	string/array	Sort the result by the given properties. Possible values are: `userid` and `alias`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
limit	integer	
preservekeys	boolean	
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 users

Retrieve all of the configured users.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.get",
    "params": {
        "output": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "userid": "1",
            "alias": "Admin",
            "name": "Zabbix",
            "surname": "Administrator",
            "url": "",
            "autologin": "1",
            "autologout": "0s",
            "lang": "ru_RU",
            "refresh": "0s",
            "type": "3",
            "theme": "default",
            "attempt_failed": "0",
            "attempt_ip": "",
            "attempt_clock": "0",
            "rows_per_page": "50"
        },
        {
            "userid": "2",
            "alias": "guest",
            "name": "Default2",
```

```
            "surname": "User",
            "url": "",
            "autologin": "0",
            "autologout": "15m",
            "lang": "en_GB",
            "refresh": "30s",
            "type": "1",
            "theme": "default",
            "attempt_failed": "0",
            "attempt_ip": "",
            "attempt_clock": "0",
            "rows_per_page": "50"
        }
    ],
    "id": 1
}
```

Retrieving users as *Admin*

As an *Admin* type user, retrieve detailed data about your own user and limited data for users in your user group.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.get",
    "params": {
        "output": "extend",
        "getAccess": true,
        "selectMedias": "extend",
        "selectMediatypes": "extend",
        "selectUsrgrps": "extend"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "userid": "1",
            "alias": "Admin",
            "name": "Zabbix",
            "surname": "Administrator",
            "usrgrps": [
                {
                    "usrgrpid": "7",
                    "name": "Zabbix administrators",
                    "gui_access": "0",
                    "users_status": "0",
                    "debug_mode": "0"
                }
            ]
        },
        {
            "userid": "3",
            "alias": "database-admin",
            "name": "John",
            "surname": "Doe",
            "url": "",
            "autologin": "0",
            "autologout": "0",
```

```
            "lang": "en_GB",
            "refresh": "30s",
            "type": "2",
            "theme": "default",
            "attempt_failed": "0",
            "attempt_ip": "",
            "attempt_clock": "0",
            "rows_per_page": "50",
            "gui_access": "0",
            "debug_mode": "0",
            "users_status": "0",
            "usrgrps": [
                {
                    "usrgrpid": "7",
                    "name": "Zabbix administrators",
                    "gui_access": "0",
                    "users_status": "0",
                    "debug_mode": "0"
                }
            ],
            "medias": [
                {
                    "mediaid": "2",
                    "userid": "3",
                    "mediatypeid": "1",
                    "sendto": [
                        "john.doe@example.com"
                    ],
                    "active": "0",
                    "severity": "63",
                    "period": "1-7,00:00-24:00"
                }
            ],
            "mediatypes": [
                {
                    "mediatypeid": "1",
                    "type": "0",
                    "name": "Email",
                    "status": "0",
                    "description": "",
                    "maxattempts": "3"
                }
            ]
        }
    ],
    "id": 1
}
```


See also

- Media
- Media type
- User group

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.

Warning:
When using this method, you also need to do user.logout to prevent the generation of a large number of open session records.

Parameters

Attention:
This method is available to unauthenticated users and must be called without the `auth` parameter in the JSON-RPC request.

`(object)` Parameters containing the user name and password.

The method accepts the following parameters.

Parameter	Type	Description
password (required)	string	User password.
user (required)	string	User name.
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:

Property	Type	Description
debug_mode	boolean	Whether debug mode is enabled for the user.
gui_access	integer	User's authentication method to the frontend. Refer to the `gui_access` property of the user group object for a list of possible values.
sessionid	string	Authentication token, which must be used in the following API requests.
userip	string	IP address of the user.

Note:
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.

Note:
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:

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "user": "Admin",
        "password": "zabbix"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "0424bd59b807674191e7d77572075f33",
    "id": 1
}
```

Requesting authenticated user's information

Authenticate and return additional information about the user.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "user": "Admin",
        "password": "zabbix",
        "userData": true
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userid": "1",
        "alias": "Admin",
        "name": "Zabbix",
        "surname": "Administrator",
        "url": "",
        "autologin": "1",
        "autologout": "0",
        "lang": "ru_RU",
        "refresh": "0",
        "type": "3",
        "theme": "default",
        "attempt_failed": "0",
        "attempt_ip": "127.0.0.1",
        "attempt_clock": "1355919038",
        "rows_per_page": "50",
        "debug_mode": true,
        "userip": "127.0.0.1",
        "sessionid": "5b56eee8be445e98f0bd42b435736e42",
        "gui_access": "0"
    },
    "id": 1
}
```

See also

- user.logout

Source

CUser::login() in *ui/include/classes/api/services/CUser.php*.

user.logout

Description

`string/object user.logout(array)`

This method allows to log out of the API and invalidates the current authentication token.

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:

```
{
    "jsonrpc": "2.0",
    "method": "user.logout",
    "params": [],
    "id": 1,
    "auth": "16a46baf181ef9602e1687f3110abf8a"
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- user.login

Source

CUser::login() in *ui/include/classes/api/services/CUser.php*.

user.update

Description

`object user.update(object/array users)`

This method allows to update existing users.

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.

Parameter	Type	Description
passwd	string	User's password. Can be empty string if user belongs to or is moved only to groups that have LDAP access.
usrgrps	array	User groups to replace existing user groups. The user groups must have the `usrgrpid` property defined.
user_medias	array	User media to replace existing media.

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:

```
{
    "jsonrpc": "2.0",
    "method": "user.update",
    "params": {
        "userid": "1",
        "name": "John",
        "surname": "Doe"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "1"
        ]
    },
    "id": 1
}
```

Source

CUser::update() in *ui/include/classes/api/services/CUser.php*.

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.

Property	Type	Description
usrgrpid	string	*(readonly)* ID of the user group.
name (required)	string	Name of the user group.

Property	Type	Description
debug_mode	integer	Whether debug mode is enabled or disabled. Possible values are: 0 - *(default)* disabled; 1 - enabled.
gui_access	integer	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.
users_status	integer	Whether the user group is enabled or disabled. Possible values are: 0 - *(default)* enabled; 1 - disabled.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Permission

The permission object has the following properties.

Property	Type	Description
id (required)	string	ID of the host group to add permission to.
permission (required)	integer	Access level to the host group. Possible values: 0 - access denied; 2 - read-only access; 3 - read-write access.

Tag based permission

The tag based permission object has the following properties.

Property	Type	Description
groupid (required)	string	ID of the host group to add permission to.
tag	string	Tag name.
value	string	Tag value.

usergroup.create

Description

`object usergroup.create(object/array userGroups)`

This method allows to create new user groups.

Parameters

`(object/array)` User groups to create.

Additionally to the standard user group properties, the method accepts the following parameters.

Parameter	Type	Description
rights	object/array	Permissions to assign to the group

Parameter	Type	Description
tag_filters	array	Tag based permissions to assign to the group
userids	string/array	IDs of users to add to the user group.

Return values

`(object)` Returns an object containing the IDs of the created user groups under the `usrgrpids` property. The order of the returned IDs matches the order of the passed user groups.

Examples

Creating a user group

Create a user group, which denies access to host group "2", and add a user to it.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.create",
    "params": {
        "name": "Operation managers",
        "rights": {
            "permission": 0,
            "id": "2"
        },
        "userids": "12"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "20"
        ]
    },
    "id": 1
}
```

See also

- Permission

Source

CUserGroup::create() in *ui/include/classes/api/services/CUserGroup.php*.

usergroup.delete

Description

`object usergroup.delete(array userGroupIds)`

This method allows to delete user groups.

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 `usrgrpids` property.

Examples

Deleting multiple user groups

Delete two user groups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.delete",
    "params": [
        "20",
        "21"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "20",
            "21"
        ]
    },
    "id": 1
}
```

Source

CUserGroup::delete() in *ui/include/classes/api/services/CUserGroup.php*.

usergroup.get

Description

`integer/array usergroup.get(object parameters)`

The method allows to retrieve user groups according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
status	integer	Return only user groups with the given status. Refer to the user group page for a list of supported statuses.
userids	string/array	Return only user groups that contain the given users.
usrgrpids	string/array	Return only user groups with the given IDs.
selectTagFilters	query	Return user group tag based permissions in the tag_filters property. It has the following properties: `groupid` - (string) ID of the host group; `tag` - (string) tag name; `value` - (string) tag value.
selectUsers	query	Return the users from the user group in the users property. See `user.get` for restrictions based on user type.

Parameter	Type	Description
selectRights	query	Return user group rights in the rights property. It has the following properties: `permission` - (integer) access level to the host group; `id` - (string) ID of the host group. Refer to the user group page for a list of access levels to host groups.
limitSelects	integer	Limits the number of records returned by subselects.
output	query	User group object properties to be returned. Since Zabbix 5.0.46, *Admin* and *User* type users may retrieve only the following properties: `usrgrpid`, `name`, `gui_access`, `users_status`, `debug_mode`. Default: `extend`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `usrgrpid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
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 enabled user groups

Retrieve all enabled user groups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.get",
    "params": {
        "output": "extend",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "usrgrpid": "7",
            "name": "Zabbix administrators",
```

```
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "1"
        },
        {
            "usrgrpid": "8",
            "name": "Guests",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0"
        },
        {
            "usrgrpid": "11",
            "name": "Enabled debug mode",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "1"
        },
        {
            "usrgrpid": "12",
            "name": "No access to the frontend",
            "gui_access": "2",
            "users_status": "0",
            "debug_mode": "0"
        },
        {
            "usrgrpid": "14",
            "name": "Read only",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0"
        },
        {
            "usrgrpid": "18",
            "name": "Deny",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0"
        }
    ],
    "id": 1
}
```


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.

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.

Parameter	Type	Description
rights	object/array	Permissions to replace the current permissions assigned to the user group.
tag_filters	array	Tag based permissions to replace the current permissions assigned to the user group.
userids	string/array	IDs of the users to replace the users in the user group.

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:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.update",
    "params": {
        "usrgrpid": "17",
        "users_status": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "17"
        ]
    },
    "id": 1
}
```

See also

- Permission

Source

CUserGroup::update() in *ui/include/classes/api/services/CUserGroup.php*.

User macro

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.

Property	Type	Description
globalmacroid	string	*(readonly)* ID of the global macro.
macro (required)	string	Macro string.
value (required)	string	Value of the macro.
type	integer	Type of macro. Possible values: 0 - *(default)* Text macro; 1 - Secret macro.
description	string	Description of the macro.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Host macro

The host macro object defines a macro available on a host, host prototype or template. It has the following properties.

Property	Type	Description
hostmacroid	string	*(readonly)* ID of the host macro.
hostid (required)	string	ID of the host that the macro belongs to.
macro (required)	string	Macro string.
value (required)	string	Value of the macro.
type	integer	Type of macro. Possible values: 0 - *(default)* Text macro; 1 - Secret macro.
description	string	Description of the macro.

Note that for some methods (update, delete) the required/optional parameter combination is different.

usermacro.create

Description

`object usermacro.create(object/array hostMacros)`

This method allows to create new host macros.

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.

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:

```
{
    "jsonrpc": "2.0",
    "result": {
        "globalmacroids": [
            "6"
        ]
    },
    "id": 1
}
```

Source

CUserMacro::createGlobal() in *ui/include/classes/api/services/CUserMacro.php*.

usermacro.delete

Description

`object usermacro.delete(array hostMacroIds)`

This method allows to delete host macros.

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::delete() in *ui/include/classes/api/services/CUserMacro.php*.

usermacro.deleteglobal

Description

`object usermacro.deleteglobal(array globalMacroIds)`

This method allows to delete global macros.

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

Description

`integer/array usermacro.get(object parameters)`

The method allows to retrieve host and global macros according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
globalmacro	flag	Return global macros instead of host macros.
globalmacroids	string/array	Return only global macros with the given IDs.
groupids	string/array	Return only host macros that belong to hosts or templates from the given host groups.

Parameter	Type	Description
hostids	string/array	Return only macros that belong to the given hosts or templates.
hostmacroids	string/array	Return only host macros with the given IDs.
selectGroups	query	Return host groups that the host macro belongs to in the groups property. Used only when retrieving host macros.
selectHosts	query	Return hosts that the host macro belongs to in the hosts property. Used only when retrieving host macros.
selectTemplates	query	Return templates that the host macro belongs to in the templates property. Used only when retrieving host macros.
sortfield	string/array	Sort the result by the given properties. Possible value: `macro`.
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 host macros for a host

Retrieve all host macros defined for host "10198".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.get",
    "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"
        },
        {
            "hostmacroid": "11",
            "hostid": "10198",
            "macro": "{$SNMP_COMMUNITY}",
            "value": "public",
            "description": "",
            "type": "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
}
```

Source

CUserMacro::get() in *ui/include/classes/api/services/CUserMacro.php*.

usermacro.update

Description

`object usermacro.update(object/array hostMacros)`

This method allows to update existing host macros.

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

Changing the value of a host macro

Change the value of a host macro to "public".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.update",
    "params": {
        "hostmacroid": "1",
        "value": "public"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "globalmacroids": [
            "1"
        ]
    },
    "id": 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.

Property	Type	Description
valuemapid	string	*(readonly)* ID of the value map.
name (required)	string	Name of the value map.
mappings (required)	array	Value mappings for current value map. The mapping object is described in detail below.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Value mappings

The value mappings object defines value mappings of the value map. It has the following properties.

Property	Type	Description
value (required)	string	Original value.
newvalue (required)	string	Value to which the original value is mapped to.

valuemap.create

Description

`object valuemap.create(object/array valuemaps)`

This method allows to create new value maps.

Parameters

`(object/array)` Value maps to create.

The method accepts value maps with the standard value map properties.

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:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.create",
    "params": {
        "name": "Service state",
        "mappings": [
            {
                "value": "0",
                "newvalue": "Down"
            },
            {
                "value": "1",
                "newvalue": "Up"
            }
        ]
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "1"
        ]
    },
    "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.

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:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.delete",
    "params": [
        "1",
        "2"
    ],
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

Source

CValueMap::delete() in *ui/include/classes/api/services/CValueMap.php*.

valuemap.get

Description

`integer/array valuemap.get(object parameters)`

The method allows to retrieve value maps according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
valuemapids	string/array	Return only value maps with the given IDs.
selectMappings	query	Return the value mappings for current value map in the mappings property. Supports `count`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `valuemapid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	

Parameter	Type	Description
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 value maps

Retrieve all configured value maps.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.get",
    "params": {
        "output": "extend"
    },
    "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
}
```

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",
            "mappings": [
                {
                    "value": "1",
                    "newvalue": "unknown"
                },
                {
                    "value": "2",
                    "newvalue": "notInstalled"
                },
                {
                    "value": "3",
                    "newvalue": "ok"
                },
                {
                    "value": "4",
                    "newvalue": "failed"
                },
                {
                    "value": "5",
                    "newvalue": "highTemperature"
                },
                {
                    "value": "6",
                    "newvalue": "replaceImmediately"
                },
                {
                    "value": "7",
                    "newvalue": "lowCapacity"
                }
            ]
        }
    ],
    "id": 1
}
```

Source

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.

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:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.update",
    "params": {
        "valuemapid": "2",
        "name": "Device status"
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "2"
        ]
    },
    "id": 1
}
```

Changing mappings for one value map.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.update",
    "params": {
        "valuemapid": "2",
        "mappings": [
            {
                "value": "0",
                "newvalue": "Online"
            },
            {
                "value": "1",
                "newvalue": "Offline"
            }
        ]
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "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.

Property	Type	Description
httptestid	string	*(readonly)* ID of the web scenario.
hostid (required)	string	ID of the host that the web scenario belongs to.
name (required)	string	Name of the web scenario.
agent	string	User agent string that will be used by the web scenario. Default: Zabbix
applicationid	string	ID of the application that the web scenario belongs to.
authentication	integer	Authentication method that will be used by the web scenario. Possible values: 0 - *(default)* none; 1 - basic HTTP authentication; 2 - NTLM authentication.
delay	string	Execution interval of the web scenario. Accepts seconds, time unit with suffix and user macro. Default: 1m.
headers	array of HTTP fields	HTTP headers that will be sent when performing a request.
http_password	string	Password used for basic HTTP or NTLM authentication.
http_proxy	string	Proxy that will be used by the web scenario given as *http://[username[:password]@]proxy.example.com[:port]*.
http_user	string	User name used for basic HTTP or NTLM authentication.
nextcheck	timestamp	*(readonly)* Time of the next web scenario execution.
retries	integer	Number of times a web scenario will try to execute each step before failing. Default: 1.

Property	Type	Description
ssl_cert_file	string	Name of the SSL certificate file used for client authentication (must be in PEM format).
ssl_key_file	string	Name of the SSL private key file used for client authentication (must be in PEM format).
ssl_key_password	string	SSL private key password.
status	integer	Whether the web scenario is enabled. Possible values are: 0 - *(default)* enabled; 1 - disabled.
templateid	string	*(readonly)* ID of the parent template web scenario.
variables	array of HTTP fields	Web scenario variables.
verify_host	integer	Whether to validate that the host name for the connection matches the one in the host's certificate. Possible values are: 0 - *(default)* skip host verification; 1 - verify host.
verify_peer	integer	Whether to validate that the host's certificate is authentic. Possible values are: 0 - *(default)* skip peer verification; 1 - verify peer.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Scenario step

The scenario step object defines a specific web scenario check. It has the following properties.

Property	Type	Description
httpstepid	string	*(readonly)* ID of the scenario step.
name (required)	string	Name of the scenario step.
no (required)	integer	Sequence number of the step in a web scenario.
url (required)	string	URL to be checked.
follow_redirects	integer	Whether to follow HTTP redirects. Possible values are: 0 - don't follow redirects; 1 - *(default)* follow redirects.
headers	array of HTTP fields	HTTP headers that will be sent when performing a request. Scenario step headers will overwrite headers specified for the web scenario.
httptestid	string	*(readonly)* ID of the web scenario that the step belongs to.
posts	string array of HTTP fields	HTTP POST variables as a string (raw post data) or as an array of HTTP fields (form field data).
required	string	Text that must be present in the response.
retrieve_mode	integer	Part of the HTTP response that the scenario step must retrieve. Possible values are: 0 - *(default)* only body; 1 - only headers; 2 - headers and body.
status_codes	string	Ranges of required HTTP status codes separated by commas.
timeout	string	Request timeout in seconds. Accepts seconds, time unit with suffix and user macro. Default: 15s. Maximum: 1h. Minimum: 1s.
variables	array of HTTP fields	Scenario step variables.

Property	Type	Description
query_fields	array of HTTP fields	Query fields - array of HTTP fields that will be added to URL when performing a request

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.

Property	Type	Description
name (required)	string	Name of header / variable / POST or GET field.
value (required)	string	Value of header / variable / POST or GET field.

httptest.create

Description

`object httptest.create(object/array webScenarios)`

This method allows to create new web scenarios.

> **Note:**
> Creating a web scenario will automatically create a set of web monitoring items.

Parameters

`(object/array)` Web scenarios to create.

Additionally to the standard web scenario properties, the method accepts the following parameters.

Parameter	Type	Description
steps (required)	array	Web scenario steps.

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:

```
{
    "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",
    "id": 1
}
```

Response:

```
{
    "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.

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:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.delete",
    "params": [
        "2",
        "3"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "httptestids": [
            "2",
            "3"
        ]
    },
    "id": 1
}
```

Source

CHttpTest::delete() in *ui/include/classes/api/services/CHttpTest.php*.

httptest.get

Description

`integer/array httptest.get(object parameters)`

The method allows to retrieve web scenarios according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
applicationids	string/array	Return only web scenarios that belong to the given applications.
groupids	string/array	Return only web scenarios that belong to the given host groups.
hostids	string/array	Return only web scenarios that belong to the given hosts.
httptestids	string/array	Return only web scenarios with the given IDs.
inherited	boolean	If set to `true` return only web scenarios inherited from a template.
monitored	boolean	If set to `true` return only enabled web scenarios that belong to monitored hosts.
templated	boolean	If set to `true` return only web scenarios that belong to templates.
templateids	string/array	Return only web scenarios that belong to the given templates.
expandName	flag	Expand macros in the name of the web scenario.
expandStepName	flag	Expand macros in the names of scenario steps.
selectHosts	query	Return the hosts that the web scenario belongs to as an array in the hosts property.
selectSteps	query	Return web scenario steps in the steps property. Supports `count`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `httptestid` and `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
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 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",
            "applicationid": "0",
            "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": [],
            "steps": [
                {
                    "httpstepid": "36",
                    "httptestid": "9",
                    "name": "Homepage",
                    "no": "1",
                    "url": "http://example.com",
                    "timeout": "15s",
                    "posts": "",
                    "required": "",
                    "status_codes": "200",
                    "variables": [
                        {
                            "name":"{var}",
                            "value":"12"
```

```
                        }
                    ],
                    "follow_redirects": "1",
                    "retrieve_mode": "0",
                    "headers": [],
                    "query_fields": []
                },
                {
                    "httpstepid": "37",
                    "httptestid": "9",
                    "name": "Homepage / About",
                    "no": "2",
                    "url": "http://example.com/about",
                    "timeout": "15s",
                    "posts": "",
                    "required": "",
                    "status_codes": "200",
                    "variables": [],
                    "follow_redirects": "1",
                    "retrieve_mode": "0",
                    "headers": [],
                    "query_fields": []
                }
            ]
        }
    ],
    "id": 1
}
```


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.

Parameters

`(object/array)` 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.

Parameter	Type	Description
steps	array	Scenario steps to replace existing steps.

Return values

`(object)` 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:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.update",
    "params": {
        "httptestid": "5",
        "status": 0
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 1
}
```

Response:

```
{
    "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:

Type	Description
boolean	A boolean value, accepts either `true` or `false`.
flag	The value is considered to be `true` if it is passed and not equal to `null`; otherwise, it is considered to be `false`.
integer	A whole number.
float	A floating point number.
string	A text string.
text	A longer text string.
timestamp	A Unix timestamp.
array	An ordered sequence of values, that is, a plain array.
object	An associative array.
query	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.

Attention:
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:

Parameter	Type	Description
countOutput	boolean	Return the number of records in the result instead of the actual data.
editable	boolean	If set to `true`, return only objects that the user has write permissions to. Default: `false`.
excludeSearch	boolean	Return results that do not match the criteria given in the `search` parameter.
filter	object	Return only those results that exactly match the given filter. Accepts an object, where the keys are property names, and the values are either a single value or an array of values to match against. Does not support properties of `text` data type.
limit	integer	Limit the number of records returned.
output	query	Object properties to be returned. Default: `extend`.
preservekeys	boolean	Use IDs as keys in the resulting array.
search	object	Return results that match the given pattern (case-insensitive). Accepts an object, 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 "%…%"` search. Supports only properties of `string` and `text` data type.
searchByAny	boolean	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`.
searchWildcardsEnabled	boolean	If set to `true`, enables the use of "*" as a wildcard character in the `search` parameter. Default: `false`.
sortfield	string/array	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.
sortorder	string/array	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.
startSearch	boolean	The `search` parameter will compare the beginning of fields, that is, perform a `LIKE "…%"` search instead. Ignored if `searchWildcardsEnabled` is set to `true`.

Examples User permission check

Does the user have permission to write to hosts whose names begin with "MySQL" or "Linux" ?

Request:

```
{
    "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
}
```

Note:
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*"
        },
        "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",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        },
        "99061": {
            "hostid": "99061",
            "host": "Linux server 01",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        }
    },
    "id": 1
}
```

Appendix 2. Changes from 4.4 to 5.0

Backward incompatible changes General

ZBX-18998 added more strict validation for JSON-RPC structure.

action

Changes:
ZBXNEXT-5548 dropped support of `def_longdata`, `def_shortdata`, `r_longdata`, `r_shortdata`, `ack_longdata`, `ack_shortdata` properties.

item, template

Changes:
ZBXNEXT-5596 dropped support of item properties `port`, `snmp_community`, `snmpv3_authpassphrase`, `snmpv3_authprotocol`, `snmpv3_contextname`, `snmpv3_privpassphrase`, `snmpv3_privprotocol`, `snmpv3_securitylevel`, `snmpv3_securityname` and added same properties for host interfaces. Added item `type` 20 - SNMP agent. Removed item types 1 - SNMPv1 agent, 4 - SNMPv2 agent, 6 - SNMPv3 agent

Other changes and bug fixes action

Changes:
ZBXNEXT-5548 changed default for `default_msg` in opmessage object from 0 to 1.

auditlog

Changes:
ZBXNEXT-4584 added a new auditlog API introducing a new method `auditlog.get`.

event

Changes:
ZBXNEXT-1882 `event.acknowledge`: added new option in `action` that allows to unacknowledge event.

host

Bug fixes:
ZBXNEXT-5694 `host.get`: fixed option `selectScreens` with count output.
Changes:
ZBXNEXT-5694 `host.get`: added new option `withProblemsSuppressed` which return hosts that have suppressed problems (`true`), suppressed problems (`false`) or all hosts (`null` - default).
ZBXNEXT-5694 `host.get`: added new option `severities` which returns hosts with given problem severities.
ZBXNEXT-5694 `host.get`: added new option `inheritedTags` which returns hosts by tags that are inherited from all linked templates.
ZBXNEXT-5694 `host.get`: added new option `selectInheritedTags` which returns inherited tags in `inheritedTags` property from all templates and their parents.

host interface

ZBXNEXT-5596 added `details` property to the relevant host interface methods.
ZBXNEXT-2297 `hostinterface.get`: new option `selectMacros` which returns user macros for host prototype.
ZBXNEXT-2297 `hostinterface.create`, `hostinterface.update`: added new property `macros`.

discoveryrule

Changes:
ZBXNEXT-3035 added support of overrides.
ZBXNEXT-5811 added support of preprocessing `type` value "25".
ZBXNEXT-5879 `discoveryrule.get`: added new filtering option `groupids` that allows to retrieve LLD rules of the given host groups.

graphprototype

Changes:
ZBXNEXT-3035 added new property `discover`.

hostprototype

Changes:
ZBXNEXT-3035 added new property `discover`.

item

Changes:
ZBXNEXT-5811 added support of preprocessing `type` value "25".

itemprototype

Changes:
ZBXNEXT-3035 added new property `discover`.
ZBXNEXT-5811 added support of preprocessing `type` value "25".

mediatype

Changes:
ZBXNEXT-5548 `mediatype.create`, `mediatype.update`: added new property `message_templates`.
ZBXNEXT-5548 `mediatype.get`: added new option `selectMessageTemplates` that returns message templates in the `message_templates` property.

triggerprototype

Changes:
ZBXNEXT-3035 added new property `discover`.

user macro

ZBXNEXT-2957 `usermacro.create`, `usermacro.createglobal`, `usermacro.get`, `usermacro.update`, `usermacro.updateglobal`: added new property `type`.

ZBXNEXT-5849 usermacro.get: added filter by value

Zabbix API changes in 5.0

5.0.47 alert

Changes:
ZBX-26023 alert.get: *Admin* and *User* type users may now retrieve "message" (0) type alert data about users in their user group.

5.0.46 alert

Changes:
ZBX-26258 alert.get: Parameter selectUsers now returns data based on the restrictions added to user.get method.
ZBX-26258 alert.get: *Admin* and *User* type users may now retrieve "message" (0) type alert data only about their own user.

mediatype

Changes:
ZBX-26258 mediatype.get: parameter selectUsers now returns data based on the restrictions added to user.get method.

user

Changes:
ZBX-26258 user.get: when requesting user media or permissions, *Admin* and *User* type users may retrieve data only about their own user.
ZBX-26258 user.get: *Admin* and *User* type users may now retrieve only the following User object properties for their own user: userid, alias, attempt_clock, attempt_failed, attempt_ip, autologin, autologout, lang, name, refresh, rows_per_page, surname, theme, type, url. For users in their user group: userid, alias, name, surname.

usergroup

Changes:
ZBX-26258 usergroup.get: parameter selectUsers now returns data based on the restrictions added to user.get method.

5.0.44 mediatype

Changes:
ZBX-25385 mediatype.get: parameter selectMessageTemplates is now supported only for *Super admin* type users.
ZBX-25385 mediatype.get: *Admin* type users may now retrieve only the following Media type object properties: mediatypeid, name, type, status, maxattempts.
ZBX-25385 mediatype.get: when requesting user-related information of media types, *Admin* type users may now retrieve only data about their own user.

5.0.26 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.

5.0.16 usergroup

Bug fixes:
ZBX-19857 `usergroup.get`: dropped support for the non-working option `web_gui_access`

5.0.13 configuration

Bug fixes:
ZBX-8999 `configuration.export`: fixed exporting of images separately from other objects

graph

Bug fixes:
ZBX-19200 `graph.get` removed `discover` field from request
ZBX-19388 `graph.update`: fixed method to properly change values on template graph instead of making a new inherited graph if case user has no permissions to child host or template

graphprototype

Bug fixes:
ZBX-19388 `graphprototype.update`: fixed method to properly change values on template graph prototype instead of making a new inherited graph prototype if case user has no permissions to child host or template

host

ZBX-19200 `host.get` removed `discover` field from request

item

ZBX-19200 `item.get` removed `discover` field from request

mediatype

Changes:
ZBXNEXT-6582 increased `maxattempts` from 10 to 100 and `attempt_interval` from 60s to 1h

task

Bug fixes:
ZBX-18941 `task.create`: improved error message by adding item or discovery rule and host name if any of them are not monitored for task with type value 6

template

ZBX-19200 `template.get` removed `discover` field from request

trigger

Bug fixes:
ZBX-19200 `trigger.get` removed `discover` field from request
ZBX-19424 `trigger.create`: fixed trigger creation on PostgreSQL with host name consisting of only numbers

5.0.5 task

Changes:
ZBXNEXT-6167 added a new method `task.get`.
ZBXNEXT-6167 added a new type of task for `task.get` method.
ZBXNEXT-6167 changed format of `task.create` request. See task.create for more details.

20. Modules

Overview It is possible to enhance Zabbix frontend functionality by adding third-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:

- third-party modules can be harmful. You must trust the modules you are installing;
- Errors in a third-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
Module did not appear in the list	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.
Frontend crashed	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.
Error message about identical namespace, ID or actions appears	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.
Technical error messages appear	Report errors to the developer of the module.

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/                                 JavaScript files used in views.
            example.something.view.js.php
    partials/                               View partial files.
        example.something.reusable.php
        js/                                 JavaScript files used in partials.
            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.

Item	Naming rules	Example
Module directory	Lowercase [a-z], underscore and decimal digits	example_v2
Action subdirectories	Lowercase [a-z] and underscore character	data_export
Action files	CamelCase, ending with action type	SomethingView.php
View and partial files	Lowercase [a-z] Words separated with dot Prefixed by `module.` followed by module name Ending with action type and .php file extension	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:

Parameter	Required	Type	Default	Description
manifest_version	Yes	Double	-	Manifest version of the module. Currently supported version is **1**.
id	Yes	String	-	Module ID. Only one module with given ID can be enabled at the same time.
name	Yes	String	-	Module name as displayed in the Administration section.
version	Yes	String	-	Module version as displayed in the Administration section.
namespace	Yes	String	-	PHP namespace for Module.php and action classes.
author	No	String	""	Module author as displayed in the Administration section.
url	No	String	""	Module URL as displayed in the Administration section.
description	No	String	""	Module description as displayed in the Administration section.

Parameter	Required	Type	Default	Description
actions	No	Object	{}	Actions to register with this module. See Actions.
config	No	Object	{}	Module configuration.

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:

Parameter	Required	Type	Default	Description
key	Yes	String	-	Action name, in lowercase [a-z], separating words with dot.
class	Yes	String	-	Action class name, including subdirectory path (if used) within the `actions` directory.
layout	No	String	”layout.htmlpage”	Action layout.
view	No	String	null	Action view.

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

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

```
{
    "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.
     *
     * @return void
     */
    public function init(): void {
        /**
         * Disable SID (Session ID) validation. Session ID validation should only be used for actions whic
         * modification, such as update or delete actions. In such case Session ID must be presented in th
         * 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
     *
     * @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 hosts.disable_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. For more information, see *Macro functions*.

