Documentation 2.4

ZABBIX

01.07.2025

Contents

Zabbix Manual **4**

Copyright notice . . . 4

1. Introduction . . . 4

1 Manual structure . . . 4

2 What is Zabbix . . . 5

3 Zabbix features . . . 5

4 Zabbix overview . . . 6

5 What's new in Zabbix 2.4.0 . . . 7

6 What's new in Zabbix 2.4.1 . . . 23

7 What's new in Zabbix 2.4.2 . . . 23

8 What's new in Zabbix 2.4.3 . . . 24

9 What's new in Zabbix 2.4.4 . . . 24

10 What's new in Zabbix 2.4.5 . . . 25

11 What's new in Zabbix 2.4.6 . . . 25

12 What's new in Zabbix 2.4.7 . . . 26

13 What's new in Zabbix 2.4.8 . . . 26

2. Zabbix concepts . . . 27

1 Zabbix definitions . . . 27

2 Server . . . 28

3 Agent . . . 30

4 Proxy . . . 33

5 Java gateway . . . 35

6 Sender . . . 37

7 Get . . . 38

3. Installation . . . 38

1 Getting Zabbix . . . 38

2 Requirements . . . 39

3 Installation from packages . . . 43

4 Installation from sources . . . 45

5 Upgrade procedure . . . 53

6 Known issues . . . 54

7 Template changes . . . 55

8 Upgrade notes for 2.4.0 . . . 55

9 Upgrade notes for 2.4.1 . . . 57

10 Upgrade notes for 2.4.2 . . . 57

11 Upgrade notes for 2.4.3 . . . 58

12 Upgrade notes for 2.4.4 . . . 58

13 Upgrade notes for 2.4.5 . . . 59

14 Upgrade notes for 2.4.6 . . . 59

15 Upgrade notes for 2.4.7 . . . 59

16 Upgrade notes for 2.4.8 . . . 59

4. Quickstart . . . 60

1 Login and configuring user . . . 60

2 New host . . . 63

3 New item . . . 64

4 New trigger . . . 66

5 Receiving problem notification . . . 68

6 New template . . . 70

5. Zabbix appliance . . . 72

6. Configuration . . . 78

1 Hosts and host groups . . . 83
2 Items . . . 89
3 Triggers . . . 207
4 Events . . . 215
5 Visualisation . . . 216
6 Templates . . . 246
7 Notifications upon events . . . 247
8 Macros . . . 276
9 Users and user groups . . . 278
7. IT services . . . 282
8. Web monitoring . . . 284
1 Web monitoring items . . . 293
2 Real life scenario . . . 294
9. Virtual machine monitoring . . . 301
1 Virtual machine discovery key fields . . . 304
10. Maintenance . . . 305
11. Regular expressions . . . 308
12. Event acknowledgment . . . 310
13. Configuration export/import . . . 311
Groups . . . 312
Hosts . . . 312
14. Discovery . . . 319
1 Network discovery . . . 319
2 Active agent auto-registration . . . 325
3 Low-level discovery . . . 327
15. Distributed monitoring . . . 343
1 Proxies . . . 344
16. Web interface . . . 346
1 Frontend sections . . . 346
2 User profile . . . 405
3 Global search . . . 409
4 Frontend maintenance mode . . . 411
5 Page parameters . . . 412
6 Definitions . . . 412
7 Creating your own theme . . . 413
8 Debug mode . . . 414
17. API . . . 414
Method reference . . . 419
Appendix 1. Reference commentary . . . 775
Appendix 2. Changes from 2.2 to 2.4 . . . 777
Zabbix API changes in 2.4 . . . 780
18. Appendixes . . . 782
1 Frequently asked questions / Troubleshooting . . . 782
2 Installation . . . 783
3 Daemon configuration . . . 786
4 Items . . . 814
5 Triggers . . . 827
6 Macros . . . 843
7 Setting time periods . . . 896
8 Command execution . . . 896
9 Recipes for monitoring . . . 897
10 Performance tuning . . . 898
11 Version compatibility . . . 901
12 Database error handling . . . 901
13 Zabbix sender dynamic link library for Windows . . . 901