If you haven't found an 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.

> **Note:**
> If installing from Zabbix Git repository, you need to run the following command prior to proceeding to the next steps:
>

 `$ make dbschema`

MySQL

Character set utf8 and utf8_bin collation is required for Zabbix server/proxy to work properly with MySQL database.

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.

```
mysql -uroot -p<password>

mysql> create database zabbix character set utf8 collate utf8_bin;
mysql> create user 'zabbix'@'localhost' identified by '<password>';
mysql> grant all privileges on zabbix.* to 'zabbix'@'localhost';
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).

```
cd database/mysql
mysql -uzabbix -p<password> zabbix < schema.sql
#### stop here if you are creating database for Zabbix proxy
mysql -uzabbix -p<password> zabbix < images.sql
mysql -uzabbix -p<password> zabbix < data.sql
```

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):

```
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`).

```
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).

For a Zabbix proxy database, only `schema.sql` should be imported (no images.sql nor data.sql).

```
cd database/postgresql
cat schema.sql | sudo -u zabbix psql zabbix
#### stop here if you are creating database for Zabbix proxy
cat images.sql | sudo -u zabbix psql zabbix
cat data.sql | sudo -u zabbix psql zabbix
```

Attention:
The above commands are provided as an example that will work in most of GNU/Linux installations. You can use different commands depending on how your system/database is configured, for example:

 `psql -U <username>`

 If you have any trouble setting up the database, please consult your Database administrator.

TimescaleDB

Instructions for creating and configuring TimescaleDB are provided in a separate section.

Oracle

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 with a *user* shell user having write access to /tmp directory. 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
```

If you are creating a database for Zabbix server you need to have images from Zabbix sources on the host where Oracle is running. Copy them to a directory */tmp/zabbix_images* on the Oracle host:

```
cd /path/to/zabbix-sources
ssh user@oracle_host "mkdir /tmp/zabbix_images"
scp -r misc/images/png_modern user@oracle_host:/tmp/zabbix_images/
```

Now prepare the database:

```
cd /path/to/zabbix-sources/database/oracle
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
```

> **Note:**
> Please set the initialization parameter CURSOR_SHARING=FORCE for best performance.

Now the temporary directory can be removed:

```
ssh user@oracle_host "rm -rf /tmp/zabbix_images"
```

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.

Additional patches

In some cases, Zabbix might generate non-optimized queries to the database. This may happen, for example, as a result of using an older database version.
Additional DB patches are available for resolving such issues on the specific database or, sometimes, a specific database version. See Known issues for the list of known problems and available patches.

2 Repairing Zabbix database character set and collation

MySQL/MariaDB

1. Check the database character set and collation.

For example:

```
mysql> SELECT @@character_set_database, @@collation_database;
+--------------------------+----------------------+
| @@character_set_database | @@collation_database |
+--------------------------+----------------------+
| utf8mb4                  | utf8mb4_general_ci   |
+--------------------------+----------------------+
```

As we see, the character set here is not 'utf8' and collation is not 'utf8_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 utf8 collate utf8_bin;
```

Fixed values:

```
mysql> SELECT @@character_set_database, @@collation_database;
+--------------------------+----------------------+
| @@character_set_database | @@collation_database |
+--------------------------+----------------------+
| utf8                     | utf8_bin             |
+--------------------------+----------------------+
```

5. Load the script to fix character set and collation on table and column level:

```
mysql <your DB name> < utf8_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;
```

Note that data encoding will be changed on disk. For example, when converting characters like Æ, Ñ, Ö from 'latin1' to 'utf8' they will go from 1 byte to 2 bytes. Thus the repaired database may require more space than before.

7. If no errors - you may want to create a database backup copy with the fixed database.

8. Start Zabbix.

3 Secure connection to the database

Overview

It is possible to configure secure TLS connections to MySQL and PostgreSQL databases from:

- Zabbix frontend
- Zabbix server or proxy

See also: Known issues

Frontend configuration

> **Note:**
> Since Zabbix 5.0.5 TLS encryption parameter names have changed slightly: for better clarity the "Database" prefix has been added. In versions 5.0.0-5.0.4 parameters are named *TLS encryption*, *TLS certificate file*, etc.

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.

> **Note:**
> Since Zabbix 5.0.5:
> 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 are available in the TLS encryption in certificates mode (since Zabbix 5.0.5 the parameters appear only if both checkboxes are marked):

Parameter	Description
Database TLS CA file	Specify the full path to a valid TLS certificate authority (CA) file.
Database TLS key file	Specify the full path to a valid TLS key file.
Database TLS certificate file	Specify the full path to a valid TLS certificate file.
Database host verification	Mark this checkbox to activate host verification. Disabled for MYSQL, because PHP MySQL library does not allow to skip the peer certificate validation step.
Database TLS cipher list	Specify a custom list of valid ciphers. The format of the cipher list must conform to the OpenSSL standard. Available for MySQL only.

Attention:
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

Configuration	Result
None (leave *Database TLS encryption* unmarked)	Connection to the database without encryption.
1. Mark *Database TLS encryption* only	Secure TLS connection to the database.
1. Mark *Database TLS encryption* 2. Specify TLS certificate authority file	Secure TLS connection to the database; Database server certificate is verified and verified that it is signed by a trusted center.

Configuration	Result
1. Mark *Database TLS encryption* 2. Specify TLS certificate authority file 3. Mark *With host verification* 4. Specify TLS cipher list (optional)	Secure TLS connection to the database; Database server certificate is checked by comparing the host name specified in the certificate with the name of the host to which it is connected; It is verified that the certificate is signed by a trusted authority.
1. Mark *Database TLS encryption* 2. Specify TLS key file 3. Specify TLS certificate file 4. Specify TLS certificate authority file 5. Mark *Database host verification* (prior to 5.0.5: *With host verification*) 6. Specify TLS cipher list (optional)	Secure TLS connections to the database are established with maximum security. The requirement for the client part to present their certificates is configured on the server side.

See also: Configuration examples for MySQL, 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.

Configuration	Result
None	Connection to the database without encryption.
1. Set DBTLSConnect=required	Server/proxy make a TLS connection to the database. An unencrypted connection is not allowed.
1. Set DBTLSConnect=verify_ca 2. Set DBTLSCAFile - specify the TLS certificate authority file	Server/proxy make a TLS connection to the database after verifying the database certificate.
1. Set DBTLSConnect=verify_full 2. Set DBTLSCAFile - specify TLS certificate authority file	Server/proxy make a TLS connection to the database after verifying the database certificate and the database host identity.
1. Set DBTLSCAFile - specify TLS certificate authority file 2. Set DBTLSCertFile - specify the client public key certificate file 3. Set DBTLSKeyFile - specify the client private key file	Server/proxy provide a client certificate while connecting to the database.
1. Set DBTLSCipher - the list of encryption ciphers that the client permits for connections using TLS protocols up to TLS 1.2 or DBTLSCipher13 - the list of encryption ciphers that the client permits for connections using TLS 1.3 protocol	(MySQL) TLS connection is made using a cipher from the provided list. (PostgreSQL) Setting this option will be considered as an error.

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 quick start guide for encrypting the connection to the database. List of encryption combinations is not limited to the ones listed on this page. There are a lot more combinations available.

Attention:
Since Zabbix 5.0.5, 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.

Note:
Since Zabbix 5.0.5 TLS encryption parameter names in the frontend have changed slightly: for better clarity the "Database" prefix has been added. In versions 5.0.0-5.0.4 parameters are named *TLS encryption*, *TLS certificate file*, etc.

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):

```
mysql> SELECT sbt.variable_value AS tls_version, t2.variable_value AS cipher, processlist_user AS user, pr
        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:

```
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;

mysql> CREATE ROLE 'zbx_srv_role', 'zbx_web_role';

mysql> GRANT SELECT, UPDATE, DELETE, INSERT, CREATE, DROP, ALTER, INDEX, REFERENCES ON zabbix.* TO 'zbx_sr
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):

```
$ mysql -u zbx_srv -p -h 10.211.55.9 --ssl-mode=REQUIRED
```

Check current status and available cipher suites:

```
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
Value: TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_128_CCM_SHA256:E
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*:

```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=required
...
```

Verify CA mode

Copy required MySQL CA to the Zabbix frontend server, assign proper permissions to allow the webserver to read this file.

> **Note:**
> *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*:

```
...
$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*:

```
...
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 or Creating SSL certificates and keys using openssl

Attention:
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=
```

Frontend

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.

Warning:
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*:

```
...
// 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:TLS_AES_1
...
// 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*:

```
...
DBHost=10.211.55.9
DBName=zabbix
```

```
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=verify_full
DBTLSCAFile=/etc/ssl/mysql/ca.pem
DBTLSCertFile=/etc/ssl/mysql/client-cert.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.

Note:
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.

Note:
Since Zabbix 5.0.5 TLS encryption parameter names in the frontend have changed slightly: for better clarity the "Database" prefix has been added. In versions 5.0.0-5.0.4 parameters are named *TLS encryption*, *TLS certificate file*, etc.

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:

```
...
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*:

```
...
### 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

Server

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
...
```

Verify CA mode

Frontend

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:*

```
...
$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:*

```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=verify_ca
DBTLSCAFile=/etc/ssl/pgsql/root.crt
...
```

Verify full mode

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
DBTLSCAFile=/etc/ssl/pgsql/root.crt
DBTLSCertFile=/etc/ssl/pgsql/client.crt
DBTLSKeyFile=/etc/ssl/pgsql/client.key
...
```

4 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.

Warning:
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 in Timescale documentation).

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.

Note:
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 zabbix`

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 timescaledb.sql | sudo -u zabbix psql zabbix
```

Attention:
Please ignore warning messages informing that the best practices are not followed while running `timescaledb.sql` script on TimescaleDB version 2.9.0 and higher. Regardless of this warning, the configuration will be completed successfully.

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 enabled. It is also possible to enable override individually either for history only or trends only.

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

To successfully remove compressed data by housekeeper, both *Override item history period* and *Override item trend period* options must be enabled. If override is disabled and tables have compressed chunks, the housekeeper will not remove data from these tables, and warnings about incorrect configuration will be displayed in the administration screen for *Housekeeping* and the *System information* section.

All of these parameters can be changed in *Administration → General → Housekeeping* after the installation.

Note:
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 5.0.26, and for TimescaleDB versions 2.0+, Zabbix detects if compression is supported. If it is not supported a warning message is written into the Zabbix server log.

Note:
Users are encouraged to get familiar with compression in Timescale 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.

Parameter	Default	Comments
Enable compression	Enabled	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 zabbix_server.conf) 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 the instructions in Timescale documentation.
Compress records older than	7d	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.

5 Elasticsearch setup

Attention:
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.

Warning:
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:

```
### 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:

Item value type	Database table	Elasticsearch type
Numeric (unsigned)	history_uint	uint
Numeric (float)	history	dbl
Character	history_str	str
Log	history_log	log
Text	history_text	text

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.

Note:
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`.

Warning:
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.

Note:
To work with Elasticsearch please refer to Requirement page for additional information.

Note:
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"
               }
```

```
            },
            "index": false,
            "type": "text"
         }
      }
   }
}'
```

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:

```
curl -X PUT \
 http://your-elasticsearch.here:9200/_ingest/pipeline/uint-pipeline \
 -H 'content-type:application/json' \
 -d '{
   "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:

```
### Option: HistoryStorageDateIndex
# Enable preprocessing of history values in history storage to store values in different indices based on
# 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.)

6 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.

Attention:
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. This parameter is supported since Zabbix 5.0.10.

7 Distribution-specific notes on setting up Nginx for Zabbix

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
```

8 Running agent as root

Since Zabbix **5.0.0**, the systemd service file for Zabbix agent in official packages explicitly includes directives for `User` and `Group`. Both are set to `zabbix`.

It is no longer possible to configure which user Zabbix agent runs as via `zabbix_agentd.conf` file, because the agent will bypass this configuration and run as the user specified in the systemd service file. To run Zabbix agent as root you need to make the modifications described below.

Zabbix agent

To override the default user and group for Zabbix agent, run:

```
systemctl edit zabbix-agent
```

Then, add the following content:

```
[Service]
User=root
Group=root
```

Reload daemons and restart the zabbix-agent service:

```
systemctl daemon-reload
systemctl restart zabbix-agent
```

For **Zabbix agent** this re-enables the functionality of configuring user in the `zabbix_agentd.conf` file. Now you need to set `User=root` and `AllowRoot=1` configuration parameters in the agent configuration file.

Zabbix agent 2

To override the default user and group for Zabbix agent 2, run:

```
systemctl edit zabbix-agent2
```

Then, add the following content:

```
[Service]
User=root
Group=root
```

Reload daemons and restart the zabbix-agent service:

```
systemctl daemon-reload
systemctl restart zabbix-agent2
```

For **Zabbix agent2** this completely determines the user that it runs as. No additional modifications are required.

9 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.

Warning:
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.

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 "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* - returns the full, real (case-sensitive) Windows host name;
- *shorthost* (supported since Zabbix 5.0.15) - 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 5.0.15 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
```

Attention:
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:

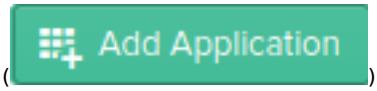
```
zabbix_agentd.exe --config <configuration_file_for_instance_1> --uninstall --multiple-agents
zabbix_agentd.exe --config <configuration_file_for_instance_2> --uninstall --multiple-agents
...
zabbix_agentd.exe --config <configuration_file_for_instance_N> --uninstall --multiple-agents
```

10 SAML setup with Okta

This section describes how to configure Okta to enable SAML 2.0 authentication for Zabbix.

Okta configuration

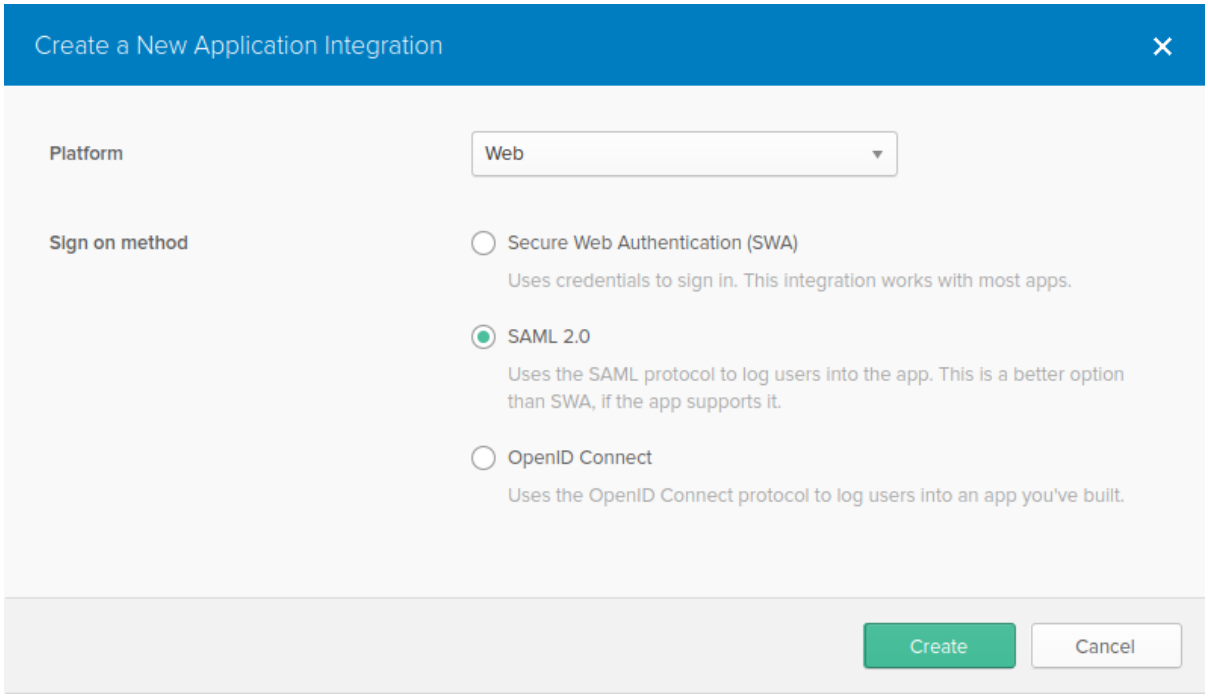
1. Go to https://okta.com and register or sign in to your account.

2. In the Okta web interface navigate to *Applications* → *Applications* and press "Add Application" button (

).

3. Press "Create New App" button (

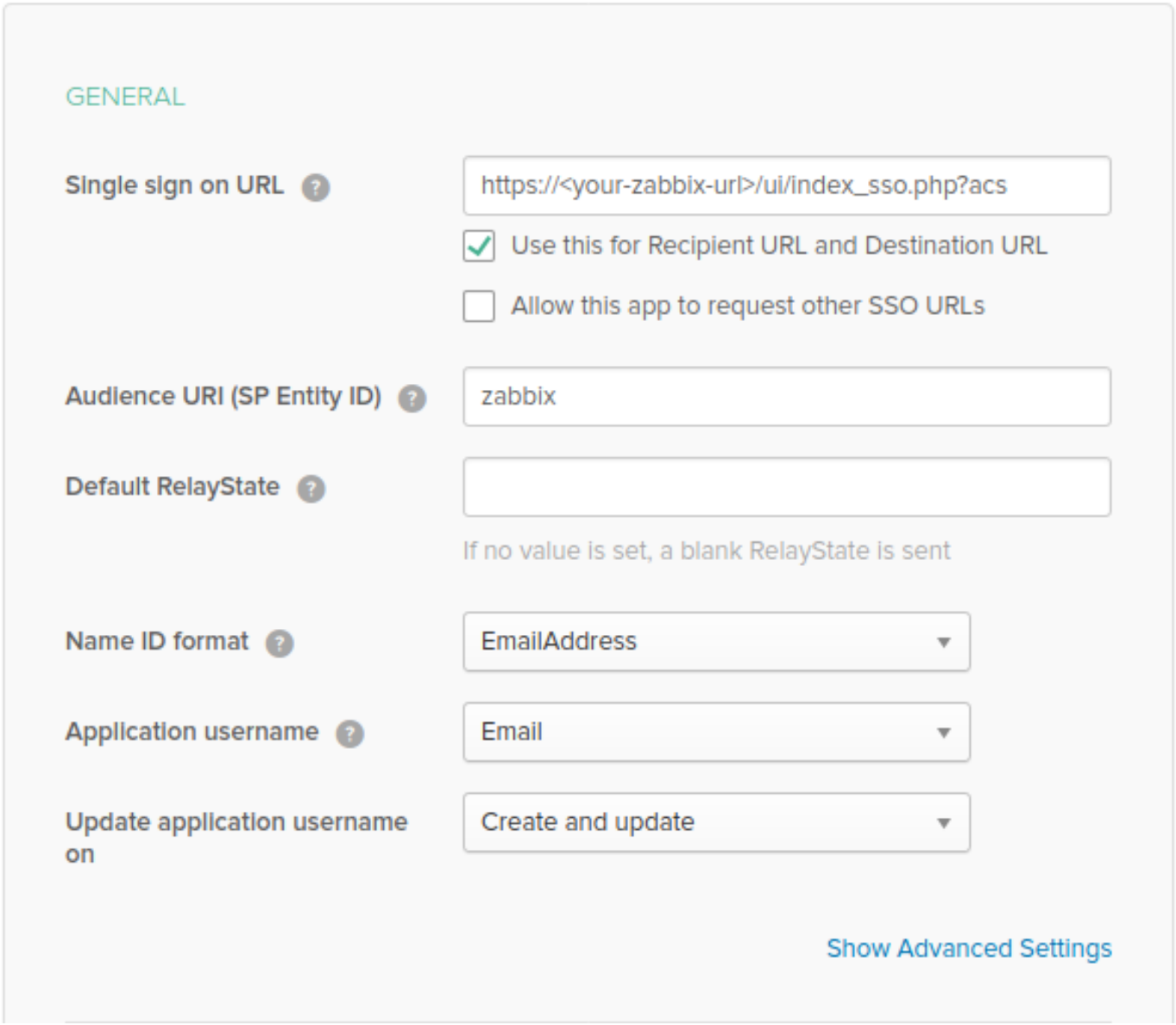
). In a popup window select *Platform:* Web, *Sign on method:* SAML 2.0 and press "Create" button.



4. Fill in the fields in the *General settings* tab (the first tab that appears) according to your preferences and press "Next".

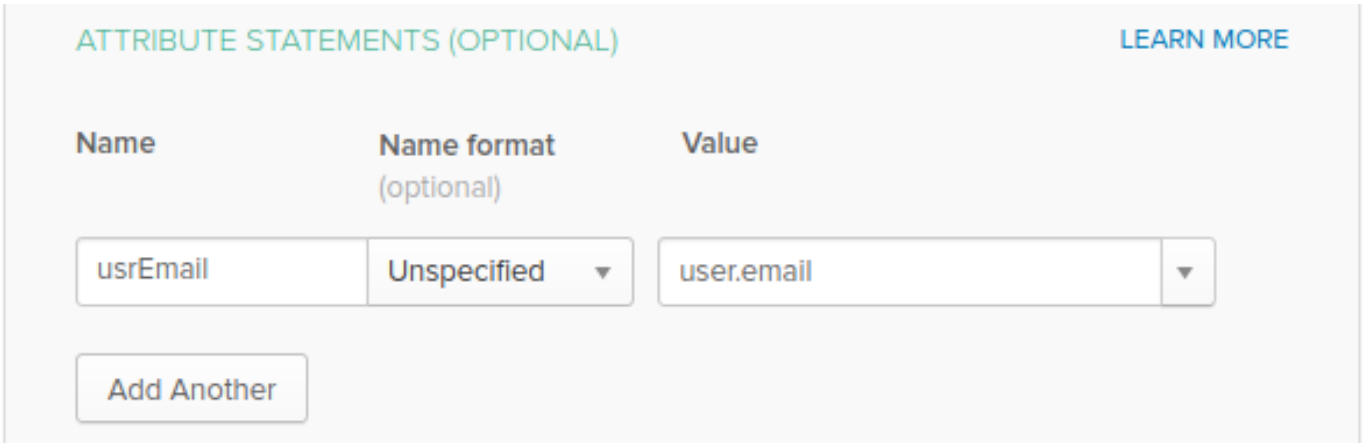
5. In the *Configure SAML* tab enter the values provided below, then press "Next".

- 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.



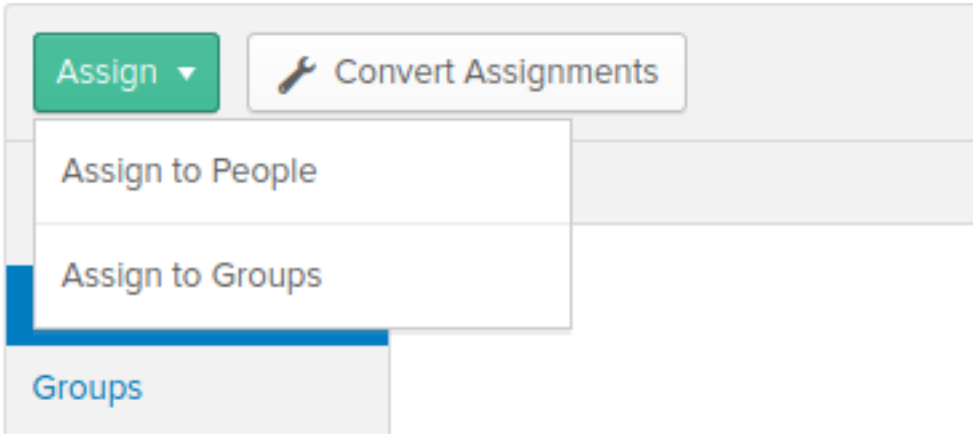
Note:
If planning to use encrypted connection, generate private and public encryption certificates, then upload public certificate to Okta. Certificate upload form appears when *Assertion Encryption* is set to Encrypted (click *Show Advanced Settings* to find this parameter).

- In the **ATTRIBUTE STATEMENTS (OPTIONAL)** section add an attribute statement with:
  - *Name:* usrEmail
  - *Name format:* Unspecified
  - *Value:* user.email



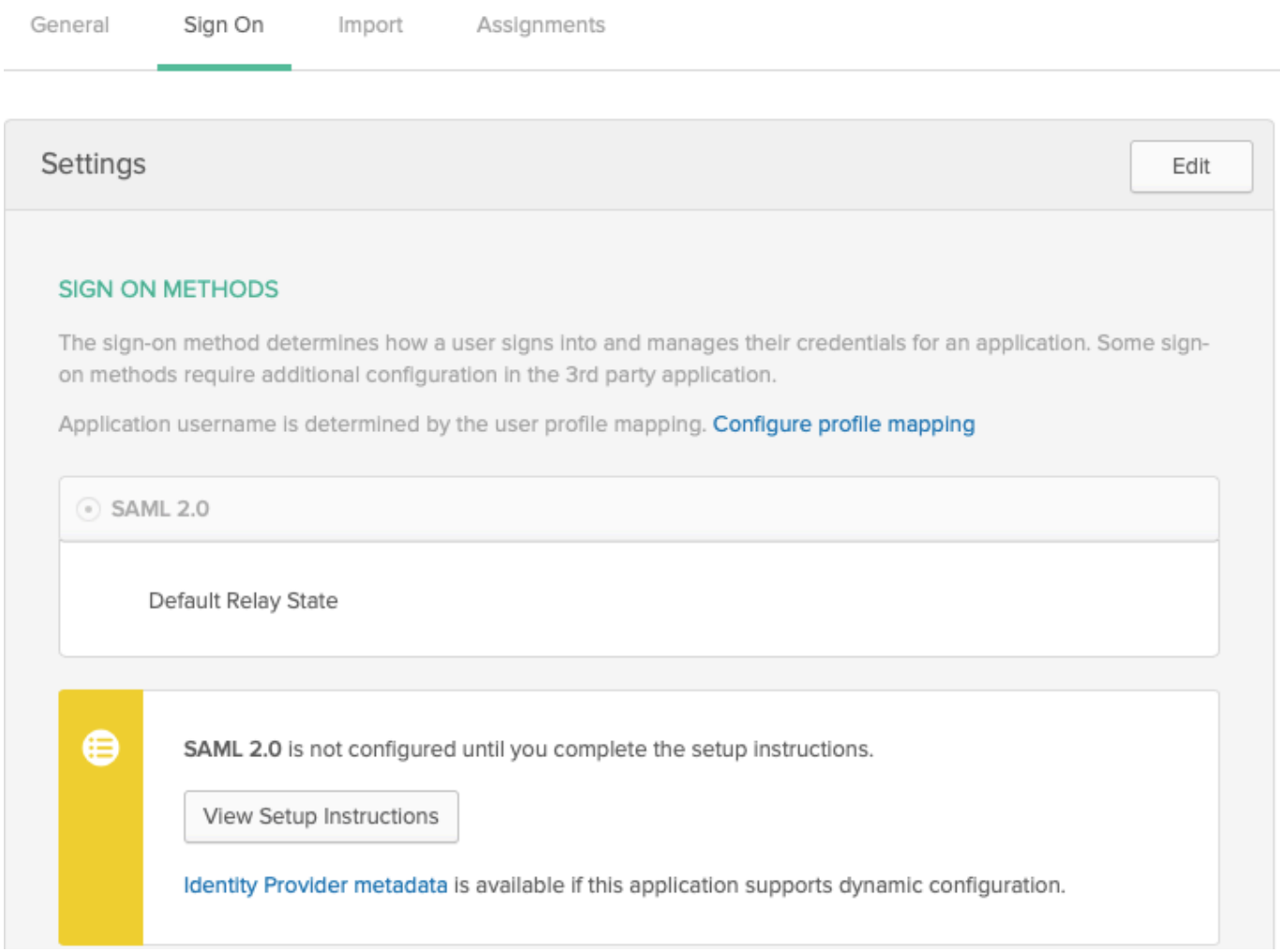
6. At the next tab, select ”I’m a software vendor. I’d like to integrate my app with Okta” and press ”Finish”.

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:

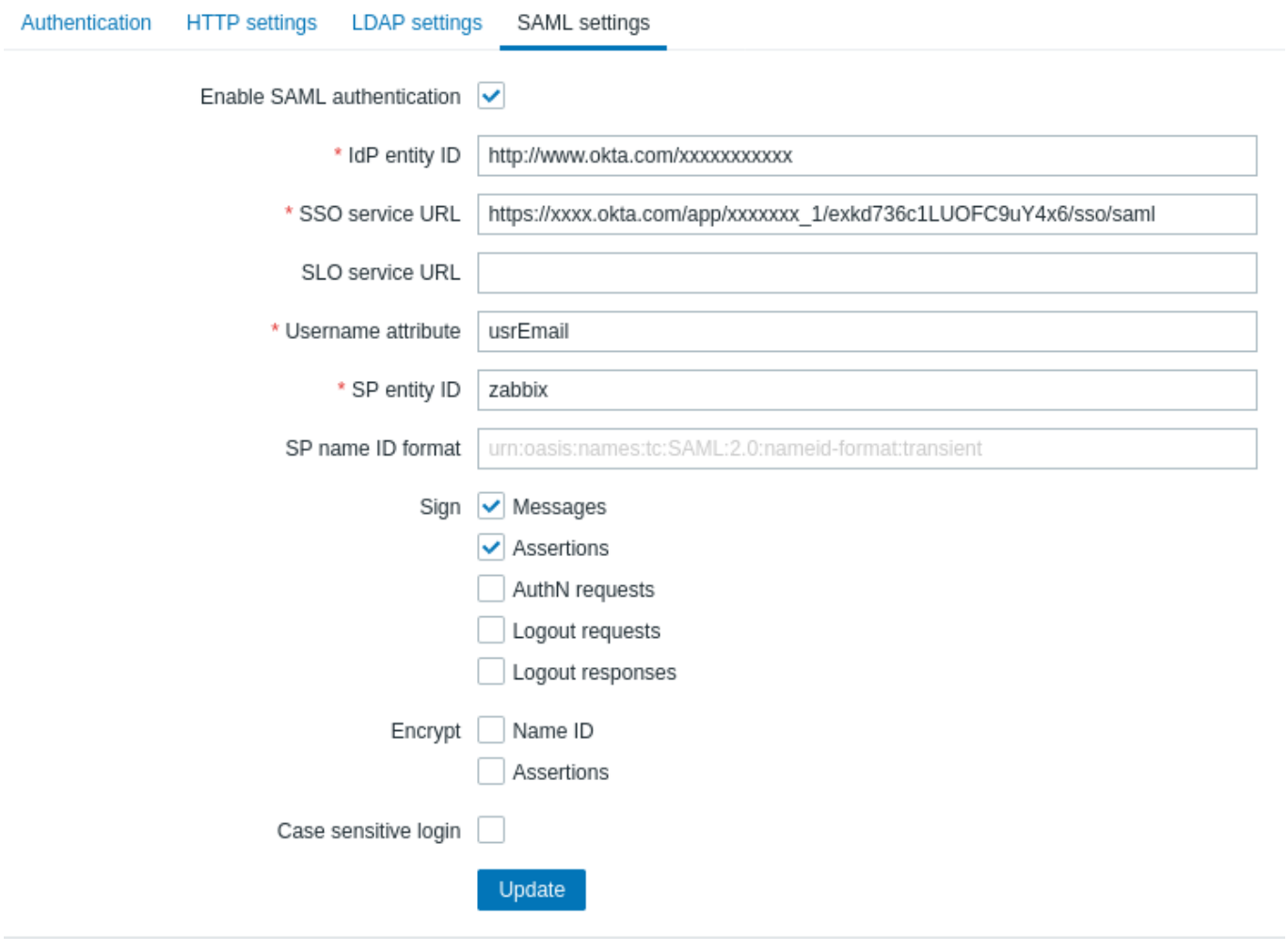
```
chmod 644 idp.crt
```

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):

```
// 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'] = []; // Additional 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.