Zabbix manpages **902**

zabbix_agentd . . . 902
NAME . . . 902
SYNOPSIS . . . 902
DESCRIPTION . . . 902
FILES . . . 903
SEE ALSO . . . 903

AUTHOR . . . 903
Index . . . 903
zabbix_get . . . 904
NAME . . . 904
SYNOPSIS . . . 904
DESCRIPTION . . . 904
EXAMPLES . . . 904
SEE ALSO . . . 904
AUTHOR . . . 904
Index . . . 905
zabbix_proxy . . . 905
NAME . . . 905
SYNOPSIS . . . 905
DESCRIPTION . . . 905
FILES . . . 906
SEE ALSO . . . 906
AUTHOR . . . 906
Index . . . 906
zabbix_sender . . . 906
NAME . . . 906
SYNOPSIS . . . 907
DESCRIPTION . . . 907
EXIT STATUS . . . 908
EXAMPLES . . . 908
SEE ALSO . . . 908
AUTHOR . . . 908
Index . . . 908
zabbix_server . . . 909
NAME . . . 909
SYNOPSIS . . . 909
DESCRIPTION . . . 909
FILES . . . 910
SEE ALSO . . . 910
AUTHOR . . . 910
Index . . . 910

Zabbix 2.4 API **910**

Zabbix Manual

Welcome to the user manual for Zabbix 2.4 software. These pages are created to help users successfully manage their monitoring tasks with Zabbix, from the simple to the more complex.

Copyright notice

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

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

1. Introduction

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

1 Manual structure

Structure

The content of this Zabbix 2.4 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.

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 software that monitors numerous parameters of a network and the health and integrity of servers. 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 visualisation 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 organisations 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.

Learn more about Zabbix features.

Users of Zabbix

Many organisations 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 visualisation 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 auto registration
- discovery of file systems, network interfaces and SNMP OIDs

Fast web interface

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

Zabbix API

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

Permissions system

- secure user authentication
- certain users can be limited to certain views

Full featured and easily extensible agent

- deployed on monitoring targets
- can be deployed on both Linux and Windows

Binary daemons

- written in C, for performance and small memory footprint
- easily portable

Ready for complex environments

- remote monitoring made easy by using a Zabbix proxy

4 Zabbix overview

Architecture

Zabbix consists of several major software components, the responsibilities of which 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.

Note:
Zabbix web interface must run on the same physical machine if SQLite is used.

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

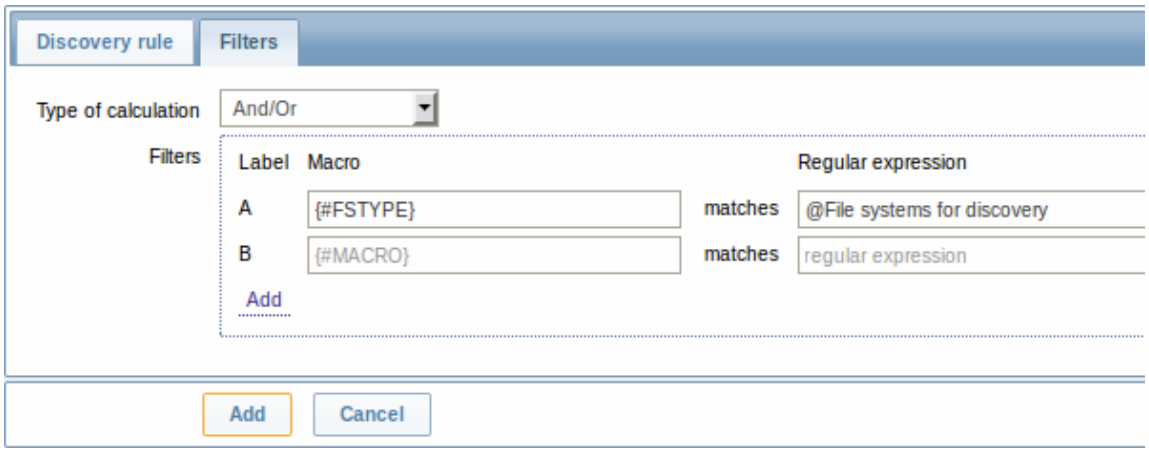
Zabbix agents are deployed on monitoring targets to actively monitor local resources and applications and report the gathered data to Zabbix server.