Note:
To sign in with SAML, user alias in Zabbix should match his Okta e-mail. This settings can be changed in the *Administration → Users* section of Zabbix web interface.

11 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.

If a locale is installed, a language becomes available in the language selector in Zabbix web interface. Languages for which locales are not installed are greyed out and cannot be selected.

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.

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

Parameter	Mandatory	Range	Default	Description
AlertScriptsPath	no		/usr/local/share/zabbix/alertscripts	Location of custom alert scripts (depends on compile-time installation variable *datadir*).
AllowRoot	no		0	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 This parameter is supported since Zabbix 2.2.0.

Parameter	Mandatory	Range	Default	Description
CacheSize	no	128K-64G	8M	Size of configuration cache, in bytes. Shared memory size for storing host, item and trigger data. The maximum value of this parameter was increased from 8GB to 64GB in Zabbix 5.0.1.
CacheUpdateFrequency	no	1-3600	60	How often Zabbix will perform update of configuration cache, in seconds. See also runtime control options.
DBHost	no		localhost	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.
DBName	yes			Database name.
DBPassword	no			Database password. Comment this line if no password is used.
DBPort	no	1024-65535		Database port when not using local socket.
DBSchema	no			Schema name. Used for PostgreSQL.
DBSocket	no			Path to MySQL socket file.
DBUser	no			Database user.
DBTLSConnect	no			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: ”required”, ”verify_ca”, ”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.

Parameter	Mandatory	Range	Default	Description
DBTLSCAFile	no (yes, if DBTLSConnect set to one of: verify_ca, verify_full)			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.
DBTLSCertFile	no			Full pathname of file containing Zabbix server certificate for authenticating to database. This parameter is supported since Zabbix 5.0.0.
DBTLSKeyFile	no			Full pathname of file containing the private key for authenticating to database. This parameter is supported since Zabbix 5.0.0.
DBTLSCipher	no			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.
DBTLSCipher13	no			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.
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 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.

Parameter	Mandatory	Range	Default	Description
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 This parameter is supported since Zabbix 5.0.10.
ExternalScripts	no		/usr/local/share/zabbix/externalscripts	Location of external scripts (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!
HistoryCacheSize	no	128K-2G	16M	Size of history cache, in bytes. Shared memory size for storing history data.
HistoryIndexCacheSize	no	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	no		0	Enable preprocessing of history values in history storage to store values in different indices based on date: 0 - disable 1 - enable
HistoryStorageURL	no			History storage HTTP[S] URL. This parameter is used for Elasticsearch setup.
HistoryStorageTypes	no		uint,dbl,str,log,text	Comma separated list of value types to be sent to the history storage. This parameter is used for Elasticsearch setup.

Parameter	Mandatory	Range	Default	Description
HousekeepingFrequency	no	0-24	1	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. This postponing behavior is in place since Zabbix 2.4.0. 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.

Parameter	Mandatory	Range	Default	Description
Include	no			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/*.` Pattern matching is supported since Zabbix 2.4.0. See special notes about limitations.
JavaGateway	no			IP address (or hostname) of Zabbix Java gateway. Only required if Java pollers are started. This parameter is supported since Zabbix 2.0.0.
JavaGatewayPort	no	1024-32767	10052	Port that Zabbix Java gateway listens on. This parameter is supported since Zabbix 2.0.0.
ListenBacklog	no	0 - INT_MAX	SOMAXCONN	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'.
ListenIP	no		0.0.0.0	List of comma delimited IP addresses that the trapper should listen on. Trapper will listen on all network interfaces if this parameter is missing. Multiple IP addresses are supported since Zabbix 1.8.3.
ListenPort	no	1024-32767	10051	Listen port for trapper.

Parameter	Mandatory	Range	Default	Description
LoadModule	no			Module to load at server startup. Modules are used to extend functionality of the server. Formats: LoadModule=<module.so> LoadModule=<path/module.so> LoadModule=</abs_path/module.so> 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.
LoadModulePath	no			Full path to location of server modules. Default depends on compilation options.
LogFile	yes, if LogType is set to *file*, otherwise no			Name of log file.
LogFileSize	no	0-1024	1	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.
LogType	no		file	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.
LogSlowQueries	no	0-3600000	0	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. This parameter is supported since Zabbix 1.8.2.

Parameter	Mandatory	Range	Default	Description
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 is supported since Zabbix 1.8.2 and applies only to deleting history and trends of already deleted items.
PidFile	no		/tmp/zabbix_server.pid	Name of PID file.
ProxyConfigFrequency	no	1-604800	3600	How often Zabbix server sends configuration data to a Zabbix proxy in seconds. Used only for proxies in a passive mode. This parameter is supported since Zabbix 1.8.3.
ProxyDataFrequency	no	1-3600	1	How often Zabbix server requests history data from a Zabbix proxy in seconds. Used only for proxies in a passive mode. This parameter is supported since Zabbix 1.8.3.
SNMPTrapperFile	no		/tmp/zabbix_traps.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 SNMPTT configuration file. This parameter is supported since Zabbix 2.0.0.
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; - preprocessing JavaScript HTTP requests; - sending notification emails (connections to SMTP server); - webhook notifications (JavaScript HTTP connections)
SSHKeyLocation	no			Location of public and private keys for SSH checks and actions

Parameter	Mandatory	Range	Default	Description
SSLCertLocation	no			Location of SSL client certificate files for client authentication. This parameter is used in web monitoring only and is supported since Zabbix 2.4.
SSLKeyLocation	no			Location of SSL private key files for client authentication. This parameter is used in web monitoring only and is supported since Zabbix 2.4.
SSLCALocation	no			Override the location of certificate authority (CA) files for SSL server certificate verification. If not set, system-wide directory will be used. 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.
StartDBSyncers	no	1-100	4	Number of pre-forked instances of DB Syncers. *Note*: Be careful when changing this value, increasing it may do more harm than good. Roughly, the default value should be enough to handle up to 4000 NVPS. The upper limit used to be 64 before version 1.8.5. This parameter is supported since Zabbix 1.8.3.
StartAlerters	no	1-100	3	Number of pre-forked instances of alerters. This parameter is supported since Zabbix 3.4.0.
StartDiscoverers	no	0-250	1	Number of pre-forked instances of discoverers. The upper limit used to be 255 before version 1.8.5.
StartEscalators	no	1-100	1	Number of pre-forked instances of escalators. This parameter is supported since Zabbix 3.0.0.
StartHTTPPollers	no	0-1000	1	Number of pre-forked instances of HTTP pollers[1]. The upper limit used to be 255 before version 1.8.5.

Parameter	Mandatory	Range	Default	Description
StartIPMIPollers	no	0-1000	0	Number of pre-forked instances of IPMI pollers. The upper limit used to be 255 before version 1.8.5.
StartJavaPollers	no	0-1000	0	Number of pre-forked instances of Java pollers[1]. This parameter is supported since Zabbix 2.0.0.
StartLLDProcessors	no	1-100	2	Number of pre-forked instances of low-level discovery (LLD) workers[1]. The LLD manager process is automatically started when an LLD worker is started. This parameter is supported since Zabbix 4.2.0.
StartPingers	no	0-1000	1	Number of pre-forked instances of ICMP pingers[1]. The upper limit used to be 255 before version 1.8.5.
StartPollersUnreachable	no	0-1000	1	Number of pre-forked instances of pollers for unreachable hosts (including IPMI and Java)[1]. Since Zabbix 2.4.0, at least one poller for unreachable hosts must be running if regular, IPMI or Java pollers are started. The upper limit used to be 255 before version 1.8.5. This option is missing in version 1.8.3.
StartPollers	no	0-1000	5	Number of pre-forked instances of pollers[1]. *Note* that a non-zero value is required for internal, aggregated and calculated items to work.
StartPreprocessors	no	1-1000	3	Number of pre-forked instances of preprocessing workers[1]. The preprocessing manager process is automatically started when a preprocessor worker is started. This parameter is supported since Zabbix 3.4.0.
StartProxyPollers	no	0-250	1	Number of pre-forked instances of pollers for passive proxies[1]. The upper limit used to be 255 before version 1.8.5. This parameter is supported since Zabbix 1.8.3.
StartSNMPTrapper	no	0-1	0	If set to 1, SNMP trapper process will be started. This parameter is supported since Zabbix 2.0.0.

Parameter	Mandatory	Range	Default	Description
StartTimers	no	1-1000	1	Number of pre-forked instances of timers. Timers process maintenance periods. This parameter is supported since Zabbix 2.2.0.
StartTrappers	no	0-1000	5	Number of pre-forked instances of trappers[1]. Trappers accept incoming connections from Zabbix sender, active agents and active proxies. At least one trapper process must be running to display server availability and view queue in the frontend. The upper limit used to be 255 before version 1.8.5.
StartVMwareCollectors	no	0-250	0	Number of pre-forked vmware collector instances. This parameter is supported since Zabbix 2.2.0.
StatsAllowedIP	no			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 This parameter is supported since Zabbix 4.2.0.
Timeout	no	1-30	3	Specifies how long we wait for agent, SNMP device or external check (in seconds).
TLSCAFile	no			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.

Parameter	Mandatory	Range	Default	Description
TLSCertFile	no			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.
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 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 GnuTLS: NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL::+SIGN-ALL:+CTYPE-X.509 Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128 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 GnuTLS: NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509 Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128 This parameter is supported since Zabbix 4.4.7.

Parameter	Mandatory	Range	Default	Description
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.
TLSCipherPSK	no			GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for PSK-based encryption. Example for GnuTLS: NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL Example for OpenSSL: kECDHEPSK+AES128:kPSK+AES128 This parameter is supported since Zabbix 4.4.7.
TLSCipherPSK13	no			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 This parameter is supported since Zabbix 4.4.7.
TLSCRLFile	no			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.
TLSKeyFile	no			Full pathname of a file containing the server private key, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.
TmpDir	no		/tmp	Temporary directory.
TrapperTimeout	no	1-300	300	Specifies how many seconds trapper may spend processing new data.
TrendCacheSize	no	128K-2G	4M	Size of trend cache, in bytes. Shared memory size for storing trends data.
UnavailableDelay	no	1-3600	60	How often host is checked for availability during the unavailability period, in seconds.

Parameter	Mandatory	Range	Default	Description
UnreachableDelay	no	1-3600	15	How often host is checked for availability during the unreachability period, in seconds.
UnreachablePeriod	no	1-3600	45	After how many seconds of unreachability treat a host as unavailable.
User	no		zabbix	Drop privileges to a specific, existing user on the system. Only has effect if run as 'root' and AllowRoot is disabled. This parameter is supported since Zabbix 2.4.0.
ValueCacheSize	no	0,128K-64G	8M	Size of history value cache, in bytes. Shared memory size for caching item history data requests. Setting to 0 disables value cache (not recommended). When value cache runs out of the shared memory a warning message is written to the server log every 5 minutes. This parameter is supported since Zabbix 2.2.0.
VMwareCacheSize	no	256K-2G	8M	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). Note that shared memory is not allocated if there are no vmware collector instances configured to start. This parameter is supported since Zabbix 2.2.0.
VMwareFrequency	no	10-86400	60	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. This parameter is supported since Zabbix 2.2.0.
VMwarePerfFrequency	no	10-86400	60	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. This parameter is supported since Zabbix 2.2.9, 2.4.4

Parameter	Mandatory	Range	Default	Description
VMwareTimeout	no	1-300	10	The maximum number of seconds vmware collector will wait for a response from VMware service (vCenter or ESX hypervisor). This parameter is supported since Zabbix 2.2.9, 2.4.4

Footnotes

[1] Note that too many data gathering processes (pollers, unreachable pollers, HTTP pollers, Java pollers, pingers, trappers, proxy-pollers) together with IPMI manager, SNMP trapper and preprocessing workers can **exhaust** the per-process file descriptor limit for the preprocessing manager.

Warning:
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

Parameter	Mandatory	Range	Default	Description
AllowRoot	no		0	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. 0 - do not allow 1 - allow This parameter is supported since Zabbix 2.2.0.
CacheSize	no	128K-64G	8M	Size of configuration cache, in bytes. Shared memory size, for storing host and item data. The maximum value of this parameter was increased from 8GB to 64GB in Zabbix 5.0.1.

Parameter	Mandatory	Range	Default	Description
ConfigFrequency	no	1-604800	3600	How often proxy retrieves configuration data from Zabbix server in seconds. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).
DataSenderFrequency	no	1-3600	1	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).
DBHost	no		localhost	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.
DBName	yes			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:memdb1?mode=memory&cache=sha` Warning: Do not attempt to use the same database Zabbix server is using.
DBPassword	no			Database password. Ignored for SQLite. Comment this line if no password is used.
DBSchema	no			Schema name. Used for PostgreSQL.
DBSocket	no		3306	Path to MySQL socket.
DBUser				Database port when not using local socket. Ignored for SQLite. Database user. Ignored for SQLite.

Parameter	Mandatory	Range	Default	Description
DBTLSConnect	no			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: ”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.
DBTLSCAFile	no (yes, if DBTLSConnect set to one of: verify_ca, verify_full)			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.
DBTLSCertFile	no			Full pathname of file containing Zabbix server certificate for authenticating to database. This parameter is supported since Zabbix 5.0.0.
DBTLSKeyFile	no			Full pathname of file containing the private key for authenticating to database. This parameter is supported since Zabbix 5.0.0.
DBTLSCipher	no			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.
DBTLSCipher13	no			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.

Parameter	Mandatory	Range	Default	Description
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)
EnableRemoteCommands	no		0	Whether remote commands from Zabbix server are allowed. 0 - not allowed 1 - allowed This parameter is supported since Zabbix 3.4.0.
ExternalScripts	no		/usr/local/share/zabbix/externalscripts	Location of external scripts (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!
HeartbeatFrequency	no	0-3600	60	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).
HistoryCacheSize	no	128K-2G	16M	Size of history cache, in bytes. Shared memory size for storing history data.
HistoryIndexCacheSize	no	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.

Parameter	Mandatory	Range	Default	Description
Hostname	no		Set by HostnameItem	Unique, case sensitive Proxy name. Make sure the proxy name is known to the server! Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128
HostnameItem	no		system.hostname	Item 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[]. Ignored if Hostname is set. This parameter is supported since Zabbix 1.8.6.

Parameter	Mandatory	Range	Default	Description
HousekeepingFrequency	no	0-24	1	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. This postponing behavior is in place since Zabbix 2.4.0. 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.

Parameter	Mandatory	Range	Default	Description
Include	no			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/*.` Pattern matching is supported since Zabbix 2.4.0. See special notes about limitations.
JavaGateway	no			IP address (or hostname) of Zabbix Java gateway. Only required if Java pollers are started. This parameter is supported since Zabbix 2.0.0.
JavaGatewayPort	no	1024-32767	10052	Port that Zabbix Java gateway listens on. This parameter is supported since Zabbix 2.0.0.
ListenBacklog	no	0 - INT_MAX	SOMAXCONN	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'.
ListenIP	no		0.0.0.0	List of comma delimited IP addresses that the trapper should listen on. Trapper will listen on all network interfaces if this parameter is missing. Multiple IP addresses are supported since Zabbix 1.8.3.
ListenPort	no	1024-32767	10051	Listen port for trapper.

Parameter	Mandatory	Range	Default	Description
LoadModule	no			Module to load at proxy startup. Modules are used to extend functionality of the proxy. Formats: LoadModule=<module.so> LoadModule=<path/module.so> LoadModule=</abs_path/module.so> 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.
LoadModulePath	no			Full path to location of proxy modules. Default depends on compilation options.
LogFile	yes, if LogType is set to *file*, otherwise no			Name of log file.
LogFileSize	no	0-1024	1	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.
LogRemoteCommands	no		0	Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled This parameter is supported since Zabbix 3.4.0.
LogType	no		file	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.
LogSlowQueries	no	0-3600000	0	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. This parameter is supported since Zabbix 1.8.2.
PidFile	no		/tmp/zabbix_proxy.pid	Name of PID file.

Parameter	Mandatory	Range	Default	Description
ProxyLocalBuffer	no	0-720	0	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.
ProxyMode	no	0-1	0	Proxy operating mode. 0 - proxy in the active mode 1 - proxy in the passive mode This parameter is supported since Zabbix 1.8.3. *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.
ProxyOfflineBuffer	no	1-720	1	Proxy will keep data for N hours in case of no connectivity with Zabbix server. Older data will be lost.
ServerPort	no	1024-32767	10051	Port of Zabbix trapper on Zabbix server. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).
Server	yes			If ProxyMode is set to *active mode*: IP address or DNS name of Zabbix server to get configuration data from and send data to. If ProxyMode is set to *passive mode*: List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix server. Incoming connections will be accepted only from the addresses 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*: Server=127.0.0.1,192.168.1.0/24,::1,2001:d

Parameter	Mandatory	Range	Default	Description
SNMPTrapperFile	no		/tmp/zabbix_traps.tmp	Temporary file used for passing data from SNMP trap daemon to the proxy. Must be the same as in zabbix_trap_receiver.pl or SNMPTT configuration file. This parameter is supported since Zabbix 2.0.0.
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 server; - agentless connections (VMware, SSH, JMX, SNMP, Telnet and simple checks); - HTTP agent connections; - preprocessing JavaScript HTTP requests
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 and is supported since Zabbix 2.4.0.
SSLKeyLocation	no			Location of SSL private key files for client authentication. This parameter is used in web monitoring only and is supported since Zabbix 2.4.0.
SSLCALocation	no			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.
StartDBSyncers	no	1-100	4	Number of pre-forked instances of DB Syncers. *Note*: Be careful when changing this value, increasing it may do more harm than good. The upper limit used to be 64 before version 1.8.5. This parameter is supported since Zabbix 1.8.3.

Parameter	Mandatory	Range	Default	Description
StartDiscoverers	no	0-250	1	Number of pre-forked instances of discoverers. The upper limit used to be 255 before version 1.8.5.
StartHTTPPollers	no	0-1000	1	Number of pre-forked instances of HTTP pollers.
StartIPMIPollers	no	0-1000	0	Number of pre-forked instances of IPMI pollers. The upper limit used to be 255 before version 1.8.5.
StartJavaPollers	no	0-1000	0	Number of pre-forked instances of Java pollers. This parameter is supported since Zabbix 2.0.0.
StartPingers	no	0-1000	1	Number of pre-forked instances of ICMP pingers. The upper limit used to be 255 before version 1.8.5.
StartPollersUnreachable	no	0-1000	1	Number of pre-forked instances of pollers for unreachable hosts (including IPMI and Java). Since Zabbix 2.4.0, at least one poller for unreachable hosts must be running if regular, IPMI or Java pollers are started. The upper limit used to be 255 before version 1.8.5. This option is missing in version 1.8.3.
StartPollers	no	0-1000	5	Number of pre-forked instances of pollers. The upper limit used to be 255 before version 1.8.5.
StartPreprocessors	no	1-1000	3	Number of pre-forked instances of preprocessing workers[1]. The preprocessing manager process is automatically started when a preprocessor worker is started. This parameter is supported since Zabbix 4.2.0.
StartSNMPTrapper	no	0-1	0	If set to 1, SNMP trapper process will be started. This parameter is supported since Zabbix 2.0.0.
StartTrappers	no	0-1000	5	Number of pre-forked instances of trappers. Trappers accept incoming connections from Zabbix sender and active agents. The upper limit used to be 255 before version 1.8.5.
StartVMwareCollectors	no	0-250	0	Number of pre-forked vmware collector instances. This parameter is supported since Zabbix 2.2.0.

Parameter	Mandatory	Range	Default	Description
StatsAllowedIP	no			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 This parameter is supported since Zabbix 4.2.0.
Timeout	no	1-30	3	Specifies how long we wait for agent, SNMP device or external check (in seconds).
TLSAccept	yes for passive proxy, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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.
TLSCAFile	no			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.
TLSCertFile	no			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.

Parameter	Mandatory	Range	Default	Description
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 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 GnuTLS: NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL::+SIGN-ALL:+CTYPE-X.509 Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128 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 GnuTLS: NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509 Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128 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.

Parameter	Mandatory	Range	Default	Description
TLSCipherPSK	no			GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for PSK-based encryption. Example for GnuTLS: NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL Example for OpenSSL: kECDHEPSK+AES128:kPSK+AES128 This parameter is supported since Zabbix 4.4.7.
TLSCipherPSK13	no			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 This parameter is supported since Zabbix 4.4.7.
TLSConnect	yes for active proxy, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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.
TLSCRLFile	no			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.
TLSKeyFile	no			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.
TLSPSKFile	no			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.

Parameter	Mandatory	Range	Default	Description
TLSPSKIdentity	no			Pre-shared key identity string, used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.
TLSServerCertIssuer	no			Allowed server certificate issuer. This parameter is supported since Zabbix 3.0.0.
TLSServerCertSubject	no			Allowed server certificate subject. This parameter is supported since Zabbix 3.0.0.
TmpDir	no		/tmp	Temporary directory.
TrapperTimeout	no	1-300	300	Specifies how many seconds trapper may spend processing new data.
User	no		zabbix	Drop privileges to a specific, existing user on the system. Only has effect if run as 'root' and AllowRoot is disabled. This parameter is supported since Zabbix 2.4.0.
UnavailableDelay	no	1-3600	60	How often host is checked for availability during the unavailability period, in seconds.
UnreachableDelay	no	1-3600	15	How often host is checked for availability during the unreachability period, in seconds.
UnreachablePeriod	no	1-3600	45	After how many seconds of unreachability treat a host as unavailable.
VMwareCacheSize	no	256K-2G	8M	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). Note that shared memory is not allocated if there are no vmware collector instances configured to start. This parameter is supported since Zabbix 2.2.0.
VMwareFrequency	no	10-86400	60	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. This parameter is supported since Zabbix 2.2.0.

Parameter	Mandatory	Range	Default	Description
VMwarePerfFrequency	no	10-86400	60	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. This parameter is supported since Zabbix 2.2.9, 2.4.4
VMwareTimeout	no	1-300	10	The maximum number of seconds vmware collector will wait for a response from VMware service (vCenter or ESX hypervisor). This parameter is supported since Zabbix 2.2.9, 2.4.4

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

Parameter	Mandatory	Range	Default	Description
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 not 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'. Alias=zabbix.userid:vfs.file.regexp[/etc/pass 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.

Parameter	Mandatory	Range	Default	Description
AllowKey	no			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.
AllowRoot	no		0	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. 0 - do not allow 1 - allow
BufferSend	no	1-3600	5	Do not keep data longer than N seconds in buffer.
BufferSize	no	2-65535	100	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.
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)

Parameter	Mandatory	Range	Default	Description
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	no		0	Whether remote commands from Zabbix server are allowed. Since Zabbix 5.0.2, 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[*]
HostInterface	no	0-255 characters		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	no			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. The system.run[] item is supported regardless of AllowKey/DenyKey values. This option is only used when HostInterface is not defined. Supported since Zabbix 4.4.0.

Parameter	Mandatory	Range	Default	Description
HostMetadata	no	0-255 characters		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. This option is supported in version 2.2.0 and higher.
HostMetadataItem	no			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. This option is supported in version 2.2.0 and higher.
Hostname	no		Set by HostnameItem	Unique, case sensitive hostname. Required for active checks and must match hostname as configured on the server. Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128
HostnameItem	no		system.hostname	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. This option is supported in version 1.8.6 and higher.

Parameter	Mandatory	Range	Default	Description
Include	no			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/*.` Pattern matching is supported since Zabbix 2.4.0. See special notes about limitations.
ListenBacklog	no	0 - INT_MAX	SOMAXCONN	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'.
ListenIP	no		0.0.0.0	List of comma delimited IP addresses that the agent should listen on. Multiple IP addresses are supported in version 1.8.3 and higher.
ListenPort	no	1024-32767	10050	Agent will listen on this port for connections from the server.
LoadModule	no			Module to load at agent startup. Modules are used to extend functionality of the agent. Formats: LoadModule=<module.so> LoadModule=<path/module.so> LoadModule=</abs_path/module.so> 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.
LoadModulePath	no			Full path to location of agent modules. Default depends on compilation options.
LogFile	yes, if LogType is set to *file*, otherwise no			Name of log file.

Parameter	Mandatory	Range	Default	Description
LogFileSize	no	0-1024	1	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.
LogType	no		file	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.
LogRemoteCommands	no		0	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.
MaxLinesPerSecond	no	1-1000	20	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.
PidFile	no		/tmp/zabbix_agentd.pid	Name of PID file.
RefreshActiveChecks	no	60-3600	120	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.

Parameter	Mandatory	Range	Default	Description
Server	yes, if StartAgents is not explicitly set to 0			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. 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:d Spaces are allowed.
ServerActive	no			Zabbix server/proxy address to get active checks from. Server/proxy address is IP address or DNS name and optional port separated by colon. Multiple Zabbix servers and Zabbix proxies can be specified, separated by comma. More than one Zabbix proxy should not be specified from each Zabbix server. If Zabbix proxy is specified then Zabbix server 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: ServerActive=127.0.0.1:20051,zabbix.example.com,

Parameter	Mandatory	Range	Default	Description
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.)
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. The upper limit used to be 16 before version 1.8.5.
Timeout	no	1-30	3	Spend no more than Timeout seconds on processing.
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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 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.
TLSCertFile	no			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.
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 This parameter is supported since Zabbix 4.4.7.

Parameter	Mandatory	Range	Default	Description
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 GnuTLS: NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL::+SIGN-ALL:+CTYPE-X.509 Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128 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 GnuTLS: NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509 Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128 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.
TLSCipherPSK	no			GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for PSK-based encryption. Example for GnuTLS: NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL Example for OpenSSL: kECDHEPSK+AES128:kPSK+AES128 This parameter is supported since Zabbix 4.4.7.

Parameter	Mandatory	Range	Default	Description
TLSCipherPSK13	no			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 This parameter is supported since Zabbix 4.4.7.
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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.
TLSCRLFile	no			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.
TLSKeyFile	no			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.
TLSPSKFile	no			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.
TLSPSKIdentity	no			Pre-shared key identity string, used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.
TLSServerCertIssuer	no			Allowed server (proxy) certificate issuer. This parameter is supported since Zabbix 3.0.0.
TLSServerCertSubject	no			Allowed server (proxy) certificate subject. This parameter is supported since Zabbix 3.0.0.

Parameter	Mandatory	Range	Default	Description
UnsafeUserParameters	no	0,1	0	Allow all characters to be passed in arguments to user-defined parameters. 0 - do not allow 1 - allow The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @ Additionally, newline characters are not allowed. Supported since Zabbix 1.8.2.
User	no		zabbix	Drop privileges to a specific, existing user on the system. Only has effect if run as 'root' and AllowRoot is disabled. This parameter is supported since Zabbix 2.4.0.
UserParameter	no			User-defined parameter to monitor. There can be several user-defined parameters. Format: UserParameter=<key>,<shell command> Note that shell command must not return empty string or EOL only. Example: UserParameter=system.test,who|wc -l

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

Parameter	Mandatory	Range	Default	Description
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 not 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'. Alias=zabbix.userid:vfs.file.regexp[/etc/pa 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.

Parameter	Mandatory	Range	Default	Description
AllowKey	no			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.
BufferSend	no	1-3600	5	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.
BufferSize	no	2-65535	100	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*).
ControlSocket	no		/tmp/agent.sock	The control socket, used to send runtime commands with '-R' option.
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)

Parameter	Mandatory	Range	Default	Description
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.
EnablePersistentBuffer	no	0-1	0	Enable usage of local persistent storage for active items. 0 - disabled 1 - enabled If persistent storage is disabled, the memory buffer will be used.
HostInterface	no	0-255 characters		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	no			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. The system.run[] item is supported regardless of AllowKey/DenyKey values. This option is only used when HostInterface is not defined. Supported since Zabbix 4.4.0.

Parameter	Mandatory	Range	Default	Description
HostMetadata	no	0-255 characters		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.
HostMetadataItem	no			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.
Hostname	no		set by HostnameItem	Unique, case sensitive hostname. Required for active checks and must match hostname as configured on the server. Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128
HostnameItem	no		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 AllowKey/DenyKey values.

Parameter	Mandatory	Range	Default	Description
Include	no			You may include individual files or all files in a directory in the configuration file. During 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/*` See special notes about limitations.
ListenIP	no		0.0.0.0	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.
ListenPort	no	1024-32767	10050	Agent will listen on this port for connections from the server.
LogFile	yes, if LogType is set to *file*, otherwise no		/tmp/zabbix_agent2.log	Log file name if LogType is 'file'.
LogFileSize	no	0-1024	1	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.
LogType	no		file	Specifies where log messages are written to: *system* - syslog, *file* - file specified by LogFile parameter, *console* - standard output.
PersistentBufferFile	no			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*).

Parameter	Mandatory	Range	Default	Description
PersistentBufferPeriod	no	1m-365d	1h	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*).
PidFile	no		/tmp/zabbix_agent2.pid	Name of PID file.
Plugins	no			A plugin can have one or more plugin-specific configuration parameters in the format: Plugins.<PluginName>.<Parameter1>=< Plugins.<PluginName>.<Parameter2>=<
Plugins.<PluginName>.CustomQueriesPath	no			Full pathname of a directory containing .sql files with custom queries. Disabled by default. Supported for the following plugins: *Oracle, PostgreSQL*. **<PluginName>** - name of the plugin. Example: `Plugins.Oracle.CustomQueriesPath`
Plugins.<PluginName>.KeepAlive	no	60-900	300	The maximum time of waiting (in seconds) before unused plugin connections are closed. Supported for the following plugins: *Ceph, Memcached, MongoDB, MySQL, Oracle, Redis, PostgreSQL*. **<PluginName>** - name of the plugin. Example: `Plugins.Memcached.KeepAlive=200`
Plugins.<PluginName>.Default.Password	no			Default password for connecting to a plugin; used if no value is specified in an item key or named session. Supported for the following plugins: *Memcached, MongoDB, MySQL, Oracle, PostgreSQL, Redis*. **<PluginName>** - name of the plugin.
Plugins.<PluginName>.Timeout	no	1-30	global timeout	Request execution timeout (how long to wait for a request to complete before shutting it down). Supported for the following plugins: *Ceph, Memcached, MongoDB, MySQL, Redis, Docker, PostgreSQL, Smart, WebCertificate*. **<PluginName>** - name of the plugin.

Parameter	Mandatory	Range	Default	Description
Plugins.so.<PluginName>.Default.TLSCAFile				Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification for encrypted communications between Zabbix agent 2 and monitored databases; used if no value is specified in a named session. Supported for: *MySQL, PostgreSQL*. **<PluginName>** - name of the plugin.
Plugins.so.<PluginName>.Default.TLSCertFile				Full pathname of a file containing the agent certificate or certificate chain for encrypted communications between Zabbix agent 2 and monitored databases; used if no value is specified in a named session. Supported for: *MySQL, PostgreSQL*. **<PluginName>** - name of the plugin.
Plugins.so.<PluginName>.Default.TLSConnect				Encryption type for communications between Zabbix agent 2 and monitored databases; used if no value is specified in a named session. Supported for: *MySQL, PostgreSQL*. **<PluginName>** - name of the plugin. Accepted values: *required* - require TLS connection; *verify_ca* - verify certificates; *verify_full* - verify certificates and IP address.
Plugins.so.<PluginName>.Default.TLSKeyFile				Full pathname of a file containing the database private key for encrypted communications between Zabbix agent 2 and monitored databases; used if no value is specified in a named session. Supported for: *MySQL, PostgreSQL*. **<PluginName>** - name of the plugin.

Parameter	Mandatory	Range	Default	Description
	Plugins.so<PluginName>.Default.User			Default username for connecting to a plugin; used if no value is specified in an item key or named session. Supported for the following plugins: *Ceph, Memcached, MongoDB, MySQL, Oracle, PostgreSQL*. **<PluginName>** - name of the plugin.
	Plugins.so.Ceph.Default.ApiKey			Default API key for connecting to Ceph; used if no value is specified in an item key or named session.
	Plugins.so.Ceph.Default.Uri		https://localhost:8003	Default URI for connecting to Ceph; used if no value is specified in an item key or named session. Should not include embedded credentials (they will be ignored). Must match the URI format. Only `https` scheme is supported; a scheme can be omitted. A port can be omitted (default=8003). Examples: `https://127.0.0.1:8003` `localhost`
	Plugins.so.Ceph.InsecureSkipVerify	false / true	false	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).
	Plugins.so.Docker.Endpoint		unix:///var/run/docker.sock	Docker daemon unix-socket location. Must contain a scheme (only `unix://` is supported).

Parameter	Mandatory	Range	Default	Description
	Plugins.Log.MaxLinesPerSecond	1-1000	20	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.

	Plugins.Memcached.Default.Uri		tcp://localhost:11211	Default URI for connecting to Memcached; used if no value is specified in an item key or named session.
				
				Should not include embedded credentials (they will be ignored).
				Must match the URI format.
				Supported schemes: `tcp`, `unix`; a scheme can be omitted.
				A port can be omitted (default=11211).
				Examples:
				`tcp://localhost:11211`
				`localhost`
				`unix:/var/run/memcached.sock`

	Plugins.Mongo.Default.Uri			Default URI for connecting to MongoDB; used if no value is specified in an item key or named 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`

Parameter	Mandatory	Range	Default	Description

Plugins.Mysql.Default.Uri			tcp://localhost:3306	Default URI for connecting to MySQL; used if no value is specified in an item key or named session.
				
				Should not include embedded credentials (they will be ignored).
				Must match the URI format.
				Supported schemes: `tcp`, `unix`; a scheme can be omitted.
				A port can be omitted (default=3306).
				Examples:
				`tcp://localhost:3306`
				`localhost`
				`unix:/var/run/mysql.sock`

Plugins.Oracle.CallTimeout		1-30	global timeout	The maximum wait time in seconds for a request to be completed.
Plugins.Oracle.ConnectTimeout		1-30	global timeout	The maximum wait time in seconds for a connection to be established.
Plugins.Oracle.Default.Service				Default service name for connecting to Oracle (SID is not supported); used if no value is specified in an item key or named session.

Plugins.Oracle.Default.Uri			tcp://localhost:1521	Default URI for connecting to Oracle; used if no value is specified in an item key or named 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=1521).
				Examples:
				`tcp://127.0.0.1:1521`
				`localhost`

Plugins.Oracle.Service			XE	A service name to be used for connection (SID is not supported).
Plugins.Postgres.Host			localhost	IP address or DNS name of the host used for PostgreSQL. Examples: `localhost`, `192.168.1.1`
Plugins.Postgres.Port			5432	A port to be used for PostgreSQL.

Parameter	Mandatory	Range	Default	Description

	Plugins.Redis.Default.Uri		tcp://localhost:6379	Default URI for connecting to Redis; used if no value is specified in an item key or named session.
				
				Should not include embedded credentials (they will be ignored).
				Must match the URI format.
				Supported schemes: `tcp`, `unix`; a scheme can be omitted (since version 5.2.3).
				A port can be omitted (default=6379).
				Examples:
				`tcp://localhost:6379`
				`localhost`
				`unix:/var/run/redis.sock`

	Plugins.Smart.Path		smartctl	Path to the smartctl executable. This parameter is supported since Zabbix 5.0.9.
	Plugins.SystemRun.EnableRemoteCommands		0	Whether remote commands from Zabbix server are allowed. 0 - not allowed 1 - allowed This parameter is **not supported** since 5.0.2, use AllowKey/DenyKey parameters instead.
	Plugins.SystemRun.LogRemoteCommands		0	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.
Plugins' named sessions	no			If Zabbix agent is used to monitor several instances of the same kind, you can create a named session with own set of authorization parameters for each instance. Named session parameters format: Plugins.<PluginName>.<SessionName1>. Plugins.<PluginName>.<SessionName2>.

Parameter	Mandatory	Range	Default	Description
Plugins.<PluginName>.Sessions.<SessionName>.Password	no			Named session password. Supported for: *Memcached, MongoDB, MySQL, Oracle, PostgreSQL, Redis*. **<PluginName>** - name of the plugin. **<SessionName>** - name of a session for using in item keys.
Plugins.<PluginName>.Sessions.<SessionName>.TLSCAFile	no			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix agent 2 and monitored databases. Supported for: *MySQL, PostgreSQL*. **<PluginName>** - name of the plugin. **<SessionName>** - name of a session for using in item keys.
Plugins.<PluginName>.Sessions.<SessionName>.TLSCertFile	no			Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications between Zabbix agent 2 and monitored databases. Supported for: *MySQL, PostgreSQL*. **<PluginName>** - name of the plugin. **<SessionName>** - name of a session for using in item keys.
Plugins.<PluginName>.Sessions.<SessionName>.TLSConnect	no			Encryption type for communications between Zabbix agent 2 and monitored databases. Supported for: *MySQL, PostgreSQL*. **<PluginName>** - name of the plugin. **<SessionName>** - 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.

Parameter	Mandatory	Range	Default	Description
	Plugins.so			<PluginName>.Sessions.<SessionName>.TLSKeyFile Full pathname of a file containing the database private key used for encrypted communications between Zabbix agent 2 and monitored databases. Supported for: *MySQL, PostgreSQL*. **<PluginName>** - name of the plugin. **<SessionName>** - name of a session for using in item keys.
	Plugins.so			<PluginName>.Sessions.<SessionName>.User Named session username. Supported for: *Ceph, Memcached, MongoDB, MySQL, Oracle, PostgreSQL*. **<PluginName>** - name of the plugin. **<SessionName>** - name of a session for using in item keys.
	Plugins.so			Ceph.Sessions.<sessionName>.ApiKey Named session API key. Supported for: *Ceph*. **<PluginName>** - name of the plugin. **<SessionName>** - name of a session for using in item keys.

	Plugins.so			Ceph.Sessions.<SessionName>.Uri
				Connection string of a named session.
				<SessionName> - 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 5.0.7).
				A port can be omitted (default=8003).
				Examples:
				`https://127.0.0.1:8003`
				`localhost`

Parameter	Mandatory	Range	Default	Description

Plugins.Memcached.Sessions.<SessionName>.Uri	no			Connection string of a named session.
				<SessionName> - 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: `tcp`, `unix`; a scheme can be omitted (since version 5.0.7).
				A port can be omitted (default=11211).
				Examples:
				`tcp://localhost:11211`
				`localhost`
				`unix:/var/run/memcached.sock`

Plugins.Mongo.Sessions.<SessionName>.Uri	no			Connection string of a named session. **<SessionName>** - 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. A port can be omitted (default=27017). Examples: `tcp://127.0.0.1:27017`, `tcp:localhost`, `localhost`

Plugins.Mysql.Sessions.<SessionName>.Uri	no			Connection string of a named session.
				<SessionName> - 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: `tcp`, `unix`; a scheme can be omitted (since version 5.0.7).
				A port can be omitted (default=3306).
				Examples:
				`tcp://localhost:3306`
				`localhost`
				`unix:/var/run/mysql.sock`

Parameter	Mandatory	Range	Default	Description
	Plugins.so	Oracle.Sessions.<SessionName>.Service		Named session service name to be used for connection (SID is not supported). Supported for: *Oracle*. **<PluginName>** - name of the plugin. **<SessionName>** -name of a session for using in item keys.

	Plugins.so	Oracle.Sessions.<SessionName>.Uri		Named session connection string for Oracle.
				<SessionName> - 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 5.0.7).
				A port can be omitted (default=1521).
				Examples:
				`tcp://127.0.0.1:1521`
				`localhost`

	Plugins.so	Postgres.Sessions.<SessionName>.Database	postgres	Database name of a named session. **<SessionName>** - name of a session for using in item keys.

	Plugins.so	Redis.Sessions.<SessionName>.Uri		Connection string of a named session.
				<SessionName> - 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: `tcp`, `unix`; a scheme can be omitted (since version 5.0.7).
				A port can be omitted (default=6379).
				Examples:
				`tcp://localhost:6379`
				`localhost`
				`unix:/var/run/redis.sock`

RefreshActiveChecks	no	60-3600	120	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.

Parameter	Mandatory	Range	Default	Description
Server	yes			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 Spaces are allowed.
ServerActive	no			Zabbix server/proxy address to get active checks from. Server/proxy address is IP address or DNS name and optional port separated by colon. Multiple Zabbix servers and Zabbix proxies can be specified, separated by comma. More than one Zabbix proxy should not be specified from each Zabbix server. If Zabbix proxy is specified then Zabbix server 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: ServerActive=127.0.0.1:20051,zabbix.example.com
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.)

Parameter	Mandatory	Range	Default	Description
StatusPort	no	1024-32767		If set, agent will listen on this port for HTTP status requests (http://localhost:<port>/status).
Timeout	no	1-30	3	Spend no more than Timeout seconds on processing.
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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
TLSCAFile	no			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.
TLSCertFile	no			Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components.
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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
TLSCRLFile	no			Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications with Zabbix components.
TLSKeyFile	no			Full pathname of a file containing the agent private key used for encrypted communications with Zabbix components.
TLSPSKFile	no			Full pathname of a file containing the agent pre-shared key used for encrypted communications with Zabbix components.

Parameter	Mandatory	Range	Default	Description
TLSPSKIdentity	no			Pre-shared key identity string, used for encrypted communications with Zabbix server.
TLSServerCertIssuer	no			Allowed server (proxy) certificate issuer.
TLSServerCertSubject	no			Allowed server (proxy) certificate subject.
UnsafeUserParameters	no	0,1	0	Allow all characters to be passed in arguments to user-defined parameters. The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) > | # @ Additionally, newline characters are not allowed.
UserParameter	no			User-defined parameter to monitor. There can be several user-defined parameters. Format: UserParameter=<key>,<shell command> Note that the shell command must not return empty string or EOL only. Example: UserParameter=system.test,who|wc -l

5 Zabbix agent (Windows)

Overview

This section lists parameters supported in a Zabbix agent (Windows) 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

Parameter	Mandatory	Range	Default	Description
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 not allowed. Different *Alias* keys may reference the same item key. Aliases can be used in *HostMetadataItem* but not in *HostnameItem* or *PerfCounter* parameters. Examples: 1. Retrieving paging file usage in percents from the server. Alias=pg_usage:perf_counter[\Paging File(_Total)\% Usage] Now shorthand key **pg_usage** may be used to retrieve data. 2. Getting CPU load with default and custom parameters. Alias=cpu.load:system.cpu.load Alias=cpu.load[*]:system.cpu.load[*] This allows use **cpu.load** key to get CPU utilization percentage with default parameters as well as use **cpu.load[percpu,avg15]** to get specific data about CPU load. 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.

Parameter	Mandatory	Range	Default	Description
AllowKey	no			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.
BufferSend	no	1-3600	5	Do not keep data longer than N seconds in buffer.
BufferSize	no	2-65535	100	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.
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.

Parameter	Mandatory	Range	Default	Description
EnableRemoteCommands	no		0	Whether remote commands from Zabbix server are allowed. Since Zabbix 5.0.2, 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[*]
HostInterface	no	0-255 characters		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	no			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. The system.run[] item is supported regardless of AllowKey/DenyKey values. This option is only used when HostInterface is not defined. Supported since Zabbix 4.4.0.
HostMetadata	no	0-255 characters		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. This option is supported in version 2.2.0 and higher.

Parameter	Mandatory	Range	Default	Description
HostMetadataItem	no			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. This option is supported in version 2.2.0 and higher.
Hostname	no		Set by HostnameItem	Unique, case sensitive hostname. Required for active checks and must match hostname as configured on the server. Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128
HostnameItem	no		system.hostname	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. This option is supported in version 1.8.6 and higher. See also a more detailed description.

Parameter	Mandatory	Range	Default	Description
Include	no			You may include individual files or all files in a directory in the configuration file (located in `C:\Program Files\Zabbix Agent` by default if Zabbix agent is installed using Windows MSI installer packages; located in the folder specified during installation if Zabbix agent is installed as a zip archive). All included files must have correct syntax, otherwise agent will not start. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. For example: `C:\Program Files\Zabbix Agent\zabbix_agentd.d\*.conf`. Pattern matching is supported since Zabbix 2.4.0. See special notes about limitations.
ListenBacklog	no	0 - INT_MAX	SOMAXCONN	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'.
ListenIP	no		0.0.0.0	List of comma-delimited IP addresses that the agent should listen on. Multiple IP addresses are supported since Zabbix 1.8.3.
ListenPort	no	1024-32767	10050	Agent will listen on this port for connections from the server.
LogFile	yes, if LogType is set to *file*, otherwise no		C:\zabbix_agentd.log	Name of the agent log file.
LogFileSize	no	0-1024	1	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.

Parameter	Mandatory	Range	Default	Description
LogType	no		file	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.
LogRemoteCommands	no		0	Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled
MaxLinesPerSecond	no	1-1000	20	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.
PerfCounter	no			Defines a new parameter <parameter_name> which is an average value for system performance counter <perf_counter_path> for the specified time period <period> (in seconds). Syntax: <parameter_name>,"<perf_counter_path>",<period> 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: PerfCounter = 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 may run "typeperf -qx" to get list of all performance counters available in Windows.

Parameter	Mandatory	Range	Default	Description
PerfCounterEn	no			Defines a new parameter <parameter_name> which is an average value for system performance counter <perf_counter_path> for the specified time period <period> (in seconds). Syntax: <parameter_name>,"<perf_counter_path>",<period> 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.
RefreshActiveChecks	no	60-3600	120	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.

Parameter	Mandatory	Range	Default	Description
Server	yes, if StartAgents is not explicitly set to 0			List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers. 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,::1,2001:d Spaces are allowed.
ServerActive	no	(*)		Zabbix server/proxy address to get active checks from. Server/proxy address is IP address or DNS name and optional port separated by colon. Multiple Zabbix servers and Zabbix proxies can be specified, separated by comma. More than one Zabbix proxy should not be specified from each Zabbix server. If Zabbix proxy is specified then Zabbix server 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: ServerActive=127.0.0.1:20051,zabbix.example.com,

Parameter	Mandatory	Range	Default	Description
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.)
StartAgents	no	0-63 (*)	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. The upper limit used to be 16 before version 1.8.5.
Timeout	no	1-30	3	Spend no more than Timeout seconds on processing
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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 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.
TLSCertFile	no			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.

Parameter	Mandatory	Range	Default	Description
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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.
TLSCRLFile	no			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.
TLSKeyFile	no			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.
TLSPSKFile	no			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.
TLSPSKIdentity	no			Pre-shared key identity string, used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.
TLSServerCertIssuer	no			Allowed server (proxy) certificate issuer. This parameter is supported since Zabbix 3.0.0.
TLSServerCertSubject	no			Allowed server (proxy) certificate subject. This parameter is supported since Zabbix 3.0.0.
UnsafeUserParameters	no	0-1	0	Allow all characters to be passed in arguments to user-defined parameters. 0 - do not allow 1 - allow The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @ Additionally, newline characters are not allowed.

Parameter	Mandatory	Range	Default	Description
UserParameter				User-defined parameter to monitor. There can be several user-defined parameters. Format: UserParameter=<key>,<shell command> Note that shell command must not return empty string or EOL only. Example: UserParameter=system.test,echo 1

Note:
(*) 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.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

Parameter	Mandatory	Range	Default	Description
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 not 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'. Alias=zabbix.userid:vfs.file.regexp[/etc/pa 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.

Parameter	Mandatory	Range	Default	Description
AllowKey	no			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.
BufferSend	no	1-3600	5	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.
BufferSize	no	2-65535	100	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*).
ControlSocket	no		\\.\pipe\agent.sock	The control socket, used to send runtime commands with '-R' option.
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)

Parameter	Mandatory	Range	Default	Description
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.
EnablePersistentBuffer	no	0-1	0	Enable usage of local persistent storage for active items. 0 - disabled 1 - enabled If persistent storage is disabled, the memory buffer will be used.
HostInterface	no	0-255 characters		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	no			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. The system.run[] item is supported regardless of AllowKey/DenyKey values. This option is only used when HostInterface is not defined. Supported since Zabbix 4.4.0.

Parameter	Mandatory	Range	Default	Description
HostMetadata	no	0-255 characters		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.
HostMetadataItem	no			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 *EnableRemoteCommands* value. The value returned by the item must be a UTF-8 string otherwise it will be ignored.
Hostname	no		set by HostnameItem	Unique, case sensitive hostname. Required for active checks and must match hostname as configured on the server. Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128
HostnameItem	no		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.

Parameter	Mandatory	Range	Default	Description
Include	no			You may include individual files or all files in a directory in the configuration file (located in `C:\Program Files\Zabbix Agent 2` by default if Zabbix agent is installed using Windows MSI installer packages; located in the folder specified during installation if Zabbix agent is installed as a zip archive). All included files must have correct syntax, otherwise agent will not start. The path can be relative to the *zabbix_agent2.conf* file location (e.g., `Include=.\zabbix_agent2.d\plugin` To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. For example: `C:\Program Files\Zabbix Agent2\zabbix_agent2.d\*.conf`. See special notes about limitations.
ListenIP	no		0.0.0.0	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.
ListenPort	no	1024-32767	10050	Agent will listen on this port for connections from the server.
LogFile	yes, if LogType is set to *file*, otherwise no		c:\zabbix_agent2.log	Log file name if LogType is 'file'.
LogFileSize	no	0-1024	1	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.
LogType	no		file	Specifies where log messages are written to: *file* - file specified by LogFile parameter, *console* - standard output.

Parameter	Mandatory	Range	Default	Description
PersistentBufferFile	no			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*).
PersistentBufferPeriod	no	1m-365d	1h	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*).
Plugins	no			A plugin can have one or more plugin-specific configuration parameters in the format: Plugins.<PluginName>.<Parameter1>=< Plugins.<PluginName>.<Parameter2>=<
Plugins.<PluginName>.CustomQueriesPath	no			Full pathname of a directory containing .sql files with custom queries. Disabled by default. Supported for the following plugins: *Oracle, PostgreSQL*. **<PluginName>** - name of the plugin. Example: `Plugins.Oracle.CustomQueriesPath`
Plugins.<PluginName>.KeepAlive	no	60-900	300	The maximum time of waiting (in seconds) before unused plugin connections are closed. Supported for the following plugins: *Ceph, Memcached, MongoDB, MySQL, Oracle, Redis, PostgreSQL*. **<PluginName>** - name of the plugin. Example: `Plugins.Memcached.KeepAlive=200`
Plugins.<PluginName>.Timeout	no	1-30	global timeout	Request execution timeout (how long to wait for a request to complete before shutting it down). Supported for the following plugins: *Ceph, Memcached, MongoDB, MySQL, Redis, Docker, PostgreSQL, Smart*. **<PluginName>** - name of the plugin.

Parameter	Mandatory	Range	Default	Description
	Plugins.Ceph.InsecureSkipVerify	false / true	false	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).
	Plugins.Docker.Endpoint		unix:///var/run/docker.sock	Docker daemon unix-socket location. Must contain a scheme (only `unix://` is supported).
	Plugins.Log.MaxLinesPerSecond	1-1000	20	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.
	Plugins.Oracle.CallTimeout	1-30	global timeout	The maximum wait time in seconds for a request to be completed.
	Plugins.Oracle.ConnectTimeout	1-30	global timeout	The maximum wait time in seconds for a connection to be established.
	Plugins.Oracle.Service		XE	A service name to be used for connection (SID is not supported).
	Plugins.Postgres.Host		localhost	IP address or DNS name of the host used for PostgreSQL. Examples: `localhost`, `192.168.1.1`
	Plugins.Postgres.Port		5432	A port to be used for PostgreSQL.
	Plugins.Smart.Path		smartctl	Path to the smartctl executable. This parameter is supported since Zabbix 5.0.9.

Parameter	Mandatory	Range	Default	Description
	Plugins.SystemRun.EnableRemoteCommands		0	Whether remote commands from Zabbix server are allowed. 0 - not allowed 1 - allowed This parameter is **not supported** since 5.0.2, use AllowKey/DenyKey parameters instead.
	Plugins.SystemRun.LogRemoteCommands		0	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.
	Plugins.WindowsEventlog.MaxLinesPerSecond	1-1000	20	Maximum number of new lines the agent will send per second to Zabbix Server or Proxy processing 'eventlog' checks. The provided value will be overridden by the parameter 'maxlines', provided in 'eventlog' item keys.
Plugins' named sessions	no			If Zabbix agent is used to monitor several instances of the same kind, you can create a named session with own set of authorization parameters for each instance. Named session parameters format: Plugins.<PluginName>.<SessionName1>. Plugins.<PluginName>.<SessionName2>.
	Plugins.<PluginName>.Sessions.<SessionName>.Password			Named session password. Supported for: *Memcached, MongoDB, MySQL, Oracle, PostgreSQL, Redis*. **<PluginName>** - name of the plugin. **<SessionName>** - name of the session for using in item keys.

Parameter	Mandatory	Range	Default	Description
Plugins.<PluginName>.Sessions.<SessionName>.TLSCAFile	no			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix agent 2 and monitored databases. Supported for: *MySQL, PostgreSQL*. **<PluginName>** - name of the plugin. **<SessionName>** - name of a session for using in item keys.
Plugins.<PluginName>.Sessions.<SessionName>.TLSCertFile	no			Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications between Zabbix agent 2 and monitored databases. Supported for: *MySQL, PostgreSQL*. **<PluginName>** - name of the plugin. **<SessionName>** - name of a session for using in item keys.
Plugins.<PluginName>.Sessions.<SessionName>.TLSConnect	no			Encryption type for communications between Zabbix agent 2 and monitored databases. Supported for: *MySQL, PostgreSQL*. **<PluginName>** - name of the plugin. **<SessionName>** - 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.
Plugins.<PluginName>.Sessions.<SessionName>.TLSKeyFile	no			Full pathname of a file containing the database private key used for encrypted communications between Zabbix agent 2 and monitored databases. Supported for: *MySQL, PostgreSQL*. **<PluginName>** - name of the plugin. **<SessionName>** - name of a session for using in item keys.

Parameter	Mandatory	Range	Default	Description
	Plugins.so	<PluginName>.Sessions.<SessionName>.User		Named session username. Supported for: *Ceph, Memcached, MongoDB, MySQL, Oracle, PostgreSQL*. **<PluginName>** - name of the plugin. **<SessionName>** - name of the session for using in item keys.
	Plugins.so	Ceph.Sessions.<sessionName>.ApiKey		Named session API key. Supported for: *Ceph*. **<PluginName>** - name of the plugin. **<SessionName>** - name of the session for using in item keys.

	Plugins.so	Ceph.Sessions.<SessionName>.Uri	https://localhost:8003	Connection string of a named session.
				<SessionName> - name of the 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 5.0.7).
				A port can be omitted (default=8003).
				Examples:
				`https://127.0.0.1:8003`
				`localhost`

	Plugins.so	Memcached.Sessions.<SessionName>.Uri	tcp://localhost:11211	Connection string of a named session.
				<SessionName> - 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: `tcp`, `unix`; a scheme can be omitted (since version 5.0.7).
				A port can be omitted (default=11211).
				Examples:
				`tcp://localhost:11211`
				`localhost`
				`unix:/var/run/memcached.sock`

Parameter	Mandatory	Range	Default	Description
Plugins.Mongo.Sessions.<SessionName>.Uri	no			Connection string of a named session. **<SessionName>** - 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. A port can be omitted (default=27017). Examples: `tcp://127.0.0.1:27017`, `tcp:localhost`, `localhost`
Plugins.Mysql.Sessions.<SessionName>.Uri	no		tcp://localhost:3306	Connection string of a named session. **<SessionName>** - 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: `tcp`, `unix`; a scheme can be omitted (since version 5.0.7). A port can be omitted (default=3306). Examples: `tcp://localhost:3306` `localhost` `unix:/var/run/mysql.sock`
Plugins.Oracle.Sessions.<SessionName>.Service	no			Named session service name to be used for connection (SID is not supported). Supported for: *Oracle*. **<PluginName>** - name of the plugin. **<SessionName>** -name of a session for using in item keys.

Parameter	Mandatory	Range	Default	Description

	Plugins.Oracle.Sessions.<SessionName>.Uri		tcp://localhost:1521	Named session connection string for Oracle.
				<SessionName> - 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 5.0.7).
				A port can be omitted (default=1521).
				Examples:
				`tcp://127.0.0.1:1521`
				`localhost`

	Plugins.Postgres.Sessions.<SessionName>.Database		postgres	Database name of a named session. **<SessionName>** - name of a session for using in item keys.

	Plugins.Redis.Sessions.<SessionName>.Uri		tcp://localhost:6379	Connection string of a named session.
				<SessionName> - 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: `tcp`, `unix`; a scheme can be omitted (since version 5.0.7).
				A port can be omitted (default=6379).
				Examples:
				`tcp://localhost:6379`
				`localhost`
				`unix:/var/run/redis.sock`

RefreshActiveChecks	no	60-3600	120	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.

Parameter	Mandatory	Range	Default	Description
Server	yes			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 Spaces are allowed.
ServerActive	no			Zabbix server/proxy address to get active checks from. Server/proxy address is IP address or DNS name and optional port separated by colon. Multiple Zabbix servers and Zabbix proxies can be specified, separated by comma. More than one Zabbix proxy should not be specified from each Zabbix server. If Zabbix proxy is specified then Zabbix server 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: ServerAc-tive=127.0.0.1:20051,zabbix.example.com
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.)

Parameter	Mandatory	Range	Default	Description
StatusPort	no	1024-32767		If set, agent will listen on this port for HTTP status requests (http://localhost:<port>/status).
Timeout	no	1-30	3	Spend no more than Timeout seconds on processing.
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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
TLSCAFile	no			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.
TLSCertFile	no			Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components.
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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
TLSCRLFile	no			Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications with Zabbix components.
TLSKeyFile	no			Full pathname of a file containing the agent private key used for encrypted communications with Zabbix components.
TLSPSKFile	no			Full pathname of a file containing the agent pre-shared key used for encrypted communications with Zabbix components.

Parameter	Mandatory	Range	Default	Description
TLSPSKIdentity	no			Pre-shared key identity string, used for encrypted communications with Zabbix server.
TLSServerCertIssuer	no			Allowed server (proxy) certificate issuer.
TLSServerCertSubject	no			Allowed server (proxy) certificate subject.
UnsafeUserParameters	no	0,1	0	Allow all characters to be passed in arguments to user-defined parameters. The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @ Additionally, newline characters are not allowed.
UserParameter	no			User-defined parameter to monitor. There can be several user-defined parameters. Format: UserParameter=<key>,<shell command> Note that the shell command must not return empty string or EOL only. Example: UserParameter=system.test,who|wc -l

7 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.

Variable	Property	Mandatory	Range	Default	Description
LISTEN_IP	zabbix.listenIP	no		0.0.0.0	IP address to listen on.
LISTEN_PORT	zabbix.listenPort	no	1024-32767	10052	Port to listen on.
PID_FILE	zabbix.pidFile	no		/tmp/zabbix_java.pid	Name of PID file. If omitted, Zabbix Java Gateway is started as a console application.
PROPERTIES_FILE	zabbix.propertiesFile	no			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: "javax.net.ssl.trustStorePassword=<p Supported since Zabbix 5.0.7.

Variable	Property	Mandatory	Range	Default	Description
START_POLLERS	zabbix.startPollers	no	1-1000	5	Number of worker threads to start.
TIMEOUT	zabbix.timeout	no	1-30	3	How long to wait for network operations.

Warning:
Port 10052 is not IANA registered.

8 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). Therefore do not define one parameter in several "Include" files (e.g. to override a general setting with a specific one).
- 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/my_specific.conf" file produces a backup copy "include/my_specific_conf.BAK" then both files will be included. Move "include/my_specific.conf.BAK" out of the "Include" directory. On Linux, contents of the "Include" directory can be checked with a "ls -al" command for unnecessary files.

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). Therefore do not define one parameter in several "Include" files (e.g. to override a general setting with a specific one).