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 it 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 2.4.0

5.1 Defining multiple filters for low level discovery The filter section in low level discovery rule definition has been split out into a separate tab and, most importantly, allows to define several filters as well as to define the calculation rules between the filters.



For more information, see low level discovery documentation.

5.2 Node-based distributed monitoring removed Previously Zabbix supported two options for distributed monitoring - using nodes and proxies. In Zabbix 2.4.0 the support of node-based distributed monitoring has been discontinued due to existing limitations and potential problems with reliability in large environments.

While a new version of distributed monitoring may be considered and developed in the future, for the time being using proxies becomes the main distributed monitoring option in Zabbix. Furthermore, the proxy solution has proven itself over time as reliable, while also being easier in configuration and maintenance. Many users who initially had node-based distributed monitoring have switched to proxy-based setups.

For the remaining node-based DM users, during an upgrade to Zabbix 2.4.0, each upgraded node will be switched to a standalone Zabbix server keeping all configuration and history information from the local node and from the child nodes as well.

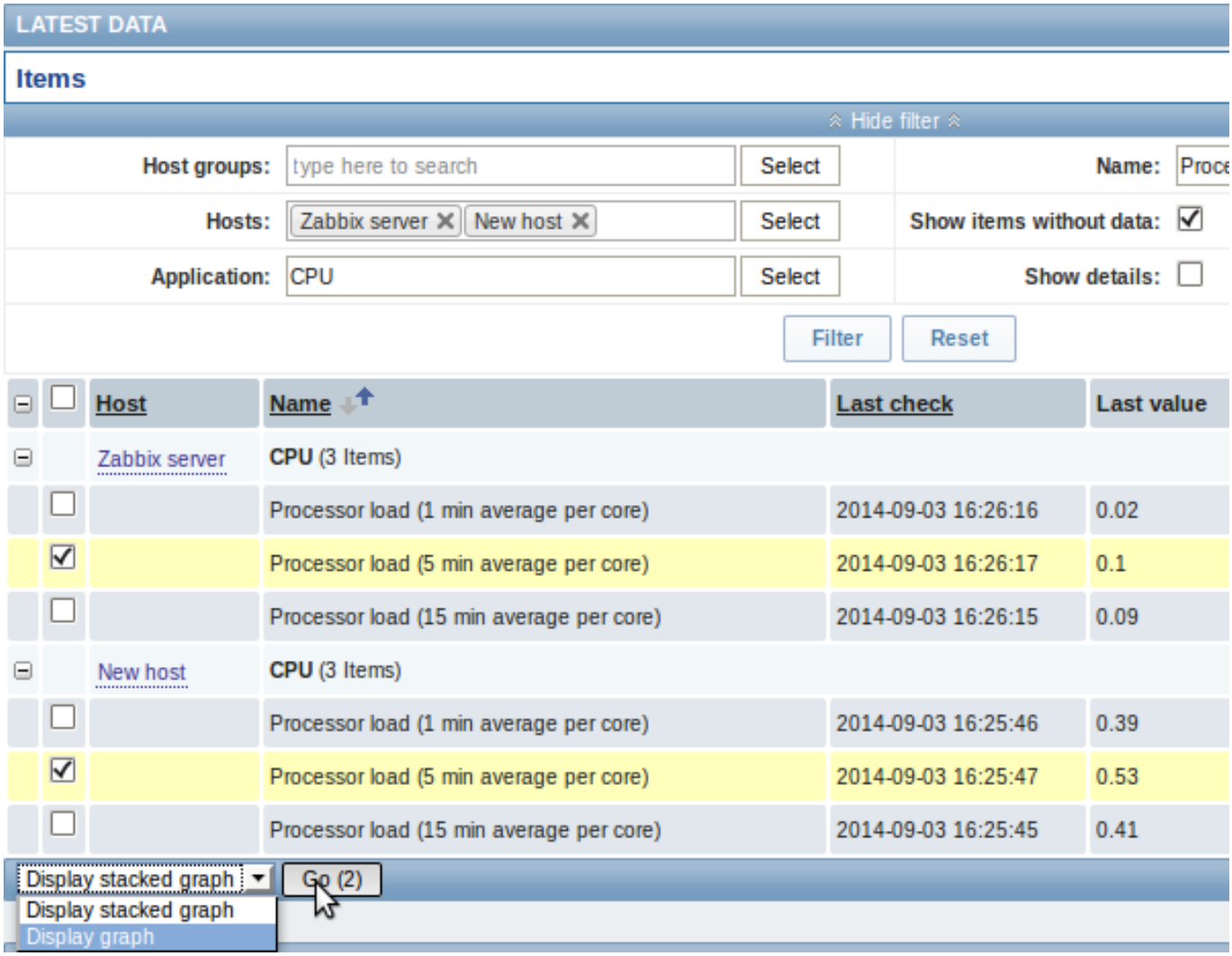
To maintain uniqueness of data from non-local nodes, all the fields will be prefixed with N<nodeid>_. If the length of the new value exceeds max field size, it will be trimmed. Highly unlikely, but if the new value already exists in the database, the situation will be caught by a check on duplicates. Global macros will be processed in a special way by adding N<nodeid>_ after the dollar sign, for example, {$N123_MACRO}.