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.

name		value type	description
server→proxy:			
request		*string*	'proxy config'
<table>		*object*	one or more objects with <table> data
	fields	*array*	array of field names
	-	*string*	field name
	data	*array*	array of rows
	-	*array*	array of columns

name	value type	description
-	*string*,*number*	column value with type depending on column type in database schema
proxy→server:		
response	*string*	the request success information ('success' or 'failed')
version	*string*	the proxy version (<major>.<minor>.<build>)

Example:

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",
            "tls_issuer",
            "tls_subject",
            "tls_psk_identity",
            "tls_psk"
        ],
        "data":[
            [
                10001,
                "Template OS Linux",
                3,
                -1,
                2,
                "",
                "",
                "Template OS Linux",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
```

```
            [
                10050,
                "Template App Zabbix Agent",
                3,
                -1,
                2,
                "",
                "",
                "Template App Zabbix Agent",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
            [
                10105,
                "Logger",
                0,
                -1,
                2,
                "",
                "",
                "Logger",
                1,
                1,
                "",
                "",
                "",
                ""
            ]
        ]
    },
    "interface":{
        "fields":[
            "interfaceid",
            "hostid",
            "main",
            "type",
            "useip",
            "ip",
            "dns",
            "port",
            "bulk"
        ],
        "data":[
            [
                2,
                10105,
                1,
                1,
                1,
                "127.0.0.1",
                "",
                "10050",
                1
            ]
        ]
    },
    ...
}
```

proxy→server:

```
{
  "response": "success",
  "version": "5.0.0"
}
```

Proxy request

The `proxy data` request is used to obtain host 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
host availability	*array*	*(optional)* array of host availability data objects
hostid	*number*	host identifier
available	*number*	Zabbix agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
error	*string*	Zabbix agent error message or empty string
snmp_available	*number*	SNMP agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
snmp_error	*string*	SNMP agent error message or empty string
ipmi_available	*number*	IPMI agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
ipmi_error	*string*	IPMI agent error message or empty string
jmx_available	*number*	JMX agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
jmx_error	*string*	JMX agent error message or empty string
history data	*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**, *ITEM_STATE_NORMAL* **1**, *ITEM_STATE_NOTSUPPORTED*
lastlogsize	*number*	*(optional)* last log size of log type items
mtime	*number*	*(optional)* modify time of log type items
discovery data	*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

name		value type	description
	type	*number*	the discovery check type: **-1** discovery rule data **0**, *SVC_SSH* - SSH service check **1**, *SVC_LDAP* - LDAP service check **2**, *SVC_SMTP* - SMTP service check **3**, *SVC_FTP* - FTP service check **4**, *SVC_HTTP* - HTTP service check **5**, *SVC_POP* - POP service check **6**, *SVC_NNTP* - NNTP service check **7**, *SVC_IMAP* - IMAP service check **8**, *SVC_TCP* - TCP port availability check **9**, *SVC_AGENT* - Zabbix agent **10**, *SVC_SNMPv1* - SNMPv1 agent **11**, *SVC_SNMPv2* - SNMPv2 agent **12**, *SVC_ICMPPING* - ICMP ping **13**, *SVC_SNMPv3* - SNMPv3 agent **14**, *SVC_HTTPS* - HTTPS service check **15**, *SVC_TELNET* - Telnet availability check
	ip	*string*	the host IP address
	dns	*string*	the host DNS name
	port	*number*	*(optional)* service port number
	key_	*string*	*(optional)* the item key for discovery check of type **9** *SVC_AGENT*
	value	*string*	*(optional)* value received from the service, can be empty for most of services
	status	*number*	*(optional)* service status: **0**, *DOBJECT_STATUS_UP* - Service UP **1**, *DOBJECT_STATUS_DOWN* - Service DOWN
auto registration		*array*	*(optional)* array of autoregistration data objects
	clock	*number*	the autoregistration data timestamp
	host	*string*	the host name
	ip	*string*	*(optional)* the host IP address
	dns	*string*	*(optional)* the resolved DNS name from IP address
	port	*string*	*(optional)* the host port
	host_metadata	*string*	*(optional)* the host metadata sent by agent (based on HostMetadata or HostMetadataItem agent configuration parameter)
tasks		*array*	*(optional)* array of tasks
	type	*number*	the task type: **0**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT* - remote command result
	status	*number*	the remote command execution status: **0**, *ZBX_TM_REMOTE_COMMAND_COMPLETED* - the remote command completed successfully **1**, *ZBX_TM_REMOTE_COMMAND_FAILED* - the remote command failed
	error	*string*	*(optional)* the error message
	parent_taskid	*number*	the parent task id
more		*number*	*(optional)* 1 - there are more history data to send
clock		*number*	*(optional)* data transfer timestamp (seconds)
ns		*number*	*(optional)* data transfer timestamp (nanoseconds)
version		*string*	the proxy version (<major>.<minor>.<build>)

name	value type	description
server→proxy:		
response	*string*	the request success information ('success' or 'failed')
tasks	*array*	*(optional)* array of tasks
type	*number*	the task type: **1**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND* - remote command
clock	*number*	the task creation time
ttl	*number*	the time in seconds after which task expires
command_type	*number*	the remote command type: **0**, *ZBX_SCRIPT_TYPE_CUSTOM_SCRIPT* - use custom script **1**, *ZBX_SCRIPT_TYPE_IPMI* - use IPMI **2**, *ZBX_SCRIPT_TYPE_SSH* - use SSH **3**, *ZBX_SCRIPT_TYPE_TELNET* - use Telnet **4**, *ZBX_SCRIPT_TYPE_GLOBAL_SCRIPT* - use global script (currently functionally equivalent to custom script)
command	*string*	the remote command to execute
execute_on	*number*	the execution target for custom scripts: **0**, *ZBX_SCRIPT_EXECUTE_ON_AGENT* - execute script on agent **1**, *ZBX_SCRIPT_EXECUTE_ON_SERVER* - execute script on server **2**, *ZBX_SCRIPT_EXECUTE_ON_PROXY* - execute script on proxy
port	*number*	*(optional)* the port for telnet and ssh commands
authtype	*number*	*(optional)* the authentication type for ssh commands
username	*string*	*(optional)* the user name for telnet and ssh commands
password	*string*	*(optional)* the password for telnet and ssh commands
publickey	*string*	*(optional)* the public key for ssh commands
privatekey	*string*	*(optional)* the private key for ssh commands
parent_taskid	*number*	the parent task id
hostid	*number*	target hostid

Example:

server→proxy:

```
{
  "request": "proxy data"
}
```

proxy→server:

```
{
    "session": "12345678901234567890123456789012"
    "host availability":[
        {
            "hostid":10106,
            "available":1,
            "error":"",
            "snmp_available":0,
            "snmp_error":"",
            "ipmi_available":0,
```

```
            "ipmi_error":"",
            "jmx_available":0,
            "jmx_error":""
        },
        {
            "hostid":10107,
            "available":1,
            "error":"",
            "snmp_available":0,
            "snmp_error":"",
            "ipmi_available":0,
            "ipmi_error":"",
            "jmx_available":0,
            "jmx_error":""
        }
    ],
    "history data":[
        {
            "itemid":"12345",
            "clock":1478609647,
            "ns":332510044,
            "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":"5.0.0"
}
```


server→proxy:


```
{
  "response": "success",
  "tasks":[
      {
         "type": 1,
         "clock": 1478608371,
         "ttl": 600,
         "commandtype": 2,
         "command": "restart_service1.sh",
         "execute_on": 2,
         "port": 80,
         "authtype": 0,
         "username": "userA",
         "password": "password1",
         "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe",
         "privatekey": "lsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u09lHNsj6tQ5QCqGKukO1De7zhd",
         "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.

name	value type	description
proxy→server:		
request	*string*	'proxy heartbeat'
host	*string*	the proxy name
version	*string*	the proxy version (<major>.<minor>.<build>)
server→proxy:		
response	*string*	the request success information ('success' or 'failed')

proxy→server:

```
{
   "request": "proxy heartbeat",
   "host": "Proxy #12",
   "version": "5.0.0"
}
```

server→proxy:

```
{
  "response": "success"
}
```

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.

name			value type	description
proxy→server:				
request			*string*	'proxy config'
host			*string*	proxy name
version			*string*	the proxy version (<major>.<minor>.<build>)
server→proxy:				
request			*string*	'proxy config'
<table>			*object*	one or more objects with <table> data
	fields		*array*	array of field names
		-	*string*	field name
	data		*array*	array of rows
		-	*array*	array of columns
			- *string,number*	column value with type depending on column type in database schema
proxy→server:				
response			*string*	the request success information ('success' or 'failed')

Example:

proxy→server:

```
{
  "request": "proxy config",
  "host": "Proxy #12",
  "version":"5.0.0"
}
```

server→proxy:

```
{
    "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",
            "tls_issuer",
            "tls_subject",
            "tls_psk_identity",
            "tls_psk"
        ],
        "data":[
            [
                10001,
                "Template OS Linux",
                3,
                -1,
                2,
                "",
                "",
                "Template OS Linux",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
            [
                10050,
                "Template App Zabbix Agent",
                3,
                -1,
                2,
                "",
                "",
                "Template App Zabbix Agent",
                1,
                1,
                "",
                "",
```

```
                "",
                ""
            ],
            [
                10105,
                "Logger",
                0,
                -1,
                2,
                "",
                "",
                "Logger",
                1,
                1,
                "",
                "",
                "",
                ""
            ]
        ]
    },
    "interface":{
        "fields":[
            "interfaceid",
            "hostid",
            "main",
            "type",
            "useip",
            "ip",
            "dns",
            "port",
            "bulk"
        ],
        "data":[
            [
                2,
                10105,
                1,
                1,
                1,
                "127.0.0.1",
                "",
                "10050",
                1
            ]
        ]
    },
    ...
}
```

proxy→server:

```
{
  "response": "success"
}
```

Proxy data request

The `proxy data` request is sent by proxy to provide host availability, history, discovery and autoregistration data. This request is sent every `DataSenderFrequency` (proxy configuration parameter) seconds. Note that active proxy will still poll Zabbix server every second for remote command tasks (with an empty `proxy data` request).

name	value type	description
proxy→server:		
request	*string*	'proxy data'
host	*string*	the proxy name
session	*string*	data session token
host availability	*array*	*(optional)* array of host availability data objects
	hostid *number*	host identifier
	available *number*	Zabbix agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
	error *string*	Zabbix agent error message or empty string
	snmp_available *number*	SNMP agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
	snmp_error *string*	SNMP agent error message or empty string
	ipmi_available *number*	IPMI agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
	ipmi_error *string*	IPMI agent error message or empty string
	jmx_available *number*	JMX agent availability **0**, *HOST_AVAILABLE_UNKNOWN* - unknown **1**, *HOST_AVAILABLE_TRUE* - available **2**, *HOST_AVAILABLE_FALSE* - unavailable
	jmx_error *string*	JMX agent error message or empty string
history data	*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**, *ITEM_STATE_NORMAL* **1**, *ITEM_STATE_NOTSUPPORTED*
	lastlogsize *number*	*(optional)* last log size of log type items
	mtime *number*	*(optional)* modify time of log type items
discovery data	*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

name		value type	description
	type	*number*	the discovery check type: **-1** discovery rule data **0**, *SVC_SSH* - SSH service check **1**, *SVC_LDAP* - LDAP service check **2**, *SVC_SMTP* - SMTP service check **3**, *SVC_FTP* - FTP service check **4**, *SVC_HTTP* - HTTP service check **5**, *SVC_POP* - POP service check **6**, *SVC_NNTP* - NNTP service check **7**, *SVC_IMAP* - IMAP service check **8**, *SVC_TCP* - TCP port availability check **9**, *SVC_AGENT* - Zabbix agent **10**, *SVC_SNMPv1* - SNMPv1 agent **11**, *SVC_SNMPv2* - SNMPv2 agent **12**, *SVC_ICMPPING* - ICMP ping **13**, *SVC_SNMPv3* - SNMPv3 agent **14**, *SVC_HTTPS* - HTTPS service check **15**, *SVC_TELNET* - Telnet availability check
	ip	*string*	the host IP address
	dns	*string*	the host DNS name
	port	*number*	*(optional)* service port number
	key_	*string*	*(optional)* the item key for discovery check of type **9** *SVC_AGENT*
	value	*string*	*(optional)* value received from the service, can be empty for most of services
	status	*number*	*(optional)* service status: **0**, *DOBJECT_STATUS_UP* - Service UP **1**, *DOBJECT_STATUS_DOWN* - Service DOWN
auto registration		*array*	*(optional)* array of autoregistration data objects
	clock	*number*	the autoregistration data timestamp
	host	*string*	the host name
	ip	*string*	*(optional)* the host IP address
	dns	*string*	*(optional)* the resolved DNS name from IP address
	port	*string*	*(optional)* the host port
	host_metadata	*string*	*(optional)* the host metadata sent by agent (based on HostMetadata or HostMetadataItem agent configuration parameter)
tasks		*array*	*(optional)* array of tasks
	type	*number*	the task type: **0**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT* - remote command result
	status	*number*	the remote command execution status: **0**, *ZBX_TM_REMOTE_COMMAND_COMPLETED* - the remote command completed successfully **1**, *ZBX_TM_REMOTE_COMMAND_FAILED* - the remote command failed
	error	*string*	*(optional)* the error message
	parent_taskid	*number*	the parent task id
more		*number*	*(optional)* 1 - there are more history data to send
clock		*number*	*(optional)* data transfer timestamp (seconds)
ns		*number*	*(optional)* data transfer timestamp (nanoseconds)
version		*string*	the proxy version (<major>.<minor>.<build>)

name	value type	description
server→proxy:		
response	*string*	the request success information (’success’ or ’failed’)
upload	*string*	upload control for historical data (history, autoregistration, host availability, network discovery). Possible values: **enabled** - normal operation **disabled** - server is not accepting data (possibly due to internal cache over limit)
tasks	*array*	*(optional)* array of tasks
	type *number*	the task type: **1**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND* - remote command
	clock *number*	the task creation time
	ttl *number*	the time in seconds after which task expires
	command_type *number*	the remote command type: **0**, *ZBX_SCRIPT_TYPE_CUSTOM_SCRIPT* - use custom script **1**, *ZBX_SCRIPT_TYPE_IPMI* - use IPMI **2**, *ZBX_SCRIPT_TYPE_SSH* - use SSH **3**, *ZBX_SCRIPT_TYPE_TELNET* - use Telnet **4**, *ZBX_SCRIPT_TYPE_GLOBAL_SCRIPT* - use global script (currently functionally equivalent to custom script)
	command *string*	the remote command to execute
	execute_on *number*	the execution target for custom scripts: **0**, *ZBX_SCRIPT_EXECUTE_ON_AGENT* - execute script on agent **1**, *ZBX_SCRIPT_EXECUTE_ON_SERVER* - execute script on server **2**, *ZBX_SCRIPT_EXECUTE_ON_PROXY* - execute script on proxy
	port *number*	*(optional)* the port for telnet and ssh commands
	authtype *number*	*(optional)* the authentication type for ssh commands
	username *string*	*(optional)* the user name for telnet and ssh commands
	password *string*	*(optional)* the password for telnet and ssh commands
	publickey *string*	*(optional)* the public key for ssh commands
	privatekey *string*	*(optional)* the private key for ssh commands
	parent_taskid *number*	the parent task id
	hostid *number*	target hostid

Example:

proxy→server:

```
{
    "request": "proxy data",
    "host": "Proxy #12",
    "session": "12345678901234567890123456789012",
    "host availability":[
        {
            "hostid":10106,
            "available":1,
```

```
            "error":"",
            "snmp_available":0,
            "snmp_error":"",
            "ipmi_available":0,
            "ipmi_error":"",
            "jmx_available":0,
            "jmx_error":""
        },
        {
            "hostid":10107,
            "available":1,
            "error":"",
            "snmp_available":0,
            "snmp_error":"",
            "ipmi_available":0,
            "ipmi_error":"",
            "jmx_available":0,
            "jmx_error":""
        }
    ],
    "history data":[
        {
            "itemid":"12345",
            "clock":1478609647,
            "ns":332510044,
            "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": 2,
            "clock":1478608371,
            "ttl": 600,
            "commandtype": 2,
            "command": "restart_service1.sh",
            "execute_on": 2,
            "port": 80,
            "authtype": 0,
            "username": "userA",
            "password": "password1",
            "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe",
            "privatekey": "lsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u09lHNsj6tQ5QCqGKukO1De7zhd",
            "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":"5.0.0"
}
```


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": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe",
         "privatekey": "lsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u09lHNsj6tQ5QCqGKukO1De7zhd",
         "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
      }
  ]
}
```


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

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* inervals.

Field	Type	Mandatory	Value
request	*string*	yes	`active checks`
host	*string*	yes	Host name.

Field	Type	Mandatory	Value
version	*string*	yes	The agent version: <major>.<minor>.
host_metadata	*string*	no	The configuration parameter HostMetadata or HostMetadataItem metric value.
interface	*string*	no	The configuration parameter HostInterface or HostInterfaceItem metric value.
ip	*string*	no	The configuration parameter ListenIP first IP if set.
port	*number*	no	The configuration parameter ListenPort value if set and not default agent listening port.

Example:

```
{
  "request": "active checks",
  "host": "Zabbix server",
  "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.

Field	Type	Mandatory	Value
response	*string*	yes	`success` | `failed`
info	*string*	no	Error information in the case of failure.
data	*array of objects*	no	Active check items.
key	*string*	no	Item key with expanded macros.
itemid	*number*	no	Item identifier.
delay	*string*	no	Item update interval.
lastlogsize	*number*	no	Item lastlogsize.
mtime	*number*	no	Item mtime.
refresh_unsupported	*number*	yes	Unsupported item refresh interval.
regexp	*array of objects*	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 sensitivity setting.

Example:

```
{
  "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.

Field	Type	Mandatory	Value
request	*string*	yes	`agent data`
host	*string*	yes	Host name.
version	*string*	yes	The agent version: <major>.<minor>.
session	*string*	yes	Unique session identifier generated each time when agent is started.
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 value event log source.
eventid	*number*	no	The value event log eventid.
severity	*number*	no	The value event log severity.
timestamp	*number*	no	The value event log timestamp.
clock	*number*	yes	The value timestamp (seconds since Epoch).
ns	*number*	yes	The value timestamp nanoseconds.

Example:

```
{
  "request": "agent data",
  "data": [
    {
      "id": 1,
      "itemid": 5678,
      "value": "2.4.0",
      "clock": 1400675595,
      "ns": 76808644
    },
    {
      "id": 2,
      "itemid": 1234,
      "lastlogsize": 112,
      "value": " 19845:20140621:141708.521 Starting Zabbix Agent [<hostname>]. Zabbix 2.4.0 (revision 5000
      "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:

```
{
  "response": "success",
  "info": "processed: 2; failed: 0; total: 2; seconds spent: 0.003534"
}
```

4 Zabbix agent 2 plugin protocol

Zabbix agent 2 protocol is based on code, size and data model.

Code

Type	Size	Comments
Byte	4	Payload type, currently only JSON is supported.

Size

Type	Size	Comments
Byte	4	Size of the current payload in bytes.

Payload data

Type	Size	Comments
Byte	Defined by the *Size* field	JSON formatted data.

Payload data definition

Common data

These parameters are present in all requests/responses:

Name	Type	Comments
id	uint32	For requests - the incrementing identifier used to link requests with responses. Unique within a request direction (i.e. from agent to plugin or from plugin to agent). For responses - ID of the corresponding request.
type	uint32	The request type.

Log request

A request sent by a plugin to write a log message into the agent log file.

direction	plugin → agent
response	no

Parameters specific to log requests:

Name	Type	Comments
severity	uint32	The message severity (log level).

Name	Type	Comments
message	string	The message to log.

Example:

```
{"id":0,"type":1,"severity":3,"message":"message"}
```

Register request

A request sent by the agent during the agent startup phase to obtain provided metrics to register a plugin.

direction	agent → plugin
response	yes

Parameters specific to register requests:

Name	Type	Comments
version	string	The protocol version <major>.<minor>

Example:

```
{"id":1,"type":2,"version":"1.0"}
```

Register response

Plugin's response to the register request.

direction	plugin → agent
response	n/a

Parameters specific to register responses:

Name	Type	Comments
name	string	The plugin name.
metrics	array of strings (optional)	The metrics with descriptions as used in the plugin. Returns RegisterMetrics(). Absent if error is returned.
interfaces	uint32 (optional)	The bit mask of plugin's supported interfaces. Absent if error is returned.
error	string (optional)	An error message returned if a plugin cannot be started. Absent, if metrics are returned.

Examples:

```
{"id":2,"type":3,"metrics":["external.test", "External exporter Test."], "interfaces": 4}
```

or

```
{"id":2,"type":3,"error":"error message"}
```

Start request

A request to execute the Start function of the Runner interface.

direction	agent → plugin
response	no

The request doesn't have specific parameters, it only contains common data parameters.

Example:

```
{"id":3,"type":4}
```

Terminate request

A request sent by the agent to shutdown a plugin.

direction	agent → plugin
response	no

The request doesn't have specific parameters, it only contains common data parameters.

Example:

```
{"id":3,"type":5}
```

Export request

A request to execute the Export function of the Exporter interface.

direction	agent → plugin
response	no

Parameters specific to export requests:

Name	Type	Comments
key	string	The plugin key.
parameters	array of strings (optional)	The parameters for Export function.

Example:

```
{"id":4,"type":6,"key":"test.key","parameters":["foo","bar"]}
```

Export response

Response from the Export function of the Exporter interface.

direction	plugin → agent
response	n/a

Parameters specific to export responses:

Name	Type	Comments
value	string (optional)	Response value from the Export function. Absent, if error is returned.
error	string (optional)	Error message if the Export function has not been executed successfully. Absent, if value is returned.

Examples:

```
{"id":5,"type":7,"value":"response"}
```

or

```
{"id":5,"type":7,"error":"error message"}
```

Configure request

A request to execute the *Configure* function of the *Configurator* interface.

direction	agent → plugin
response	n/a

Parameters specific to *Configure* requests:

Name	Type	Comments
global_options	JSON object	JSON object containing global agent configuration options.
private_options	JSON object (optional)	JSON object containing private plugin configuration options, if provided.

Example:

```
{"id":6,"type":8,"global_options":{...},"private_options":{...}}
```

Validate request

A request to execute *Validate* function of the *Configurator* interface.

direction	agent → plugin
response	yes

Parameters specific to *Validate* requests:

Name	Type	Comments
private_options	JSON object (optional)	JSON object containing private plugin configuration options, if provided.

Example:

```
{"id":7,"type":9,"private_options":{...}}
```

Validate response

Response from *Validate* function of *Configurator* interface.

direction	plugin → agent
response	n/a

Parameters specific to *Validate* responses:

Name	Type	Comments
error	string (optional)	An error message returned if the Validate function is not executed successfully. Absent if executed successfully.

Example:

```
{"id":8,"type":10}
```

or

```
{"id":8,"type":10,"error":"error message"}
```

5 Zabbix sender protocol

Please refer to the trapper item page for more information.

6 Header

Overview

The header is present in all request and response messages between Zabbix components. It is required to determine the message length, if it is compressed or not, if it is a large packet or not.

Zabbix communications protocol has 1GB packet size limit per connection. The limit of 1GB is applied to both the received packet data length and the uncompressed data length.

When sending configuration to Zabbix proxy, the packet size limit is increased to 4GB to allow syncing large configurations. When data length before compression exceeds 4GB, Zabbix server automatically starts using the large packet format (0x04 flag) which increases the packet size limit to 16GB.

Note that while a large packet format can be used for sending any data, currently only the Zabbix proxy configuration syncer can handle packets that are larger than 1GB.

Structure

The header consists of four fields. All numbers in the header are formatted as little-endian.

Field	Size	Size (large packet)	Description
<PROTOCOL>	4	4	"ZBXD" or 5A 42 58 44
<FLAGS>	1	1	Protocol flags: 0x01 - Zabbix communications protocol 0x02 - compression 0x04 - large packet
<DATALEN>	4	8	Data length.
<RESERVED>	4	8	When compression is used (0x02 flag) - the length of uncompressed data When compression is not used - 00 00 00 00

Examples

Here are some 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. These code snippets assume that the data is not larger than 1GB, thus the large packet format is not used.

Python

```
packet = b"ZBXD\1" + struct.pack("<II", len(data), 0) + data
```

or

```
def zbx_create_header(plain_data_size, compressed_data_size=None):
    protocol = b"ZBXD"
    flags = 0x01
    if compressed_data_size is None:
        datalen = plain_data_size
        reserved = 0
    else:
        flags |= 0x02
        datalen = compressed_data_size
        reserved = plain_data_size
    return protocol + struct.pack("<BII", flags, datalen, reserved)

packet = zbx_create_header(len(data)) + data
```

Perl

```
my $packet = "ZBXD\1" . pack("(II)<", length($data), 0) . $data;
```

or

```
sub zbx_create_header($;$)
{
```

```
    my $plain_data_size = shift;
    my $compressed_data_size = shift;

    my $protocol = "ZBXD";
    my $flags = 0x01;
    my $datalen;
    my $reserved;

    if (!defined($compressed_data_size))
    {
        $datalen = $plain_data_size;
        $reserved = 0;
    }
    else
    {
        $flags |= 0x02;
        $datalen = $compressed_data_size;
        $reserved = $plain_data_size;
    }

    return $protocol . chr($flags) . pack("(II)<", $datalen, $reserved);
}

my $packet = zbx_create_header(length($data)) . $data;
```

PHP

```
$packet = "ZBXD\1" . pack("VV", strlen($data), 0) . $data;
```

or

```
function zbx_create_header($plain_data_size, $compressed_data_size = null)
{
    $protocol = "ZBXD";
    $flags = 0x01;
    if (is_null($compressed_data_size))
    {
        $datalen = $plain_data_size;
        $reserved = 0;
    }
    else
    {
        $flags |= 0x02;
        $datalen = $compressed_data_size;
        $reserved = $plain_data_size;
    }
    return $protocol . chr($flags) . pack("VV", $datalen, $reserved);
}

$packet = zbx_create_header(strlen($data)) . $data;
```

Bash

```
datalen=$(printf "%08x" ${#data})
datalen="\\x${datalen:6:2}\\x${datalen:4:2}\\x${datalen:2:2}\\x${datalen:0:2}"
printf "ZBXD\1${datalen}\0\0\0\0%s" "$data"
```

7 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:

Field		Type	Description
clock		number	Number of seconds since Epoch to the moment when problem was detected (integer part).
ns		number	Number of nanoseconds to be added to `clock` to get a precise problem detection time.
value		number	1 (always).
eventid		number	Problem event ID.
name		string	Problem event name.
severity		number	Problem event severity (0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster). This property is exported since Zabbix 5.0.12.
hosts		array	List of hosts involved in the trigger expression; there should be at least one element in array.
-		object	
	host	string	Host name.
	name	string	Visible host name.
groups		array	List of host groups of all hosts involved in the trigger expression; there should be at least one element in array.
-		string	Host group name.
tags		array	List of problem tags (can be empty).
-		object	
	tag	string	Tag name.
	value	string	Tag value (can be empty).

The following information is exported for a recovery event:

Field	Type	Description
clock	number	Number of seconds since Epoch to the moment when problem was resolved (integer part).
ns	number	Number of nanoseconds to be added to `clock` to get a precise problem resolution time.
value	number	0 (always).
eventid	number	Recovery event ID.
p_eventid	number	Problem event ID.

Examples

Problem:

```
{"clock":1519304285,"ns":123456789,"value":1,"name":"Either Zabbix agent is unreachable on Host B or polle
```

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 Ho
```

```
{"clock":1519304286,"ns":123456789,"value":1,"eventid":43,"name":"Either Zabbix agent is unreachable on Ho
```

Recovery:

```
{"clock":1519304346,"ns":987654321,"value":0,"eventid":44,"p_eventid":43}
```

```
{"clock":1519304346,"ns":987654321,"value":0,"eventid":44,"p_eventid":42}
```

Item values

The following information is exported for a collected item value:

Field		Type	Description
host		object	Host name of the item host.
	host	string	Host name.
	name	string	Visible host name.
groups		array	List of host groups of the item host; there should be at least one element in array.
	-	string	Host group name.
applications		array	List of the item applications; empty if there are none.
	-	string	Application name.
itemid		number	Item ID.
name		string	Visible item name.
clock		number	Number of seconds since Epoch to the moment when value was collected (integer part).
ns		number	Number of nanoseconds to be added to `clock` to get a precise value collection time.
timestamp (*Log* only)		number	0 if not available.
source (*Log* only)		string	Empty string if not available.
severity (*Log* only)		number	0 if not available.
eventid (*Log* only)		number	0 if not available.
value		number (for numeric items) or string (for text items)	Collected item value.
type		number	Collected value type: 0 - numeric float, 1 - character, 2 - log, 3 - numeric unsigned, 4 - text

Examples

Numeric (unsigned) value:

```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"applications":
```

Numeric (float) value:

```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"applications":
```

Character, text value:

```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"applications":
```

Log value:

```
{"host":{"host":"Host A","name":"Host A visible"},"groups":["Group X","Group Y","Group Z"],"applications":
```

Trends

The following information is exported for a calculated trend value:

Field		Type	Description
host		object	Host name of the item host.
	host	string	Host name.
	name	string	Visible host name.
groups		array	List of host groups of the item host; there should be at least one element in array.
	-	string	Host group name.
applications		array	List of the item applications; empty if there are none.
	-	string	Application name.
itemid		number	Item ID.
name		string	Visible item name.
clock		number	Number of seconds since Epoch to the moment when value was collected (integer part).
count		number	Number of values collected for a given hour.

Field	Type	Description
min	number	Minimum item value for a given hour.
avg	number	Average item value for a given hour.
max	number	Maximum item value for a given hour.
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"],"applications":
```

Numeric (float) value:

```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"applications":
```

5 Items

1 Items supported by platform

The table displays support for Zabbix agent items on various platforms:

- Items marked with "**X**" are supported, the ones marked with "**-**" are not supported.
- If an item is marked with "**?**", it is not known whether it is supported or not.
- If an item is marked with "**r**", it means that it requires root privileges.
- Parameters that are included in angle brackets **<**like_this**>** are optional.

Note:
Windows-only Zabbix agent items are not included in this table.