5.3 Ad-hoc graphs for several items New in Zabbix is the ability to create instant ad-hoc graphs for several items. If previously it was possible to access a simple graph for one item at any time in the latest data page, creating custom graphs with several items took a little more effort.

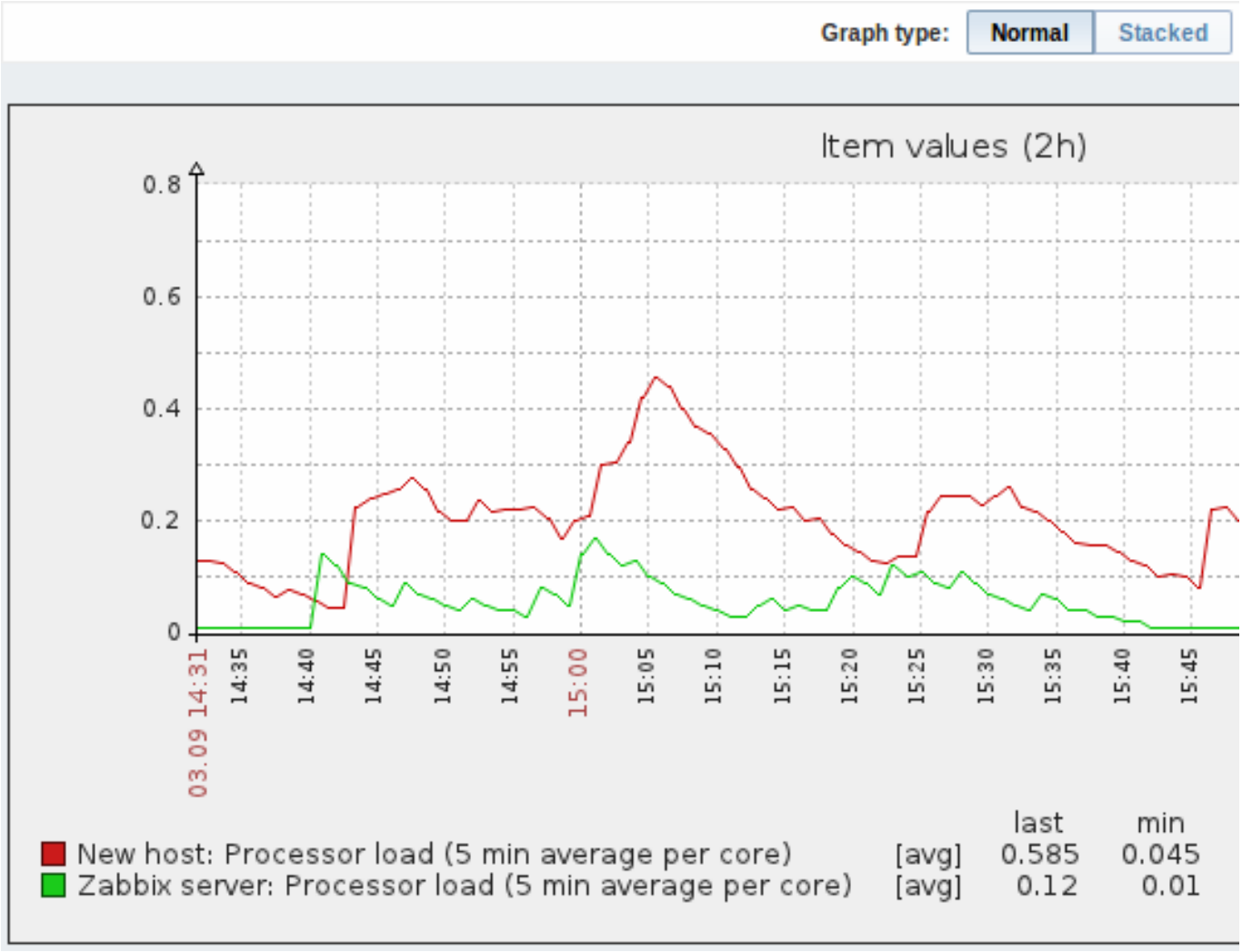
While offering quite a bit of flexibility, creating custom graphs was not particularly quick or easy to maintain, especially, if you wanted to compare a handful of items where their respective hosts were created and then deleted after a while. To address this issue it has now been made possible in Zabbix to create ad-hoc graphs for several items in a very quick way.