NetBSD											▼▼
OpenBSD										▼▼	
Mac									▼▼		
OS X											
Tru64								▼▼			
AIX							▼▼				
HP-UX						▼▼					
Solaris					▼▼						
FreeBSD				▼▼							
Linux 2.6 (and later)			▼▼								
Linux 2.4		▼▼									
Windows	▼▼										
▼ Item ▼	1	2	3	4	5	6	7	8	9	10	11
agent.hostname	X	X	X	X	X	X	X	X	X	X	X
agent.ping	X	X	X	X	X	X	X	X	X	X	X
agent.variant	X	X	X	X	X	X	X	X	X	X	X
agent.version	X	X	X	X	X	X	X	X	X	X	X
kernel.maxfiles	-	X	X	X	-	-	-	?	X	X	X
kernel.maxproc	-	-	X	X	X	-	-	?	X	X	X
log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<persistent_dir>]	X[4]	X	X	X	X	X	X	X	X	X	X
persistent_dir	-	X	X	X	X	X	X	X	X	X	X
▲ **log.count[file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<persistent_dir>]**	X[4]	X	X	X	X	X	X	X	X	X	X

persistent_dir		-	X	X	X	X	X	X	X	X	X	X
▲												
logrt[file_regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>,<persistent_dir>]		X[4]	X	X	X	X	X	X	X	X	X	X
persistent_dir		-	X	X	X	X	X	X	X	X	X	X
▲												
logrt.count[file_regexp,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>,<persistent_dir>]		X[4]	X	X	X	X	X	X	X	X	X	X
persistent_dir		-	X	X	X	X	X	X	X	X	X	X
▲												
net.dns[<ip>,zone,<type>,<timeout>,<count>]		X	X	X	X	X	X	X	X	X	X	X
net.dns.record[<ip>,zone,<type>,<timeout>,<count>]		X	X	X	X	X	X	X	X	X	X	X
net.if.collisions[if]		X	X	X	X	X	-	X	-	X	X	r
net.if.discovery		X	X	X	X	X	X	X	-	-	X	X
net.if.in[if,<mode>]		X	X	X	X	X	X[1]	X	-	X	X	r
mode	bytes	X	X	X	X	X	X[2]	X	-	X	X	r
▲	*(de-fault)*											
	packets	X	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X	X[2]	X	-	X	X	r
	dropped	X	X	X	X	-	X	-	-	X	X	r
	overruns	-	X	X	-	-	-	-	-	-	-	-
	frame	-	X	X	-	-	-	-	-	-	-	-
	compressed	-	X	X	-	-	-	-	-	-	-	-
	multicast	-	X	X	-	-	-	-	-	-	-	-
net.if.out[if,<mode>]		X	X	X	X	X	X[1]	X	-	X	X	r
mode	bytes	X	X	X	X	X	X[2]	X	-	X	X	r
▲	*(de-fault)*											
	packets	X	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X	X[2]	X	-	X	X	r
	dropped	X	X	X	-	-	X	-	-	-	-	-
	overruns	-	X	X	-	-	-	-	-	-	-	-
	collision	-	X	X	-	-	-	-	-	-	-	-
	carrier	-	X	X	-	-	-	-	-	-	-	-
	compressed	-	X	X	-	-	-	-	-	-	-	-
net.if.total[if,<mode>]		X	X	X	X	X	X[1]	X	-	X	X	r
mode	bytes	X	X	X	X	X	X[2]	X	-	X	X	r
▲	*(de-fault)*											
	packets	X	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X	X[2]	X	-	X	X	r
	dropped	X	X	X	-	-	X	-	-	-	-	-
	overruns	-	X	X	-	-	-	-	-	-	-	-
	compressed	-	X	X	-	-	-	-	-	-	-	-
net.tcp.listen[port]		X	X	X	X	X	-	-	-	X	-	-
net.tcp.port[<ip>,port]		X	X	X	X	X	X	X	X	X	X	X
net.tcp.service[service,<ip>,<port>]		X	X	X	X	X	X	X	X	X	X	X
net.tcp.service.perf[service,<ip>,<port>]		X	X	X	X	X	X	X	X	X	X	X
net.udp.listen[port]		X	X	X	X	X	-	-	-	X	-	-
net.udp.service[service,<ip>,<port>]		X	X	X	X	X	X	X	X	X	X	X
net.udp.service.perf[service,<ip>,<port>]		X	X	X	X	X	X	X	X	X	X	X
		1	2	3	4	5	6	7	8	9	10	11
proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]		X	X	X	-	X[3]	-	-	-	-	-	-
type	total	-	X	X	-	X	-	-	-	-	-	-
▲	*(de-fault)*											
	user	-	X	X	-	X	-	-	-	-	-	-
	system	-	X	X	-	X	-	-	-	-	-	-
mode	avg1	-	X	X	-	X	-	-	-	-	-	-
▲	*(de-fault)*											
	avg5	-	X	X	-	X	-	-	-	-	-	-
	avg15	-	X	X	-	X	-	-	-	-	-	-

zone ▲	current *(default)*	-	-	-	-	X	-	-	-	-	-	-
	all	-	-	-	-	X	-	-	-	-	-	-
proc.mem[<name>,<user>,<mode>,<cmdline><memtype>]		X	X	X	X	X[3]	X	X	-	X	X	
mode ▲	sum *(default)*	-	X	X	X	X	-	X	X	-	X	X
	avg	-	X	X	X	X	-	X	X	-	X	X
	max	-	X	X	X	X	-	X	X	-	X	X
	min	-	X	X	X	X	-	X	X	-	X	X
memtype ▲		-	X	X	X	X	-	X	-	-	-	-
proc.num[<name>,<user>,<state>,<cmdline><zone>]		X	X	X	X	X[3]	X	X	X	-	X	X
state ▲	all *(default)*	-	X	X	X	X	X	X	X	-	X	X
	disk	-	X	X	X	-	-	-	-	-	X	X
	sleep	-	X	X	X	X	X	X	X	-	X	X
	zomb	-	X	X	X	X	X	X	X	-	X	X
	run	-	X	X	X	X	X	X	X	-	X	X
	trace	-	X	X	X	-	-	-	-	-	X	X
cmdline ▲		-	X	X	X	X	X	X	X	-	X	X
zone ▲	current *(default)*	-	-	-	-	X	-	-	-	-	-	-
	all	-	-	-	-	X	-	-	-	-	-	-
sensor[device,sensor,<mode>]		-	X	X	-	-	-	-	-	-	X	-
system.boottime		-	X	X	X	X	-	-	-	X	X	X
system.cpu.discovery		X	X	X	X	X	X	X	X	X	X	X
system.cpu.intr		-	X	X	X	X	-	X	-	-	X	X
system.cpu.load[<cpu>,<mode>]		X	X	X	X	X	X	X	X	X	X	X
cpu ▲	all *(default)*	X	X	X	X	X	X	X	X	X	X	X
	percpu	X	X	X	X	X	X	X	-	X	X	X
mode ▲	avg1 *(default)*	X	X	X	X	X	X	X	X	X	X	X
	avg5	X	X	X	X	X	X	X	X	X	X	X
	avg15	X	X	X	X	X	X	X	X	X	X	X
system.cpu.num[<type>]		X	X	X	X	X	X	X	-	X	X	X
type ▲	online *(default)*	X	X	X	X	X	X	X	-	X	X	X
	max	-	X	X	X	X	-	-	-	X	-	-
system.cpu.switches		-	X	X	X	X	-	X	-	-	X	X
system.cpu.util[<cpu>,<type>,<mode>,<logical_or_physical>]		X	X	X	X	X	X	X	X	-	X	X
type ▲	user *(default)*	-	X	X	X	X	X	X	X	-	X	X
	nice	-	X	X	X	-	X	-	X	-	X	X
	idle	-	X	X	X	X	X	X	X	-	X	X
	system *(default for Windows)*	X	X	X	X	X	X	X	X	-	X	X
	iowait	-	-	X	-	X	-	X	-	-	-	-
	interrupt	-	-	X	X	-	-	-	-	-	X	-

	softirq	-	-	X	-	-	-	-	-	-	-	-
	steal	-	-	X	-	-	-	-	-	-	-	-
	guest	-	-	X	-	-	-	-	-	-	-	-
	guest_nice	-	-	X	-	-	-	-	-	-	-	-
mode ▲	avg1 *(default)*	X	X	X	X	X	X	X	X	-	X	X
	avg5	X	X	X	X	X	X	X	-	-	X	X
	avg15	X	X	X	X	X	X	X	-	-	X	X
logical_or_physical ▲ *(since Zabbix 5.0.3)*	logical *(default)*	-	-	-	-	-	-	X	-	-	-	-
	physical	-	-	-	-	-	-	X	-	-	-	-
		1	2	3	4	5	6	7	8	9	10	11
system.hostname[<type>,<transform>]		X	X	X	X	X	X	X	X	X	X	X
system.hw.chassis[<info>]		-	X	X	-	-	-	-	-	-	-	-
system.hw.cpu[<cpu>,<info>]		X	X	X	-	-	-	-	-	-	-	-
system.hw.devices[<type>]		X	X	X	-	-	-	-	-	-	-	-
system.hw.macaddr[<interface>,<format>]		-	X	X	-	-	-	-	-	-	-	-
system.localtime[<type>]		X	X	X	X	X	X	X	X	X	X	X
type ▲	utc *(default)*	X	X	X	X	X	X	X	X	X	X	X
	local	X	X	X	X	X	X	X	X	X	X	X
system.run[command,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	wait *(default)*	X	X	X	X	X	X	X	X	X	X	X
	nowait	X	X	X	X	X	X	X	X	X	X	X
system.stat[resource,<type>]		-	-	-	-	-	-	X	-	-	-	-
system.sw.arch		X	X	X	X	X	X	X	X	X	X	X
system.sw.os[<info>]		-	X	X	-	-	-	-	-	-	-	-
system.sw.packages[<package>,<manager>,<format>]		-	X	X	-	-	-	-	-	-	-	-
system.swap.in[<device>,<type>] *(specifying a device is only supported under Linux)*		X	X	X	-	X	-	-	-	-	X	-
type ▲ *(pages will only work if device was not specified)*	count *(default under all except Linux)*	-	X	X	-	X	-	-	-	-	X	-
	sectors	-	X	X	-	-	-	-	-	-	-	-
	pages *(default under Linux)*	-	X	X	-	X	-	-	-	-	X	-

system.swap.out[**<device>,<type>**] *(specifying a device is only supported under Linux)*		-	X	X	-	X	-	-	-	-	X	-
type ▲ *(pages will only work if device was not specified)*	count *(default under all except Linux)*	-	X	X	-	X	-	-	-	-	X	-
	sectors	-	X	X	-	-	-	-	-	-	-	-
	pages *(default under Linux)*	-	X	X	-	X	-	-	-	-	X	-
system.swap.size[**<device>,<type>**] *(specifying a device is only supported under FreeBSD, for other platforms must be empty or "all")*		X	X	X	X	X	-	X	X	-	X	-
type ▲	free *(default)*	X	X	X	X	X	-	X	X	-	X	-
	total	X	X	X	X	X	-	X	X	-	X	-
	used	X	X	X	X	X	-	X	X	-	X	-
	pfree	X	X	X	X	X	-	X	X	-	X	-
	pused	X [6]	X	X	X	X	-	X	X	-	X	-
system.uname		X	X	X	X	X	X	X	X	X	X	X
system.uptime		X	X	X	X	X	-	X	?	X	X	X
system.users.num		-	X	X	X	X	X	X	X	X	X	X
systemd.unit.discovery		X	X	-	-	-	-	-	-	-	-	-
systemd.unit.get		-	X	X	-	-	-	-	-	-	-	-
systemd.unit.info		-	X	X	-	-	-	-	-	-	-	-
		1	2	3	4	5	6	7	8	9	10	11
vfs.dev.discovery		-	X	X	-	-	-	-	-	-	-	-
vfs.dev.read[**<device>,<type>,<mode>**]		X	X	X	X	X	-	X	-	-	X	-
type ▲	sectors	-	X	X	-	-	-	-	-	-	-	-

	operations *(default for OpenBSD, AIX)*	-	X	X	X	X	-	X	-	-	X	-
	bytes *(default for Solaris)*	-	-	-	X	X	-	X	-	-	X	-
	sps *(default for Linux)*	-	X	X	-	-	-	-	-	-	-	-
	ops	-	X	X	X	-	-	-	-	-	-	-
	bps *(default for FreeBSD)*	-	-	-	X	-	-	-	-	-	-	-
mode ▲ *(compatible only with type in: sps, ops, bps)*	avg1 *(default)*	-	X	X	X	-	-	-	-	-	-	-
	avg5	-	X	X	X	-	-	-	-	-	-	-
	avg15	-	X	X	X	-	-	-	-	-	-	-
vfs.dev.write[<device>,<type>,<mode>]			X	X	X	X	-	X	-	-	X	-
type ▲	sectors	-	X	X	-	-	-	-	-	-	-	-
	operations *(default for OpenBSD, AIX)*	-	X	X	X	X	-	X	-	-	X	-
	bytes *(default for Solaris)*	-	-	-	X	X	-	X	-	-	X	-
	sps *(default for Linux)*	-	X	X	-	-	-	-	-	-	-	-
	ops	-	X	X	X	-	-	-	-	-	-	-
	bps *(default for FreeBSD)*	-	-	-	X	-	-	-	-	-	-	-

mode ▲ *(compatible only with type in: sps, ops, bps)*	avg1 *(default)*	-	X	X	X	-	-	-	-	-	-	-
	avg5	-	X	X	X	-	-	-	-	-	-	-
	avg15	-	X	X	X	-	-	-	-	-	-	-
vfs.dir.count[dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age		X	X	X	X	X	X	X	X	X	X	X
vfs.dir.size[dir,<regex_incl>,<regex_excl>,<mode>,<max_depth>,<regex_excl_dir>]		?	X	X	X	?	?	?	?	?	?	?
vfs.file.cksum[file]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.contents[file,<encoding>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.exists[file,<types_incl>,<types_excl>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.md5sum[file]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.regexp[file,regexp,<encoding>,<output>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.regmatch[file,regexp,<encoding>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.size[file]		X	X	X	X	X	X	X	X	X	X	X
		1	2	3	4	5	6	7	8	9	10	11
vfs.file.time[file,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲ *(default)*	modify	X	X	X	X	X	X	X	X	X	X	X
	access	X	X	X	X	X	X	X	X	X	X	X
	change	X[5]	X	X	X	X	X	X	X	X	X	X
vfs.fs.discovery		X	X	X	X	X	X	X	-	X	X	X
vfs.fs.get		X	X	X	X	X	X	X	-	X	X	X
vfs.fs.inode[fs,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲ *(default)*	total	-	X	X	X	X	X	X	X	X	X	X
	free	-	X	X	X	X	X	X	X	X	X	X
	used	-	X	X	X	X	X	X	X	X	X	X
	pfree	-	X	X	X	X	X	X	X	X	X	X
	pused	-	X	X	X	X	X	X	X	X	X	X
vfs.fs.size[fs,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲ *(default)*	total	X	X	X	X	X	X	X	X	X	X	X
	free	X	X	X	X	X	X	X	X	X	X	X
	used	X	X	X	X	X	X	X	X	X	X	X
	pfree	X	X	X	X	X	X	X	X	X	X	X
	pused	X	X	X	X	X	X	X	X	X	X	X
vm.memory.size[<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲ *(default)*	total	X	X	X	X	X	X	X	X	X	X	X
	active	-	-	-	X	-	X	-	-	X	X	X
	anon	-	-	-	-	-	-	-	-	-	-	X
	buffers	-	X	X	X	-	-	-	-	-	X	X
	cached	X	X	X	X	-	-	X	-	-	X	X
	exec	-	-	-	-	-	-	-	-	-	-	X
	file	-	-	-	-	-	-	-	-	-	-	X
	free	X	X	X	X	X	X	X	X	X	X	X
	inactive	-	-	-	X	-	-	-	-	X	X	X
	pinned	-	-	-	-	-	-	X	-	-	-	-
	shared	-	X	-	X	-	-	-	-	-	X	X
	wired	-	-	-	X	-	-	-	-	X	X	X
	used	X	X	X	X	X	X	X	X	X	X	X
	pused	X	X	X	X	X	X	X	X	X	X	X

available	X	X	X	X	X	X	X	X	X	X	X
pavailable	X	X	X	X	X	X	X	X	X	X	X
web.page.get[host,<path>,<port>]	X	X	X	X	X	X	X	X	X	X	X
web.page.perf[host,<path>,<port>]	X	X	X	X	X	X	X	X	X	X	X
web.page.regexp[host,<path>,<port>,regexp,<length>,<output>]	X	X	X	X	X	X	X	X	X	X	X
	1	2	3	4	5	6	7	8	9	10	11

Note:
See also a description of vm.memory.size parameters.

Footnotes

[1] net.if.in, net.if.out and net.if.total items do not provide statistics of loopback interfaces (e.g. lo0).

[2] These values for these items 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.

[3] 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.

[4] Not supported on Windows Event Log.

[5] On Windows XP vfs.file.time[file,change] may be equal to vfs.file.time[file,access].

[6] Supported only by Zabbix agent 2 since Zabbix 5.0.5; not supported by Zabbix agent.

2 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.

Warning:
Some of these parameters are platform-specific and might not be available on your platform. See Items supported by platform for details.

Platform-specific calculation of **available** and **used**:

Platform	**"available"**	**"used"**
AIX	free + cached	real memory in use
FreeBSD	inactive + cached + free	active + wired + cached
HP UX	free	total - free
Linux<3.14	free + buffers + cached	total - free
Linux 3.14+ (also backported to 3.10 on RHEL 7)	/proc/meminfo, see "MemAvailable" in Linux kernel documentation for details. 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.	total - free
NetBSD	inactive + execpages + file + free	total - free
OpenBSD	inactive + free + cached	active + wired
OSX	inactive + free	active + wired
Solaris	free	total - free
Win32	free	total - free

Attention:
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 instantly 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

3 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] No such file or directory**
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).

> **Note:**
> 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

```
{
    "request":"active checks",
    "host":"<hostname>"
}
```

Server response

```
{
    "response":"success",
    "data":[
        {
            "key":"log[/home/zabbix/logs/zabbix_agentd.log]",
            "delay":30,
            "lastlogsize":0,
            "mtime":0
        },
        {
            "key":"agent.version",
            "delay":600,
            "lastlogsize":0,
            "mtime":0
        },
        {
            "key":"vfs.fs.size[/nono]",
            "delay":600,
            "lastlogsize":0,
            "mtime":0
        }
    ]
}
```

The server must respond with success. For each returned item, all properties **key**, **delay**, **lastlogsize** and **mtime** must exist, regardless of whether item is a log item or not.

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

Attention:
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


```
{
    "request":"agent data",
    "session": "12345678901234567890123456789012",
    "data":[
        {
            "host":"<hostname>",
            "key":"agent.version",
            "value":"2.4.0",
            "id": 1,
            "clock":1400675595,
            "ns":76808644
        },
        {
            "host":"<hostname>",
            "key":"log[/home/zabbix/logs/zabbix_agentd.log]",
            "lastlogsize":112,
            "value":" 19845:20140621:141708.521 Starting Zabbix Agent [<hostname>]. Zabbix 2.4.0 (revision
            "id": 2,
            "clock":1400675595,
            "ns":77053975
        },
        {
            "host":"<hostname>",
            "key":"vfs.fs.size[/nono]",
            "state":1,
            "value":"Cannot obtain filesystem information: [2] No such file or directory",
            "id": 3,
            "clock":1400675595,
            "ns":78154128
        }
    ],
    "clock": 1400675595,
    "ns": 78211329
}
```


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


```
{
    "response":"success",
    "info":"processed: 3; failed: 0; total: 3; seconds spent: 0.003534"
}
```


Attention:
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.

Attention:
Error message will be trimmed to 2048 symbols on server side.

Older XML protocol

Note:
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.

4 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

```
{
    "request":"sender data",
    "data":[
        {
            "host":"<hostname>",
            "key":"trap",
            "value":"test value"
        }
    ]
}
```

Zabbix server response

```
{
    "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.

```
{
    "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
        }
    ],
    "clock":1516712029,
    "ns":873386094
}
```

Zabbix server response

```
{
    "response":"success",
    "info":"processed: 2; failed: 0; total: 2; seconds spent: 0.060904"
}
```

5 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.

Note:
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

Item key	User group	
	Recommended	Minimum technically acceptable (functionality is limited)
agent.hostname	Guests	Guests
agent.ping	Guests	Guests
agent.variant	Guests	Guests
agent.version	Guests	Guests
log	Administrators	Guests
log.count	Administrators	Guests
logrt	Administrators	Guests
logrt.count	Administrators	Guests
net.dns	Guests	Guests
net.dns.record	Guests	Guests
net.if.discovery	Guests	Guests
net.if.in	Guests	Guests
net.if.out	Guests	Guests
net.if.total	Guests	Guests
net.tcp.listen	Guests	Guests
net.tcp.port	Guests	Guests
net.tcp.service	Guests	Guests
net.tcp.service.perf	Guests	Guests
net.udp.service	Guests	Guests
net.udp.service.perf	Guests	Guests
proc.num	Administrators	Guests
system.cpu.discovery	Performance Monitor Users	Performance Monitor Users
system.cpu.load	Performance Monitor Users	Performance Monitor Users
system.cpu.num	Guests	Guests
system.cpu.util	Performance Monitor Users	Performance Monitor Users
system.hostname	Guests	Guests
system.localtime	Guests	Guests
system.run	Administrators	Guests
system.sw.arch	Guests	Guests
system.swap.size	Guests	Guests
system.uname	Guests	Guests
system.uptime	Performance Monitor Users	Performance Monitor Users
vfs.dir.count	Administrators	Guests
vfs.dir.size	Administrators	Guests
vfs.file.cksum	Administrators	Guests

Item key	User group	
vfs.file.contents	Administrators	Guests
vfs.file.exists	Administrators	Guests
vfs.file.md5sum	Administrators	Guests
vfs.file.regexp	Administrators	Guests
vfs.file.regmatch	Administrators	Guests
vfs.file.size	Administrators	Guests
vfs.file.time	Administrators	Guests
vfs.fs.discovery	Administrators	Guests
vfs.fs.size	Administrators	Guests
vm.memory.size	Guests	Guests
web.page.get	Guests	Guests
web.page.perf	Guests	Guests
web.page.regexp	Guests	Guests
zabbix.stats	Guests	Guests

Windows-specific item keys

Item key	User group	
	Recommended	Minimum technically acceptable (functionality is limited)
eventlog	Event Log Readers	Guests
net.if.list	Guests	Guests
perf_counter	Performance Monitor Users	Performance Monitor Users
proc_info	Administrators	Guests
service.discovery	Guests	Guests
service.info	Guests	Guests
services	Guests	Guests
wmi.get	Administrators	Guests
vm.vmemory.size	Guests	Guests

6 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 back-ends 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

7 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.

8 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/). **Sensors-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 **sensor** 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*. According to specification, these are voltages on chip pins and generally speaking may need scaling.

```
$ 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.

9 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.

Supported value	Description	Source in procentry64 structure	Tries to be compatible with
vsize [1]	Virtual memory size	pi_size	
pmem	Percentage of real memory	pi_prm	ps -o pmem
rss	Resident set size	pi_trss + pi_drss	ps -o rssize
size	Size of process (code + data)	pi_dvm	"ps gvw" SIZE column
dsize	Data size	pi_dsize	
tsize	Text (code) size	pi_tsize	"ps gvw" TSIZ column
sdsize	Data size from shared library	pi_sdsize	
drss	Data resident set size	pi_drss	
trss	Text resident set size	pi_trss	

Notes for AIX:

1. When choosing parameters for proc.mem[] item key on AIX, try to specify narrow process selection criteria. Otherwise there is a risk of getting unwanted processes counted into proc.mem[] result.

Example:

```
$ zabbix_agentd -t proc.mem[,,,NonExistingProcess,rss]
proc.mem[,,,NonExistingProcess,rss]           [u|2879488]
```

This example shows how specifying only command line (regular expression to match) parameter results in Zabbix agent self-accounting - probably not what you want.

2. Do not use "ps -ef" to browse processes - it shows only non-kernel processes. Use "ps -Af" to see all processes which will be seen by Zabbix agent.

3. Let's go through example of 'topasrec' how Zabbix agent proc.mem[] selects processes.

```
$ ps -Af | grep topasrec
root 10747984        1   0   Mar 16      -  0:00 /usr/bin/topasrec  -L -s 300 -R 1 -r 6 -o /var/perf daily
```

proc.mem[] has arguments:

```
proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]
```

The 1st criterion is a process name (argument <name>). In our example Zabbix agent will see it as 'topasrec'. In order to match, you need to either specify 'topasrec' or to leave it empty. The 2nd criterion is a user name (argument <user>). To match, you need to either specify 'root' or to leave it empty. The 3rd criterion used in process selection is an argument <cmd-line>. Zabbix agent will see its value as '/usr/bin/topasrec -L -s 300 -R 1 -r 6 -o /var/perf/daily/ -ypersistent=1 -O type=bin -ystart_time=04:08:54,Mar16,2023'. To match, you need to either specify a regular expression which matches this string or to leave it empty.

Arguments <mode> and <memtype> are applied after using the three criteria mentioned above.

FreeBSD

See values supported for 'memtype' parameter on FreeBSD in the table.

Supported value	Description	Source in kinfo_proc structure	Tries to be compatible with
vsize	Virtual memory size	kp_eproc.e_vm.vm_map.size or ki_size	ps -o vsz
pmem	Percentage of real memory	calculated from rss	ps -o pmem
rss	Resident set size	kp_eproc.e_vm.vm_rssize or ki_rssize	ps -o rss
size [1]	Size of process (code + data + stack)	tsize + dsize + ssize	
tsize	Text (code) size	kp_eproc.e_vm.vm_tsize or ki_tsize	ps -o tsiz
dsize	Data size	kp_eproc.e_vm.vm_dsize or ki_dsize	ps -o dsiz
ssize	Stack size	kp_eproc.e_vm.vm_ssize or ki_ssize	ps -o ssiz

Linux

See values supported for 'memtype' parameter on Linux in the table.

Supported value	Description	Source in /proc/<pid>/status file
vsize [1]	Virtual memory size	VmSize
pmem	Percentage of real memory	(VmRSS/total_memory) * 100
rss	Resident set size	VmRSS
data	Size of data segment	VmData
exe	Size of code segment	VmExe
hwm	Peak resident set size	VmHWM
lck	Size of locked memory	VmLck
lib	Size of shared libraries	VmLib
peak	Peak virtual memory size	VmPeak
pin	Size of pinned pages	VmPin
pte	Size of page table entries	VmPTE

Supported value	Description	Source in /proc/<pid>/status file
size	Size of process code + data + stack segments	VmExe + VmData + VmStk
stk	Size of stack segment	VmStk
swap	Size of swap space used	VmSwap

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.

Supported value	Description	Source in psinfo structure	Tries to be compatible with
vsize [1]	Size of process image	pr_size	ps -o vsz
pmem	Percentage of real memory	pr_pctmem	ps -o pmem
rss	Resident set size It may be underestimated - see rss description in "man ps".	pr_rssize	ps -o rss

Footnotes

1 Default value.

10 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` command shows processes of interest as

```
$ ps -fu zabbix
UID        PID  PPID  C STIME TTY          TIME CMD
...
zabbix    6318     1  0 12:01 ?        00:00:00 sbin/zabbix_agentd -c /home/zabbix/ZBXNEXT-1078/zabbix_age
zabbix    6319  6318  0 12:01 ?        00:00:01 sbin/zabbix_agentd: collector [idle 1 sec]
zabbix    6320  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #1 [waiting for connection]
zabbix    6321  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #2 [waiting for connection]
zabbix    6322  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #3 [waiting for connection]
zabbix    6323  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: active checks #1 [idle 1 sec]
...
```

Selecting processes by name and user does the job:

```
$ zabbix_get -s localhost -k 'proc.num[zabbix_agentd,zabbix]'
6
```

Now let's rename `zabbix_agentd` executable to `zabbix_agentd_30` and restart it.

`ps` now shows

```
$ ps -fu zabbix
UID        PID  PPID  C STIME TTY          TIME CMD
...
zabbix    6715     1  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30 -c /home/zabbix/ZBXNEXT-1078/zabbix_
zabbix    6716  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: collector [idle 1 sec]
zabbix    6717  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #1 [waiting for connection
zabbix    6718  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #2 [waiting for connection
zabbix    6719  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #3 [waiting for connection
zabbix    6720  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: active checks #1 [idle 1 sec]
...
```

Now selecting processes by name and user produces an incorrect result:

```
$ zabbix_get -s localhost -k 'proc.num[zabbix_agentd_30,zabbix]'
1
```

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
/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
```

The process name in status file is truncated to 15 characters.

A similar result can be seen with ps command:

```
$ 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

```
$ for i in 6715 6716 6717 6718 6719 6720; do cat /proc/$i/cmdline | awk '{gsub(/\x0/,"<NUL>"); print};'; d
sbin/zabbix_agentd_30<NUL>-c<NUL>/home/zabbix/ZBXNEXT-1078/zabbix_agentd.conf<NUL>
sbin/zabbix_agentd_30: collector [idle 1 sec]<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><N
sbin/zabbix_agentd_30: listener #2 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><N
sbin/zabbix_agentd_30: listener #3 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><N
sbin/zabbix_agentd_30: active checks #1 [idle 1 sec]<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.

Using cmdline parameter with a proper regular expression produces a correct result:

```
$ zabbix_get -s localhost -k 'proc.num[,zabbix,,zabbix_agentd_30[ :]]'
6
```

Be careful when using proc.mem[] and proc.num[] items for monitoring programs which modify their commandlines.

Before putting name and cmdline parameters into 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.mem[] and proc.num[] items

Let's take as an example one of kernel threads:

```
$ ps -ef| grep kthreadd
root         2     0  0 09:33 ?        00:00:00 [kthreadd]
```

It can be selected with process name parameter:

```
$ 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 kthreadd
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 kthreadd
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.

11 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.

Item net.tcp.service parameters

ftp

Creates a TCP connection and expects the first 4 characters of the response to be ”220 ”, then sends ”QUIT\r\n”. Default port 21 is used if not specified.

http

Creates a TCP connection without expecting and sending anything. Default port 80 is used if not specified.

https

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.

imap

Creates a TCP connection and expects the first 4 characters of the response to be ”* OK”, then sends ”a1 LOGOUT\r\n”. Default port 143 is used if not specified.

ldap

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.

nntp

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 multiline 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.

Item net.udp.service parameters

ntp

Sends an SNTP packet over UDP and validates the response according to RFC 4330, section 5. Default port 123 is used if not specified.

12 Unreachable/unavailable host settings

Overview

Several configuration parameters define how Zabbix server should behave when an agent check (Zabbix, SNMP, IPMI, JMX) fails and a host becomes unreachable.

Unreachable host

A host 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 host availability in any way.

From that moment **UnreachableDelay** defines how often a host 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
Zabbix agent item "system.cpu.load[percpu,avg15]" on host "New host" failed: another network error, wait f
```

Note that the exact item that failed is indicated and the item type (Zabbix agent).

Note:
The *Timeout* parameter will also affect how early a host 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 a host is rechecked more than once before a host becomes unavailable.

Switching host back to available

When unreachability period is over, the host is polled again, decreasing priority for item, that turned host into unreachable state. If the unreachable host reappears, the monitoring returns to normal automatically:

```
resuming Zabbix agent checks on host "New host": connection restored
```

Note:
Once host becomes available, it does not poll all its items immediately for two reasons:

- It might overload the host.
- The host restore time is not always matching planned item polling schedule time.

So, after the host becomes available, items are not polled immediately, but they are getting rescheduled to their next polling round.

Unavailable host

After the UnreachablePeriod ends and the host has not reappeared, the host is treated as unavailable.

In the server log it is indicated by messages like these:

```
temporarily disabling Zabbix agent checks on host "New host": host unavailable
```

and in the frontend the host availability icon for the respective interface goes from green (or gray) to red (note that on mouseover a tooltip with the error description is displayed):

The **UnavailableDelay** parameter defines how often a host is checked during host 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 host is restored, the monitoring returns to normal automatically, too:

```
enabling Zabbix agent checks on host "New host": host 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.