For that, similarly as for accessing simple graphs, you go to *Monitoring* → *Latest data*.

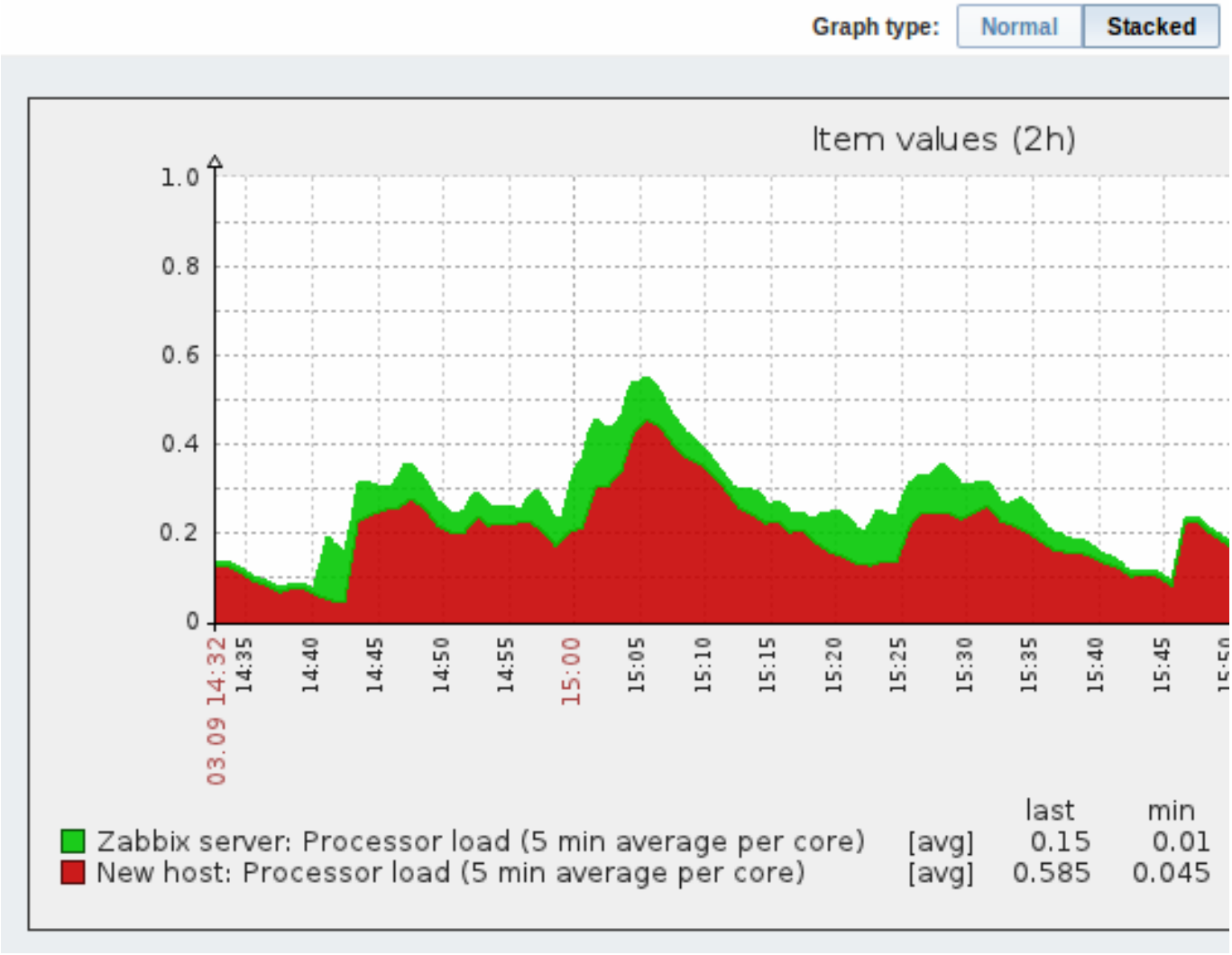
The section no longer has the host and host group selection dropdowns. Instead, those two choices can be made within an expanded filter section, which can be used flexibly for displaying the exact set of items you want. Additonally, items now have checkboxes in the listing.