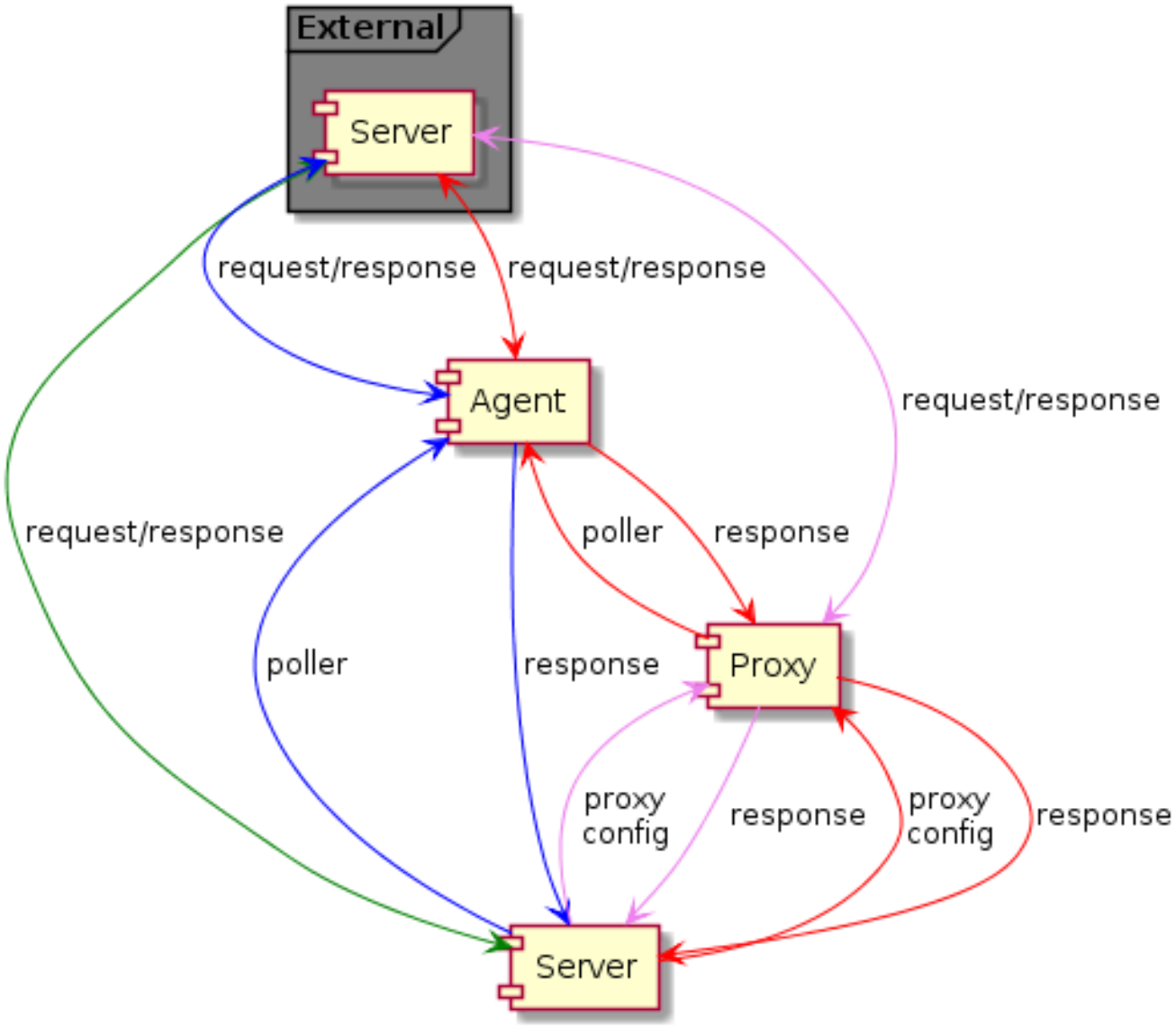
Items

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>]`)
- - Server → proxy → external Zabbix instance (`zabbix[stats,<ip>,<port>]`)
- - Server → agent → external Zabbix instance (`zabbix.stats[<ip>,<port>]`)
- - Server → proxy → agent → external Zabbix instance (`zabbix.stats[<ip>,<port>]`)

To make sure that the target instance allows querying it by the external instance, list the address of the external instance in the 'StatsAllowedIP' parameter on the target instance.

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:

- Template App Remote Zabbix server
- Template App 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:

```
{
  "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,
        "pused": 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:

```
apt install krb5-user
```

For RHEL/CentOS:

```
yum install krb5-workstation
```

Step 2

Configure Kerberos configuration file (see MIT documentation for details)

```
cat /etc/krb5.conf
[libdefaults]
    default_realm = EXAMPLE.COM

#### The following krb5.conf variables are only for MIT Kerberos.
    kdc_timesync = 1
    ccache_type = 4
    forwardable = true
    proxiable = true

[realms]
    EXAMPLE.COM = {
    }

[domain_realm]
    .example.com=EXAMPLE.COM
    example.com=EXAMPLE.COM
```

Step 3

Create a Kerberos ticket for user *zabbix*. Run the following command as user *zabbix*:

```
kinit zabbix
```

> **Attention:**
> 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.

6 Triggers

1 Supported trigger functions

All functions supported in trigger expressions are listed here.

FUNCTION	Description	Parameters	Comments
abschange	The amount of absolute difference between the previous and latest value.		Supported value types: float, int, str, text, log For example: (previous value;latest value=abschange) 1;5=4 3;1=2 0;-2.5=2.5 For strings returns: 0 - values are equal 1 - values differ
avg (sec|#num,<time_shift>)			

FUNCTION				
		Average value of an item within the defined evaluation period.	**sec** or **#num** - maximum evaluation period[1] in seconds or in latest collected values (preceded by a hash mark) **time_shift** (optional) - evaluation point is moved the number of seconds back in time	Supported value types: float, int Examples: => avg(#5) → average value for the five latest values => avg(1h) → average value for an hour => avg(1h,1d) → average value for an hour one day ago. The `time_shift` parameter is supported since Zabbix 1.8.2. It is useful when there is a need to compare the current average value with the average value time_shift seconds back.
band (<sec|#num>,mask,<time_shift>)				

FUNCTION				
		Value of "bitwise AND" of an item value and mask.	**sec** (ignored, equals #1) or **#num** (optional) - the Nth most recent value **mask** (mandatory) - 64-bit unsigned integer (0 - 18446744073709551615) **time_shift** (optional) - evaluation point is moved the number of seconds back in time	Supported value types: int Take note that `#num` works differently here than with many other functions (see last()). 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: => band(,12)=8 or band(,12)=4 → 3rd or 4th bit set, but not both at the same time => band(,20)=16 → 3rd bit not set and 5th bit set. This function is supported since Zabbix 2.2.0.

FUNCTION		
change	The amount of difference between the previous and latest value.	Supported value types: float, int, str, text, log For example: (previous value;latest value=change) 1;5=+4 3;1=-2 0;-2.5=-2.5 See also: abschange for comparison For strings returns: 0 - values are equal 1 - values differ
count (sec|#num,<pattern>,<operator>,<time_shift>)		

FUNCTION			
	Number of values within the defined evaluation period.	**sec** or **#num** - maximum evaluation period[1] in seconds or in latest collected values (preceded by a hash mark) **pattern** (optional) - required pattern **operator** (optional) Supported operators: *eq* - equal *ne* - not equal *gt* - greater *ge* - greater or equal *lt* - less *le* - less or equal *like* - matches if contains pattern (case-sensitive) *band* - bitwise AND *regexp* - case sensitive match of regular expression given in `pattern` *iregexp* - case insensitive match of regular expression given in `pattern` Note that: *eq*	Supported value types: float, integer, string, text, log Float items match with the precision of 2.22e-16 (since 5.0.13); before that or if database is not upgraded the precision is 0.000001. With *band* as third parameter, the second `pattern` parameter can be specified as two numbers, separated by '/': **number_to_compare_with/** 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 *num-*

FUNCTION			
date	Current date in YYYYMMDD format.		Supported value types: *any* Example of returned value: 20150731
dayofmonth	Day of month in range of 1 to 31.		Supported value types: *any* This function is supported since Zabbix 1.8.5.
dayofweek	Day of week in range of 1 to 7 (Mon - 1, Sun - 7).		Supported value types: *any*
delta (sec|#num,<time_shift>)	Difference between the maximum and minimum values within the defined evaluation period (’max()’ minus ’min()’).	**sec** or **#num** - maximum evaluation period[1] in seconds or in latest collected values specified (preceded by a hash mark) **time_shift** (optional) - evaluation point is moved the number of seconds back in time	Supported value types: float, int The `time_shift` parameter is supported since Zabbix 1.8.2.
diff			

FUNCTION			
		Checking if last and previous values differ.	Supported value types: float, int, str, text, log Returns: 1 - last and previous values differ 0 - otherwise

forecast (sec|#num,<time_shift>,time,<fit>,<mode>)

FUNCTION			
	Future value, max, min, delta or avg of the item.	**sec** or **#num** - maximum evaluation period[1] in seconds or in latest collected values specified (preceded by a hash mark) **time_shift** (optional) - evaluation point is moved the number of seconds back in time **time** - forecasting horizon in seconds **fit** (optional) - 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	Supported value types: float, int If value to return is larger than 1.7976931348623157E+ or less than -1.7976931348623157E+ return value is cropped to 1.7976931348623157E+ or -1.7976931348623157E+ correspondingly. Becomes not supported only if misused in expression (wrong item type, invalid parameters), otherwise returns -1 in case of errors. Examples: => forecast(#10,,1h) → forecast of item value after one hour based on last 10 values => forecast(1h,,30m) → forecast of item value after 30 minutes based on last hour data => forecast(1h,1d,12h) → forecast

FUNCTION

fuzzytime (sec)

FUNCTION				
		Checking how much the passive agent time differs from the Zabbix server/proxy time.	**sec** - seconds	Supported value types: float, int Returns: 1 - difference between the passive item value (as times-tamp) and Zabbix server/proxy times-tamp (clock of value collection) is less than or equal to T seconds 0 - otherwise Usually used with the 'sys-tem.localtime' item to check that local time is in sync with the local time of Zabbix server. *Note* that 'sys-tem.localtime' must be config-ured as a passive check. Can be used also with vfs.file.time[/path/file,mo key to check that file didn't get updates for long time. Example: => fuzzy-time(60)=0

FUNCTION			
iregexp (<pattern>,<sec|#num>)	This function is a non case-sensitive analog of regexp().	see regexp()	Supported value types: str, log, text
last (<sec|#num>,<time_shift>)			

FUNCTION				
	The most recent value.	**sec** (ignored, equals #1) or **#num** (optional) - the Nth most recent value **time_shift** (optional) - evaluation point is moved the number of seconds back in time	Supported value types: float, int, str, text, log Take note that `#num` 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 exact order of values if more than two values exist within one second in history. The `#num` parameter is supported since Zabbix 1.6.2. The `time_shift` parameter is supported since Zabbix 1.8.2.
logeventid (<pattern>)				

FUNCTION			
	Checking if event ID of the last log entry matches a regular expression.	**pattern** (optional) - regular expression describing the required pattern, Perl Compatible Regular Expression (PCRE) style.	Supported value types: log Returns: 0 - does not match 1 - matches This function is supported since Zabbix 1.8.5.
logseverity	Log severity of the last log entry.		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 (<pattern>)			

FUNCTION			
	Checking if log source of the last log entry matches a regular expression.	**pattern** (optional) - regular expression describing the required pattern, Perl Compatible Regular Expression (PCRE) style.	Supported value types: log Returns: 0 - does not match 1 - matches Normally used for Windows event logs. For example, logsource("VMware Server").
max (sec|#num,<time_shift>)	Highest value of an item within the defined evaluation period.	**sec** or **#num** - maximum evaluation period[1] in seconds or in latest collected values (preceded by a hash mark) **time_shift** (optional) - evaluation point is moved the number of seconds back in time	Supported value types: float, int The `time_shift` parameter is supported since Zabbix 1.8.2.
min (sec|#num,<time_shift>)			

FUNCTION				
		Lowest value of an item within the defined evaluation period.	**sec** or **#num** - maximum evaluation period[1] in seconds or in latest collected values (preceded by a hash mark) **time_shift** (optional) - evaluation point is moved the number of seconds back in time	Supported value types: float, int The `time_shift` parameter is supported since Zabbix 1.8.2.
nodata (sec,<mode>)				

FUNCTION				
		Checking for no data received.	**sec** - evaluation period in seconds. The period should not be less than 30 seconds because the history syncer process calculates this function only every 30 seconds. nodata(0) is disallowed. **mode** - if set to *strict*, this function will be insensitive to proxy availability (see comments for details).	Supported value types: *any* 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

FUNCTION			
now	Number of seconds since the Epoch (00:00:00 UTC, January 1, 1970).		Supported value types: *any*
percentile (sec|#num,<time_shift>,percentage)	P-th percentile of a period, where P (percentage) is specified by the third parameter.	**sec** or **#num** - maximum evaluation period[1] in seconds or in latest collected values (preceded by a hash mark) **time_shift** (optional) - evaluation point is moved the number of seconds back in time **percentage** - a floating-point number between 0 and 100 (inclusive) with up to 4 digits after the decimal point	Supported value types: float, int This function is supported since Zabbix 3.0.0.
prev	Previous value.		Supported value types: float, int, str, text, log Returns the same as last(#2).
regexp (<pattern>,<sec|#num>)			

FUNCTION			
	Checking if the latest (most recent) value matches regular expression.	**pattern** (optional) - regular expression, Perl Compatible Regular Expression (PCRE) style. **sec** or **#num** (optional) - maximum evaluation period[1] in seconds or in latest collected values (preceded by a hash mark). In this case, more than one value may be processed.	Supported value types: 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. This function is case-sensitive.

str (<pattern>,<sec|#num>)

FUNCTION			
	Finding a string in the latest (most recent) value.	**pattern** (optional) - required string **sec** or **#num** (optional) - maximum evaluation period[1] in seconds or in latest collected values (preceded by a hash mark). In this case, more than one value may be processed.	Supported value types: 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. This function is case-sensitive. *Tip*: You may use the 'count' function with the *like* operator to count string values that match a pattern.
strlen (<sec|#num>,<time_shift>)			

FUNCTION				
		Length of the latest (most recent) value in characters (not bytes).	**sec** (ignored, equals #1) or **#num** (optional) - the Nth most recent value **time_shift** (optional) - evaluation point is moved the number of seconds back in time	Supported value types: str, text, log Take note that `#num` works differently here than with many other functions. Examples: => strlen()(is equal to strlen(#1)) → length of the latest value => strlen(#3) → length of the third most recent value => strlen(,1d) → length of the most recent value one day ago. This function is supported since Zabbix 1.8.4.
sum (sec|#num,<time_shift>)				

FUNCTION			
	Sum of collected values within the defined evaluation period.	**sec** or **#num** - maximum evaluation period[1] in seconds or in latest collected values (preceded by a hash mark) **time_shift** (optional) - evaluation point is moved the number of seconds back in time	Supported value types: float, int The `time_shift` parameter is supported since Zabbix 1.8.2.
time	Current time in HHMMSS format.		Supported value types: *any* Example of returned value: 123055
timeleft (sec|#num,<time_shift>,threshold,<fit>)			

FUNCTION				
		Time in seconds needed for an item to reach a specified threshold.	**sec** or **#num** - maximum evaluation period[1] in seconds or in latest collected values (preceded by a hash mark) **time_shift** (optional) - evaluation point is moved the number of seconds back in time **threshold** - value to reach **fit** (optional) - see forecast()	Supported value types: float, int If value to return is larger than 1.7976931348623157E+ return value is cropped to 1.7976931348623157E+ Returns 1.7976931348623157E+ if threshold cannot be reached. Becomes not supported only if misused in expression (wrong item type, invalid parameters), otherwise returns -1 in case of errors. Examples: => timeleft(#10,,0) → time until item value reaches zero based on last 10 values => timeleft(1h,,100) → time until item value reaches 100 based on last hour data => timeleft(1h,1d,0) → time until item value

FUNCTION

Warning:
Important notes:
1) Some of the functions cannot be used for non-numeric values!
2) String arguments should be double quoted. Otherwise, they might get misinterpreted.
3) For all trigger functions **sec** and **time_shift** must be an integer with an optional time unit suffix and has absolutely nothing to do with the item's data type.

Footnotes

[1] The evaluation period is up to the latest collected value (not the server 'now' time). A function is evaluated as soon as at least one value is received (unless the `time_shift` parameter is used), i.e. Zabbix will not wait for all the values within the defined evaluation period before evaluating the function.

Functions and unsupported items

Since Zabbix 3.2, **nodata()**, **date()**, **dayofmonth()**, **dayofweek()**, **now()** and **time()** functions are calculated for unsupported items, too. Other functions require that the referenced item is in a supported state.

7 Macros

1 Supported macros

Overview

The table contains a complete list of macros supported by Zabbix out-of-the-box.

Note:
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*.

Macro	Supported in	Description
{ACTION.ID}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*Numeric ID of the triggered action.* Supported since 2.2.0.
{ACTION.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*Name of the triggered action.* Supported since 2.2.0.
{ALERT.MESSAGE}	→ Alert script parameters	*'Default message' value from action configuration.* Supported since 3.0.0.
{ALERT.SENDTO}	→ Alert script parameters	*'Send to' value from user media configuration.* Supported since 3.0.0.
{ALERT.SUBJECT}	→ Alert script parameters	*'Default subject' value from action configuration.* Supported since 3.0.0.
{DATE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*Current date in yyyy.mm.dd. format.*

Macro	Supported in	Description
{DISCOVERY.DEVICE.IPADDRESS}	→ Discovery notifications and commands	*IP address of the discovered device.* Available always, does not depend on host being added.
{DISCOVERY.DEVICE.DNS}	→ Discovery notifications and commands	*DNS name of the discovered device.* Available always, does not depend on host being added.
{DISCOVERY.DEVICE.STATUS}	→ Discovery notifications and commands	*Status of the discovered device*: can be either UP or DOWN.
{DISCOVERY.DEVICE.UPTIME}	→ Discovery notifications and commands	*Time since the last change of discovery status for a particular device*, with precision down to a second since Zabbix 5.0.4 (down to a minute before that). For example: 1h 29m 01s. For devices with status DOWN, this is the period of their downtime.
{DISCOVERY.RULE.NAME}	→ Discovery notifications and commands	*Name of the discovery rule that discovered the presence or absence of the device or service.*
{DISCOVERY.SERVICE.NAME}	→ Discovery notifications and commands	*Name of the service that was discovered.* For example: HTTP.
{DISCOVERY.SERVICE.PORT}	→ Discovery notifications and commands	*Port of the service that was discovered.* For example: 80.
{DISCOVERY.SERVICE.STATUS}	→ Discovery notifications and commands	*Status of the discovered service:// can be either UP or DOWN.* | |*{DISCOVERY.SERVICE.UPTIME}* |→ *Discovery notifications and commands* |Time since the last change of discovery status for a particular service*, with precision down to a second since Zabbix 5.0.4 (down to a minute before that). For example: 1h 29m 01s. For services with status DOWN, this is the period of their downtime.* | |*{ESC.HISTORY}* |→ *Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications* |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*).
{EVENT.ACK.STATUS}	→ Trigger-based notifications and commands → Problem update notifications and commands	*Acknowledgment status of the event (Yes/No).*
{EVENT.AGE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*Age of the event that triggered an action*, with precision down to a second since Zabbix 5.0.4 (down to a minute before that). Useful in escalated messages.
{EVENT.DATE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*Date of the event that triggered an action.*
{EVENT.DURATION}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications	*Duration of the event (time difference between problem and recovery events)*, with precision down to a second since Zabbix 5.0.4 (down to a minute before that). Useful in problem recovery messages. Supported since 5.0.0.

Macro	Supported in	Description
{EVENT.ID}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications → Trigger URLs	*Numeric ID of the event that triggered an action.*
{EVENT.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands	*Name of the problem event that triggered an action.* Supported since 4.0.0.
{EVENT.NSEVERITY}	→ Trigger-based notifications and commands → Problem update notifications and commands	*Numeric value of the event severity.* Possible values: 0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster. Supported since 4.0.0.
{EVENT.OBJECT}	→ Trigger-based notifications and commands → Problem update notifications and commands	*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. Supported since 4.4.0.
{EVENT.OPDATA}	→ Trigger-based notifications and commands → Problem update notifications and commands	*Operational data of the underlying trigger of a problem.* Supported since 4.4.0.
{EVENT.RECOVERY.DATE}	→ Problem recovery notifications and commands	*Date of the recovery event.* Supported since 2.2.0.
{EVENT.RECOVERY.ID}	→ Problem recovery notifications and commands	*Numeric ID of the recovery event.* Supported since 2.2.0.
{EVENT.RECOVERY.NAME}	→ Problem recovery notifications and commands	*Name of the recovery event.* Supported since 4.4.1.
{EVENT.RECOVERY.STATUS}	→ Problem recovery notifications and commands	*Verbal value of the recovery event.* Supported since 2.2.0.
{EVENT.RECOVERY.TAGS}	→ Problem recovery notifications and commands	*A comma separated list of recovery event tags.* Expanded to an empty string if no tags exist. Supported since 3.2.0.
{EVENT.RECOVERY.TAGSJSON}	→ Problem recovery notifications and commands	*A JSON array containing event tag objects.* Expanded to an empty array if no tags exist. Supported since 5.0.0.
{EVENT.RECOVERY.TIME}	→ Problem recovery notifications and commands	*Time of the recovery event.* Supported since 2.2.0.
{EVENT.RECOVERY.VALUE}	→ Problem recovery notifications and commands	*Numeric value of the recovery event.* Supported since 2.2.0.
{EVENT.SEVERITY}	→ Trigger-based notifications and commands → Problem update notifications and commands	*Name of the event severity.* Supported since 4.0.0.
{EVENT.SOURCE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*Numeric value of the event source.* Possible values: 0 - Trigger, 1 - Discovery, 2 - Autoregistration, 3 - Internal. Supported since 4.4.0.
{EVENT.STATUS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*Verbal value of the event that triggered an action.* Supported since 2.2.0.
{EVENT.TAGS}	→ Trigger-based notifications and commands → Problem update notifications and commands	*A comma separated list of event tags.* Expanded to an empty string if no tags exist. Supported since 3.2.0.
{EVENT.TAGSJSON}	→ Trigger-based notifications and commands → Problem update notifications and commands	*A JSON array containing event tag objects.* Expanded to an empty array if no tags exist. Supported since 5.0.0.

Macro	Supported in	Description
{EVENT.TAGS.<tag name>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Webhook media type URL names and URLs	*Event tag value referenced by the tag name.* A tag name containing non-alphanumeric characters (including non-English multibyte-UTF characters) should be double quoted. Quotes and backslashes inside a quoted tag name must be escaped with a backslash. Supported since 4.4.2.
{EVENT.TIME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*Time of the event that triggered an action.*
{EVENT.UPDATE.ACTION}	→ Problem update notifications and commands	*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.
{EVENT.UPDATE.DATE}	→ Problem update notifications and commands	*Date of problem update (acknowledgment, etc).* Deprecated name: {ACK.DATE}
{EVENT.UPDATE.HISTORY}	→ Trigger-based notifications and commands → Problem update notifications and commands	*Log of problem updates (acknowledgments, etc).* Deprecated name: {EVENT.ACK.HISTORY}
{EVENT.UPDATE.MESSAGE}	→ Problem update notifications and commands	*Problem update message.* Deprecated name: {ACK.MESSAGE}
{EVENT.UPDATE.STATUS}	→ Trigger-based notifications and commands → Problem update notifications and commands	*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.
{EVENT.UPDATE.TIME}	→ Problem update notifications and commands	*Time of problem update (acknowledgment, etc).* Deprecated name: {ACK.TIME}
{EVENT.VALUE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*Numeric value of the event that triggered an action (1 for problem, 0 for recovering).* Supported since 2.2.0.

Macro	Supported in	Description
{HOST.CONN<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Global scripts (including confirmation text) → Map element labels, map URL names and values → Item key parameters[1] → Host interface IP/DNS → Trapper item ”Allowed hosts” field → Database monitoring additional parameters → SSH and Telnet scripts → JMX item endpoint field → Web monitoring[4] → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget/screen element → Trigger names, operational data and descriptions → Trigger URLs → Tag 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.	*Host IP address or DNS name, depending on host settings*[2]. Supported in trigger names since 2.0.0.
{HOST.DESCRIPTION<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Map element labels	*Host description.* Supported since 2.4.0.
{HOST.DNS<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Global scripts (including confirmation text) → Map element labels, map URL names and values → Item key parameters[1] → Host interface IP/DNS → Trapper item ”Allowed hosts” field → Database monitoring additional parameters → SSH and Telnet scripts → JMX item endpoint field → Web monitoring[4] → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget/screen element → Trigger names, operational data and descriptions → Trigger URLs → Tag 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.	*Host DNS name*[2]. Supported in trigger names since 2.0.0.

Macro	Supported in	Description
{HOST.HOST<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Autoregistration notifications and commands → Internal notifications → Global scripts (including confirmation text) → Item key parameters → Map element labels, map URL names and values → Host interface IP/DNS → Trapper item "Allowed hosts" field → Database monitoring additional parameters → SSH and Telnet scripts → JMX item endpoint field → Web monitoring[4] → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget/screen element → Trigger names, operational data and descriptions → Trigger URLs → Tag 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.	*Host name.* {HOSTNAME<1-9>} is deprecated.
{HOST.ID<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Map element labels, map URL names and values → URL field of dynamic URL dashboard widget/screen element → Trigger URLs → Tag names and values	*Host ID.*
{HOST.IP<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Autoregistration notifications and commands → Internal notifications → Global scripts (including confirmation text) → Map element labels, map URL names and values → Item key parameters[1] → Host interface IP/DNS → Trapper item "Allowed hosts" field → Database monitoring additional parameters → SSH and Telnet scripts → JMX item endpoint field → Web monitoring[4] → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget/screen element → Trigger names, operational data and descriptions → Trigger URLs → Tag 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.	*Host IP address*[2]. Supported since 2.0.0. {IPADDRESS<1-9>} is deprecated.
{HOST.METADATA}	→ Autoregistration notifications and commands	*Host metadata.* Used only for active agent autoregistration. Supported since 2.2.0.

Macro	Supported in	Description
{HOST.NAME<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Global scripts (including confirmation text) → Map element labels, map URL names and values → Item key parameters → Host interface IP/DNS → Trapper item "Allowed hosts" field → Database monitoring additional parameters → SSH and Telnet scripts → Web monitoring[4] → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget/screen element → Trigger names, operational data and descriptions → Trigger URLs → Tag 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.	*Visible host name.* Supported since 2.0.0.
{HOST.PORT<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Autoregistration notifications and commands → Internal notifications → Trigger names, operational data and descriptions → Trigger URLs → JMX item endpoint field → Tag names and values	*Host (agent) port*[2]. Supported in autoregistration since 2.0.0. Supported in trigger names, trigger descriptions, internal and trigger-based notifications since 2.2.2.
{HOSTGROUP.ID}	→ Map element labels, map URL names and values	*Host group ID.*
{INVENTORY.ALIAS<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Alias field in host inventory.*
{INVENTORY.ASSET.TAG<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Asset tag field in host inventory.*
{INVENTORY.CHASSIS<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Chassis field in host inventory.*
{INVENTORY.CONTACT<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Contact field in host inventory.* {PROFILE.CONTACT<1-9>} is deprecated.
{INVENTORY.CONTRACT.NUMBER<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Contract number field in host inventory.*

Macro	Supported in	Description
{INVENTORY.DEPLOYMENT.STATUS<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Deployment status field in host inventory.*
{INVENTORY.HARDWARE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Hardware field in host inventory.* `{PROFILE.HARDWARE<1-9>}` is deprecated.
{INVENTORY.HARDWARE.FULL<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Hardware (Full details) field in host inventory.*
{INVENTORY.HOST.NETMASK<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Host subnet mask field in host inventory.*
{INVENTORY.HOST.NETWORKS<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Host networks field in host inventory.*
{INVENTORY.HOST.ROUTER<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Host router field in host inventory.*
{INVENTORY.HW.ARCH<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Hardware architecture field in host inventory.*
{INVENTORY.HW.DATE.DECOMM<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Date hardware decommissioned field in host inventory.*
{INVENTORY.HW.DATE.EXPIRY<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Date hardware maintenance expires field in host inventory.*
{INVENTORY.HW.DATE.INSTALL<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Date hardware installed field in host inventory.*
{INVENTORY.HW.DATE.PURCHASE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Date hardware purchased field in host inventory.*
{INVENTORY.INSTALLER.NAME<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Installer name field in host inventory.*

Macro	Supported in	Description
{INVENTORY.LOCATION<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Location field in host inventory.* {PROFILE.LOCATION<1-9>} is deprecated.
{INVENTORY.LOCATION.LAT<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Location latitude field in host inventory.*
{INVENTORY.LOCATION.LON<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Location longitude field in host inventory.*
{INVENTORY.MACADDRESSA<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*MAC address A field in host inventory.* {PROFILE.MACADDRESS<1-9>} is deprecated.
{INVENTORY.MACADDRESSB<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*MAC address B field in host inventory.*
{INVENTORY.MODEL<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Model field in host inventory.*
{INVENTORY.NAME<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Name field in host inventory.* {PROFILE.NAME<1-9>} is deprecated.
{INVENTORY.NOTES<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Notes field in host inventory.* {PROFILE.NOTES<1-9>} is deprecated.
{INVENTORY.OOB.IP<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*OOB IP address field in host inventory.*
{INVENTORY.OOB.NETMASK<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*OOB subnet mask field in host inventory.*
{INVENTORY.OOB.ROUTER<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*OOB router field in host inventory.*
{INVENTORY.OS<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*OS field in host inventory.* {PROFILE.OS<1-9>} is deprecated.

Macro	Supported in	Description
{INVENTORY.OS.FULL<1 9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*OS (Full details) field in host inventory.*
{INVENTORY.OS.SHORT<1 9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*OS (Short) field in host inventory.*
{INVENTORY.POC.PRIMARY.CELL<1 9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Primary POC cell field in host inventory.*
{INVENTORY.POC.PRIMARY.EMAIL<1 9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Primary POC email field in host inventory.*
{INVENTORY.POC.PRIMARY.NAME<1 9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Primary POC name field in host inventory.*
{INVENTORY.POC.PRIMARY.NOTES<1 9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Primary POC notes field in host inventory.*
{INVENTORY.POC.PRIMARY.PHONE.A<1 9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Primary POC phone A field in host inventory.*
{INVENTORY.POC.PRIMARY.PHONE.B<1 9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Primary POC phone B field in host inventory.*
{INVENTORY.POC.PRIMARY.SCREEN<1 9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Primary POC screen name field in host inventory.*
{INVENTORY.POC.SECONDARY.CELL<1 9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Secondary POC cell field in host inventory.*
{INVENTORY.POC.SECONDARY.EMAIL<1 9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Secondary POC email field in host inventory.*
{INVENTORY.POC.SECONDARY.NAME<1 9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Secondary POC name field in host inventory.*

Macro	Supported in	Description
{INVENTORY.POC.SECONDARY.NOTES<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Secondary POC notes field in host inventory.*
{INVENTORY.POC.SECONDARY.PHONE.A<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Secondary POC phone A field in host inventory.*
{INVENTORY.POC.SECONDARY.PHONE.B<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Secondary POC phone B field in host inventory.*
{INVENTORY.POC.SECONDARY.SCREEN<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Secondary POC screen name field in host inventory.*
{INVENTORY.SERIALNO.A<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Serial number A field in host inventory.* `{PROFILE.SERIALNO<1-9>}` is deprecated.
{INVENTORY.SERIALNO.B<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Serial number B field in host inventory.*
{INVENTORY.SITE.ADDRESS.A<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Site address A field in host inventory.*
{INVENTORY.SITE.ADDRESS.B<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Site address B field in host inventory.*
{INVENTORY.SITE.ADDRESS.C<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Site address C field in host inventory.*
{INVENTORY.SITE.CITY<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Site city field in host inventory.*
{INVENTORY.SITE.COUNTRY<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Site country field in host inventory.*
{INVENTORY.SITE.NOTES<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Site notes field in host inventory.*

Macro	Supported in	Description
{INVENTORY.SITE.RACK<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Site rack location field in host inventory.*
{INVENTORY.SITE.STATE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Site state/province field in host inventory.*
{INVENTORY.SITE.ZIP<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Site ZIP/postal field in host inventory.*
{INVENTORY.SOFTWARE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Software field in host inventory.* {PROFILE.SOFTWARE<1-9>} is deprecated.
{INVENTORY.SOFTWARE.APPA<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Software application A field in host inventory.*
{INVENTORY.SOFTWARE.APPB<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Software application B field in host inventory.*
{INVENTORY.SOFTWARE.APPC<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Software application C field in host inventory.*
{INVENTORY.SOFTWARE.APPD<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Software application D field in host inventory.*
{INVENTORY.SOFTWARE.APPE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Software application E field in host inventory.*
{INVENTORY.SOFTWARE.FULL<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Software (Full details) field in host inventory.*
{INVENTORY.TAG<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Tag field in host inventory.* {PROFILE.TAG<1-9>} is deprecated.
{INVENTORY.TYPE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Type field in host inventory.* {PROFILE.DEVICETYPE<1-9>} is deprecated.

Macro	Supported in	Description
{INVENTORY.TYPE.FULL<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Type (Full details) field in host inventory.*
{INVENTORY.URL.A<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*URL A field in host inventory.*
{INVENTORY.URL.B<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*URL B field in host inventory.*
{INVENTORY.URL.C<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*URL C field in host inventory.*
{INVENTORY.VENDOR<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values	*Vendor field in host inventory.*
{ITEM.DESCRIPTION<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications	*Description of the Nth item in the trigger expression that caused a notification.* Supported since 2.0.0.
{ITEM.ID<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → HTTP agent type item, item prototype and discovery rule fields: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	*Numeric ID of the Nth item in the trigger expression that caused a notification.* Supported since 1.8.12.
{ITEM.KEY<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → HTTP agent type item, item prototype and discovery rule fields: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	*Key of the Nth item in the trigger expression that caused a notification.* Supported since 2.0.0. `{TRIGGER.KEY}` is deprecated.
{ITEM.KEY.ORIG<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications	*Original key (with macros not expanded) of the Nth item in the trigger expression that caused a notification.* Supported since 2.0.6.

Macro	Supported in	Description
{ITEM.LASTVALUE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger names, operational data and descriptions → Tag names and values → Trigger URLs[7]	*The latest value of the Nth item in the trigger expression that caused a notification.* It will resolve to *UNKNOWN* in the frontend if the latest history value has been collected more than the *ZBX_HISTORY_PERIOD* time ago (defined in defines.inc.php). 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. Supported since 1.4.3. It is alias to `{{HOST.HOST}:{ITEM.KEY}.last()}`. The resolved value for text/log items is truncated to 20 characters by the frontend in the following locations: - Operational data; - Trigger description; - Trigger URLs. To resolve to a full value, you may use macro functions. No values are truncated by the server. Customizing the macro value is supported for this macro; starting with Zabbix 3.2.0.
{ITEM.LOG.AGE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands	*Age of the log item event*, with precision down to a second since Zabbix 5.0.4 (down to a minute before that).
{ITEM.LOG.DATE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands	*Date of the log item event.*
{ITEM.LOG.EVENTID<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands	*ID of the event in the event log.* For Windows event log monitoring only.
{ITEM.LOG.NSEVERITY<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands	*Numeric severity of the event in the event log.* For Windows event log monitoring only.
{ITEM.LOG.SEVERITY<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands	*Verbal severity of the event in the event log.* For Windows event log monitoring only.
{ITEM.LOG.SOURCE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands	*Source of the event in the event log.* For Windows event log monitoring only.
{ITEM.LOG.TIME<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands	*Time of the log item event.*
{ITEM.NAME<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications	*Name of the Nth item (with macros resolved) in the trigger expression that caused a notification.*
{ITEM.NAME.ORIG<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications	*Original name (i.e. without macros resolved) of the Nth item in the trigger expression that caused a notification.* Supported since 2.0.6.
{ITEM.STATE<1-9>}	→ Item-based internal notifications	*The latest state of the Nth item in the trigger expression that caused a notification.* Possible values: **Not supported** and **Normal**. Supported since 2.2.0.

Macro	Supported in	Description
{ITEM.VALUE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger names, operational data and descriptions → Tag names and values → Trigger URLs[7]	Resolved to either: 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. 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} In the first case it will resolve to *UNKNOWN* if the history value has already been deleted or has never been stored. In the second case, and in the frontend only, it will resolve to *UNKNOWN* if the latest history value has been collected more than the *ZBX_HISTORY_PERIOD* time ago (defined in defines.inc.php). The resolved value for text/log items is truncated to 20 characters by the frontend in the following locations: - Operational data; - Trigger description; - Trigger URLs. To resolve to a full value, you may use macro functions. No values are truncated by the server. Customizing the macro value is supported for this macro, starting with Zabbix 3.2.0.
{LLDRULE.DESCRIPTION}	→ LLD-rule based internal notifications	*Description of the low-level discovery rule which caused a notification.* Supported since 2.2.0.
{LLDRULE.ID}	→ LLD-rule based internal notifications	*Numeric ID of the low-level discovery rule which caused a notification.* Supported since 2.2.0.
{LLDRULE.KEY}	→ LLD-rule based internal notifications	*Key of the low-level discovery rule which caused a notification.* Supported since 2.2.0.
{LLDRULE.KEY.ORIG}	→ LLD-rule based internal notifications	*Original key (with macros not expanded) of the low-level discovery rule which caused a notification.* Supported since 2.2.0.
{LLDRULE.NAME}	→ LLD-rule based internal notifications	*Name of the low-level discovery rule (with macros resolved) that caused a notification.* Supported since 2.2.0.
{LLDRULE.NAME.ORIG}	→ LLD-rule based internal notifications	*Original name (i.e. without macros resolved) of the low-level discovery rule that caused a notification.* Supported since 2.2.0.
{LLDRULE.STATE}	→ LLD-rule based internal notifications	*The latest state of the low-level discovery rule.* Possible values: **Not supported** and **Normal**. Supported since 2.2.0.
{MAP.ID}	→ Map element labels, map URL names and values	*Network map ID.*
{MAP.NAME}	→ Map element labels, map URL names and values → Text field in map shapes	*Network map name.* Supported since 3.4.0.

Macro	Supported in	Description
{PROXY.DESCRIPTION<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*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. Supported since 2.4.0.
{PROXY.NAME<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*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. Supported since 1.8.4.
{TIME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*Current time in hh:mm:ss.*
{TRIGGER.DESCRIPTION}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications	*Trigger description.* Supported since 2.0.4. Starting with 2.2.0, all macros supported in a trigger description will be expanded if {TRIGGER.DESCRIPTION} is used in notification text. {TRIGGER.COMMENT} is deprecated.
{TRIGGER.EVENTS.ACK}	→ Trigger-based notifications and commands → Problem update notifications and commands → Map element labels	*Number of acknowledged events for a map element in maps, or for the trigger which generated current event in notifications.* Supported since 1.8.3.
{TRIGGER.EVENTS.PROBLEM.ACK}	→ Trigger-based notifications and commands → Problem update notifications and commands → Map element labels	*Number of acknowledged PROBLEM events for all triggers disregarding their state.* Supported since 1.8.3.
{TRIGGER.EVENTS.PROBLEM.UNACK}	→ Trigger-based notifications and commands → Problem update notifications and commands → Map element labels	*Number of unacknowledged PROBLEM events for all triggers disregarding their state.* Supported since 1.8.3.
{TRIGGER.EVENTS.UNACK}	→ Trigger-based notifications and commands → Problem update notifications and commands → Map element labels	*Number of unacknowledged events for a map element in maps, or for the trigger which generated current event in notifications.* Supported in map element labels since 1.8.3.
{TRIGGER.HOSTGROUP.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications	*A sorted (by SQL query), comma-space separated list of host groups in which the trigger is defined.* Supported since 2.0.6.
{TRIGGER.PROBLEM.EVENTS.ACK}	→ Map element labels	*Number of acknowledged PROBLEM events for triggers in PROBLEM state.* Supported since 1.8.3.
{TRIGGER.PROBLEM.EVENTS.UNACK}	→ Map element labels	*Number of unacknowledged PROBLEM events for triggers in PROBLEM state.* Supported since 1.8.3.
{TRIGGER.EXPRESSION}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications	*Trigger expression.* Supported since 1.8.12.

Macro	Supported in	Description
{TRIGGER.EXPRESSION.RECOVERY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications	*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.
{TRIGGER.ID}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Map element labels, map URL names and values → Trigger URLs → Trigger tag values	*Numeric trigger ID which triggered this action.* Supported in trigger URLs since Zabbix 1.8.8, in trigger tag values since 4.4.1.
{TRIGGER.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications	*Name of the trigger* (with macros resolved). Note that since 4.0.0 {EVENT.NAME} can be used in actions to display the triggered event/problem name with macros resolved.
{TRIGGER.NAME.ORIG}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications	*Original name of the trigger* (i.e. without macros resolved). Supported since 2.0.6.
{TRIGGER.NSEVERITY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications	*Numerical trigger severity.* Possible values: 0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster. Supported starting from Zabbix 1.6.2.
{TRIGGER.SEVERITY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications	*Trigger severity name.* Can be defined in *Administration → General → Trigger severities*.
{TRIGGER.STATE}	→ Trigger-based internal notifications	*The latest state of the trigger.* Possible values: **Unknown** and **Normal**. Supported since 2.2.0.
{TRIGGER.STATUS}	→ Trigger-based notifications and commands → Problem update notifications and commands	*Trigger value at the time of operation step execution.* Can be either PROBLEM or OK. {STATUS} is deprecated.
{TRIGGER.TEMPLATE.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications	*A sorted (by SQL query), comma-space separated list of templates in which the trigger is defined, or *UNKNOWN* if the trigger is defined in a host.* Supported since 2.0.6.
{TRIGGER.URL}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications	*Trigger URL.*
{TRIGGER.VALUE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger expressions	*Current trigger numeric value*: 0 - trigger is in OK state, 1 - trigger is in PROBLEM state.
{TRIGGERS.UNACK}	→ Map element labels	*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.
{TRIGGERS.PROBLEM.UNACK}	→ Map element labels	*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. Supported since 1.8.3.
{TRIGGERS.ACK}	→ Map element labels	*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. Supported since 1.8.3.
{TRIGGERS.PROBLEM.ACK}	→ Map element labels	*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. Supported since 1.8.3.

Macro	Supported in	Description
{USER.ALIAS}	→ Global scripts (including confirmation text)	*Alias (username) of the user* who started the script. Supported since 5.0.2.
{USER.FULLNAME}	→ Problem update notifications and commands → Global scripts (including confirmation text)	*Name, surname and alias 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
{USER.NAME}	→ Global scripts (including confirmation text)	*Name of the user* who started the script. Supported since 5.0.2.
{USER.SURNAME}	→ Global scripts (including confirmation text)	*Surname name of the user* who started the script. Supported since 5.0.2.
{host:key.func(param)}	→ Trigger-based notifications and commands → Problem update notifications and commands → Map element/shape labels [3] → Link labels in maps[3] → Graph names[5] → Trigger expressions[6]	*Simple macros, as used in building trigger expressions*. Supported for shape labels in maps since 3.4.2.
{$MACRO}	→ See: User macros supported by location	*User-definable macros.*
{#MACRO}	→ See: Low-level discovery macros	*Low-level discovery macros.* Supported since 2.0.0. Customizing the macro value is supported for this macro, starting with Zabbix 4.0.0.
$1...$9	→ Trigger names → User parameter commands	*Positional macros/references.*

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.

[2] In remote commands, 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.

[3] Only the **avg**, **last**, **max** and **min** functions, with seconds as parameter are supported in this macro in map labels.

[4] Supported since Zabbix 2.2.0, {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. Macros {HOST.*} in web scenario *Name* and web scenario step *Name* are not supported since Zabbix 5.0.6.

[5] Supported since Zabbix 2.2.0. Only the **avg**, **last**, **max** and **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:

```
* {Cisco switch:ifAlias[{#SNMPINDEX}].last()}
* {{HOST.HOST}:ifAlias[{#SNMPINDEX}].last()}
```

[6] While supported to build trigger expressions, simple macros may not be used inside each other.

[7] Supported since 4.0.0.

Indexed macros

The indexed macro syntax of {MACRO**<1-9>**} works only in the context of **trigger expressions**. It can be used to reference hosts 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).

Additionally the {HOST.HOST<1-9>} macro is also supported within the {host:key.func(param)} macro in **graph names**. For example, {{HOST.HOST2}:key.func()} in the graph name will refer to the host of the second item in the graph.

Warning:
Indexed macros will not resolve in any other context, except the two cases mentioned here. For other contexts, use macros **without** index (i. e.{HOST.HOST}, {HOST.IP}, etc) instead.

2 User macros supported by location

Overview

This section contains a list of locations, where user-definable macros are supported.

Note:
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 actions, user macros can be used in the following fields:

Location		Multiple macros/mix with text[1]
Trigger-based notifications and commands		yes
Trigger-based internal notifications		yes
Problem update notifications		yes
Time period condition		no
Operations		
	Default operation step duration	no
	Step duration	no

Hosts

In a host configuration, user macros can be used in the following fields:

Location		Multiple macros/mix with text[1]
Interface IP/DNS		DNS only
Interface port		no
SNMP v1, v2		
	SNMP community	yes
SNMP v3		
	Context name	yes
	Security name	yes
	Authentication passphrase	yes
	Privacy passphrase	yes
IPMI		
	Username	yes
	Password	yes
Tags[2]		
	Tag names	yes
	Tag values	yes

Items / item prototypes

In an item or an item prototype configuration, user macros can be used in the following fields:

Location	Multiple macros/mix with text[1]
Name (deprecated)	yes
Item key parameters	yes
Update interval	no
Custom intervals	no
History storage period	no

Location	Multiple macros/mix with text[1]
Trend storage period	no
Calculated/aggregate item	
Formula (expression constants and function parameters; item key parameters)	yes
Database monitor	
Username	yes
Password	yes
SQL query	yes
HTTP agent	
URL[3]	yes
Query fields	yes
Timeout	no
Request body	yes
Headers (names and values)	yes
Required status codes	yes
HTTP proxy	yes
HTTP authentication username	yes

Location	Multiple macros/mix with text[1]
HTTP authentication password	yes
SSL certificate file	yes
SSL key file	yes
SSL key password	yes
Allowed hosts	yes
JMX agent	
JMX endpoint	yes
SNMP agent	
SNMP OID	yes
SSH agent	
Username	yes
Public key file	yes
Private key file	yes
Password	yes
Script	yes
TELNET agent	
Username	yes
Password	yes
Script	yes
Zabbix trapper	
Allowed hosts	yes
Preprocessing	
Step parameters (including custom scripts)	yes

Low-level discovery

In a low-level discovery rule, user macros can be used in the following fields:

Location		Multiple macros/mix with text[1]
Name (deprecated)		yes
Key parameters		yes
Update interval		no
Custom interval		no
Keep lost resources period		no
SNMP agent		
	SNMP OID	yes
SSH agent		
	Username	yes
	Public key file	yes
	Private key file	yes
	Password	yes
	Script	yes
TELNET agent		
	Username	yes
	Password	yes
	Script	yes
Zabbix trapper		
	Allowed hosts	yes
Database monitor		
	Username	yes
	Password	yes
	SQL query	yes
JMX agent		
	JMX endpoint	yes
HTTP agent		
	URL[3]	yes
	Query fields	yes
	Timeout	no
	Request body	yes
	Headers (names and values)	yes
	Required status codes	yes
	HTTP authentication username	yes
	HTTP authentication password	yes
Filters		
	Regular expression	yes
Overrides		
	Filters: regular expression	yes
	Operations: update interval (for item prototypes)	no
	Operations: history storage period (for item prototypes)	no
	Operations: trend storage period (for item prototypes)	no

Network discovery

In a network discovery rule, user macros can be used in the following fields:

Location		Multiple macros/mix with text[1]
Update interval		no
SNMP v1, v2		
	SNMP community	yes
	SNMP OID	yes
SNMP v3		
	Context name	yes
	Security name	yes
	Authentication passphrase	yes
	Privacy passphrase	yes

Location		Multiple macros/mix with text[1]
	SNMP OID	yes

Proxies

In a proxy configuration, user macros can be used in the following field:

Location	Multiple macros/mix with text[1]
Interface port (for passive proxy)	no

Templates

In a template configuration, user macros can be used in the following fields:

Location		Multiple macros/mix with text[1]
Tags[2]		
	Tag names	yes
	Tag values	yes

Triggers

In a trigger configuration, user macros can be used in the following fields:

Location		Multiple macros/mix with text[1]
Name		yes
Operational data		yes
Expression (only in constants and function parameters; secret macros are not supported)		yes
Description		yes
URL[3]		yes
Tag for matching		yes
Tags[2]		
	Tag names	yes
	Tag values	yes

Web scenario

In a web scenario configuration, user macros can be used in the following fields:

Location	Multiple macros/mix with text[1]
Name	yes
Update interval	no
Agent	yes
HTTP proxy	yes
Variables (values only)	yes

Location		Multiple macros/mix with text[1]
Headers (names and values)		yes
Steps		
	Name	yes
	URL[3]	yes
	Variables (values only)	yes
	Headers (names and values)	yes
	Timeout	no
	Required string	yes
	Required status codes	no
Authentication		
	User	yes
	Password	yes
	SSL certificate	yes
	SSL key file	yes
	SSL key password	yes

Other locations

In addition to the locations listed here, user macros can be used in the following fields:

Location		Multiple macros/mix with text[1]
Global scripts (including confirmation text)		yes
Monitoring → Screens		
	URL[3] parameter in *URL* screen element	yes
Administration → Users → Media		
	When active	no
Administration → General → Working time		
	Working time	no
Administration → Media types → Message templates		
	Subject	yes
	Message	yes

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
- last parameter of aggregate checks
- 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)
- slide show configuration ('Default delay' field)
- 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* ('Refresh unsupported items' field)

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:

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>120
{host:system.uptime[].last()}<86400
{host:system.cpu.load.avg(600)}<10
{host:vm.memory.size[available].last()}<20971520
```

could be changed to:

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>2m
{host:system.uptime.last()}<1d
{host:system.cpu.load.avg(10m)}<10
{host:vm.memory.size[available].last()}<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:

Symbol	Description
d	Day of the week: 1 - Monday, 2 - Tuesday ,... , 7 - Sunday
hh	Hours: 00-24
mm	Minutes: 00-59

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.

Attention:
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. This is true starting from version 1.8.7, for everything, while Working time has always worked this way.

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

Attention:
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).

See also

- External checks
- User parameters
- system.run items
- Custom alert scripts
- Remote commands
- Global scripts

11 Version compatibility

Supported agents

Zabbix agents starting with version 1.4 are compatible with Zabbix 5.0.

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 new and improved items, improved performance and reduced memory usage, use the latest 5.0 agent.

Note that Zabbix agent newer than 5.0 cannot be used with Zabbix server 5.0.

Notes for Windows XP

On Windows XP/Server 2003, do not use agent templates that are newer than Zabbix 4.0.x. The newer templates use English performance counters, which are only supported since Windows Vista/Server 2008.

Supported Zabbix proxies

Zabbix 5.0.x server can only work with Zabbix 5.0.x proxies. Zabbix 5.0.x proxies can only work with Zabbix 5.0.x server.

> **Attention:**
> It is no longer possible to start the upgraded server and have older and 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. For more information, see the upgrade procedure.

Warnings about using incompatible Zabbix daemon versions are logged.

Supported XML files

XML files, exported with 1.8, 2.0, 2.2, 2.4, 3.0, 3.2, 3.4, 4.0, 4.2 and 4.4 are supported for import in Zabbix 5.0.

> **Attention:**
> In Zabbix 1.8 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.

12 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

- CR_CONN_HOST_ERROR
- CR_SERVER_GONE_ERROR
- CR_CONNECTION_ERROR
- CR_SERVER_LOST
- CR_UNKNOWN_HOST
- ER_SERVER_SHUTDOWN
- ER_ACCESS_DENIED_ERROR
- ER_ILLEGAL_GRANT_FOR_TABLE
- ER_TABLEACCESS_DENIED_ERROR
- ER_UNKNOWN_ERROR

13 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:

```
int zabbix_sender_send_values(const char *address, unsigned short port,const char *source, const zabbix_
char **result);‘{.c}
```

The following data structures are used by the Zabbix sender dynamic link library:

```
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;
```

14 Python library for Zabbix API

Overview

zabbix_utils is a Python library for:

- working with Zabbix API;

- acting like Zabbix sender;
- acting like Zabbix get.

It is supported for Zabbix 5.0, 6.0, 6.4 and later.

15 Issues with SELinux

Socket-based inter-process communication has been added since Zabbix 3.4. 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 are present while the process is running.

16 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:'dbconfig.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-proxied 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

```
# ..
location / {
# ..
proxy_cookie_path /zabbix /;
proxy_pass http://192.168.0.94/zabbix/;
# ..
```

Example configuration for Apache

```
# ..
ProxyPass "/" http://host/zabbix/
ProxyPassReverse "/" http://host/zabbix/
ProxyPassReverseCookiePath /zabbix /
ProxyPassReverseCookieDomain host zabbix.example.com
# ..
```

17 Agent vs agent 2 comparison

This section describes the differences between the Zabbix agent and the Zabbix agent 2.

Parameter	Zabbix agent	Zabbix agent 2
Programming language	C	Go with some parts in C
Daemonization	yes	no (yes on Windows since 5.0.4)
Supported extensions	Custom loadable modules in C.	Custom plugins in Go.
Requirements		
Supported platforms	Linux, IBM AIX, FreeBSD, NetBSD, OpenBSD, HP-UX, Mac OS X, Solaris: 9, 10, 11, Windows: all desktop and server versions since XP	Linux, Windows, on which an up-to-date supported Go version can be installed.
Supported crypto libraries	GnuTLS 3.1.18 and newer OpenSSL 1.0.1, 1.0.2, 1.1.0, 1.1.1, 3.0.x. Note that 3.0.x is supported since Zabbix 5.0.28. LibreSSL - tested with versions 2.7.4, 2.8.2 (certain limitations apply, see the Encryption page for details).	Linux: OpenSSL 1.0.1 and later, 3.0.x. Note that 3.0.x is supported since Zabbix 5.0.28. MS Windows: OpenSSL 1.1.1 or later, 3.0.x. Note that 3.0.x is supported since Zabbix 5.0.28. The OpenSSL library must have PSK support enabled. LibreSSL is not supported.
Monitoring processes		
Processes	A separate active check process for each server/proxy record.	Single process with automatically created threads. The maximum number of threads is determined by the GOMAXPROCS environment variable.
Metrics	**UNIX**: see a list of supported items. **Windows**: see a list of additional Windows-specific items.	**UNIX**: All metrics supported by Zabbix agent. Additionally, the agent 2 provides Zabbix-native monitoring solution for: Docker, Memcached, MySQL, PostgreSQL, Redis, systemd (see a list of agent 2 specific items). **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.
Concurrency	Active checks for single server are executed sequentially.	Checks from different plugins or multiple checks within one plugin can be executed concurrently.
Scheduled/flexible intervals	Supported for passive checks only.	Supported for passive and active checks.
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.
Drop user privileges	yes (Unix-like systems only)	no
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
Index Return to Main Contents

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, **--config** *config-file*
Use the alternate *config-file* instead of the default one.

-R, **--runtime-control** *runtime-option*
Perform administrative functions according to *runtime-option*.

Runtime control options: **log_level_increase**
Increase log level

log_level_decrease
Decrease log level

help
List available runtime control options

metrics
List available metrics

version
Display version

-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_agent2.conf
Default location of Zabbix agent 2 configuration file (if not modified during compile time).

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agentd(8), **zabbix_get**(8), **zabbix_js**(8), **zabbix_proxy**(8), **zabbix_sender**(8), **zabbix_server**(8)

AUTHOR

Zabbix LLC

Index

This document was created on: 14:07:57 GMT, October 10, 2019

zabbix_agentd

Section: Maintenance Commands (8)
Updated: 2019-01-29
Index Return to Main Contents

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

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).

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agent2(8), **zabbix_get**(1), **zabbix_js**(1), **zabbix_proxy**(8), **zabbix_sender**(1), **zabbix_server**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

This document was created on: 20:50:13 GMT, March 18, 2020

zabbix_get

Section: User Commands (1)
Updated: 2020-02-29
Index Return to Main Contents

NAME

zabbix_get - Zabbix get utility

SYNOPSIS

zabbix_get -s *host-name-or-IP* [**-p** *port-number*] [**-I** *IP-address*] **-k** *item-key*
zabbix_get -s *host-name-or-IP* [**-p** *port-number*] [**-I** *IP-address*] **--tls-connect cert --tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-agent-cert-issuer** *cert-issuer*] [**--tls-agent-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] **-k** *item-key*
zabbix_get -s *host-name-or-IP* [**-p** *port-number*] [**-I** *IP-address*] **--tls-connect psk --tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] **-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.

-I, **--source-address** *IP-address*
Specify source IP address.

-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.

EXAMPLES

zabbix_get -s 127.0.0.1 -p 10050 -k ”system.cpu.load[all,avg1]”
zabbix_get -s 127.0.0.1 -p 10050 -k ”system.cpu.load[all,avg1]” --tls-connect cert --tls-ca-file /home/zabbix/zabbix_ca_file --tls-agent-cert-issuer ”CN=Signing CA,OU=IT operations,O=Example Corp,DC=example,DC=com” --tls-agent-cert-subject ”CN=server1,OU=IT operations,O=Example Corp,DC=example,DC=com” --tls-cert-file /home/zabbix/zabbix_get.crt --tls-key-file /home/zabbix/zabbix_get.key
zabbix_get -s 127.0.0.1 -p 10050 -k ”system.cpu.load[all,avg1]” --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_agent2(8), **zabbix_agentd**(8), **zabbix_js**(1), **zabbix_proxy**(8), **zabbix_sender**(1), **zabbix_server**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

This document was created on: 20:50:27 GMT, March 18, 2020

zabbix_js

Section: User Commands (1)
Updated: 2019-01-29
Index Return to Main Contents

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. Valid range: 1-60 seconds (default: 10)

-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

This document was created on: 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.

FILES

/usr/local/etc/zabbix_proxy.conf
Default location of Zabbix proxy configuration file (if not modified during compile time).

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agentd(8), **zabbix_get**(1), **zabbix_sender**(1), **zabbix_server**(8), **zabbix_js**(1), **zabbix_agent2**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

This document was created on: 15:49:29 GMT, September 04, 2020

zabbix_sender

Section: User Commands (1)
Updated: 2020-02-29
Index Return to Main Contents

NAME

zabbix_sender - Zabbix sender utility

SYNOPSIS

zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] **-s** *host* **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] [**-T**] [**-N**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] [**-T**] [**-N**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] **-s** *host* **--tls-connect cert** **--tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-server-cert-issuer** *cert-issuer*] [**--tls-server-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **--tls-connect cert** **--tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-server-cert-issuer** *cert-issuer*] [**--tls-server-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] [**-T**] [**-N**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **--tls-connect cert** **--tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-server-cert-issuer** *cert-issuer*] [**--tls-server-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **--tls-connect cert** **--tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-server-cert-issuer** *cert-issuer*] [**--tls-server-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] [**-T**] [**-N**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] **-s** *host* **--tls-connect psk** **--tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **--tls-connect psk** **--tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] [**-T**] [**-N**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **--tls-connect psk** **--tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-s** *host*] **--tls-connect psk** **--tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] [**-T**] [**-N**] [**-r**] **-i** *input-file*
zabbix_sender -h
zabbix_sender -V

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**, **TLSCRLFile**, **TLSServerCertIssuer**, **TLSServerCertSubject**, **TLSCertFile**, **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.

-I, **--source-address** *IP-address*
Specify source IP address. When used together with **--config**, overrides **SourceIP** parameter specified in agentd configuration file.

-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. Size limit for sending values from an input file depends on the size described in Zabbix communication protocol. 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.

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 certificate.

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

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).

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agent2(8) **zabbix_agentd**(8), **zabbix_get**(1), **zabbix_js**(1), **zabbix_proxy**(8), **zabbix_server**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

This document was created on: 20:49:51 GMT, March 18, 2020

zabbix_server

Section: Maintenance Commands (8)
Updated: 2020-09-04
Index Return to Main Contents

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*.

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

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:

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.

diaginfo[=*target*]
Gather diagnostic information in the server log file. Available since Zabbix 5.0.4.

snmp_cache_reload
Reload SNMP cache, clear the SNMP properties (engine time, engine boots, engine id, credentials) for all hosts.

housekeeper_execute
Start the housekeeping procedure. Ignored if the housekeeping procedure is currently in progress.

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”

FILES

/usr/local/etc/zabbix_server.conf
Default location of Zabbix server 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_js**(1), **zabbix_agent2**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

This document was created on: 15:49:23 GMT, September 04, 2020