To get a quick graph at any time, select the desired items, choose *Display stacked graph* or *Display graph* in the dropdown below, click on *Go* and have the respective graph created instantly.



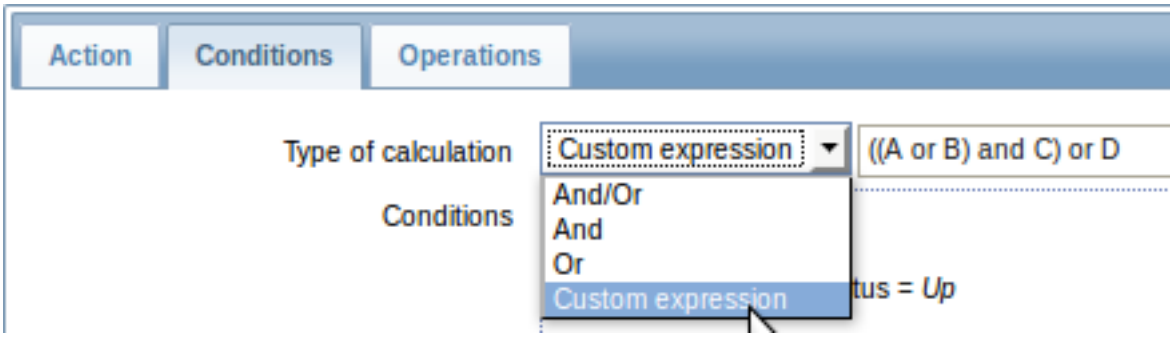
Note that in the created graph window you have the time period selector available and the possibility to switch from the ”normal” line graph to a stacked one (and back).



5.4 Custom calculation of action conditions While the previous ways of calculating action conditions (And, Or, And/Or) offered some flexibility, there were clear limitations as well. In a simple case of using And/Or, if you had two groups of the same condition type, you could not use `AND` within one group and `OR` within the other.

To lift such limitations, action conditions now can be calculated using a completely **custom** formula, such as

(A and B) and (C or D)
(A and B) or (C and D)
((A or B) and C) or D
etc.



The formula must include all conditions (represented as uppercase letters A, B, C, ...) and may include spaces, tabs, brackets ( ), **and** (case sensitive), **or** (case sensitive).

The custom calculation option is available for action conditions as well as for multiple filters in low-level discovery where it was first introduced.

5.5 Support of new trigger operators The list of operators supported in trigger expressions has been extended with three new ones:

- **>=** - greater than or equal
- **<=** - less than or equal
- **not** - logical NOT

In additional development, the syntax of three existing operators has been changed with the aim of making trigger expressions easier to read. Thus:

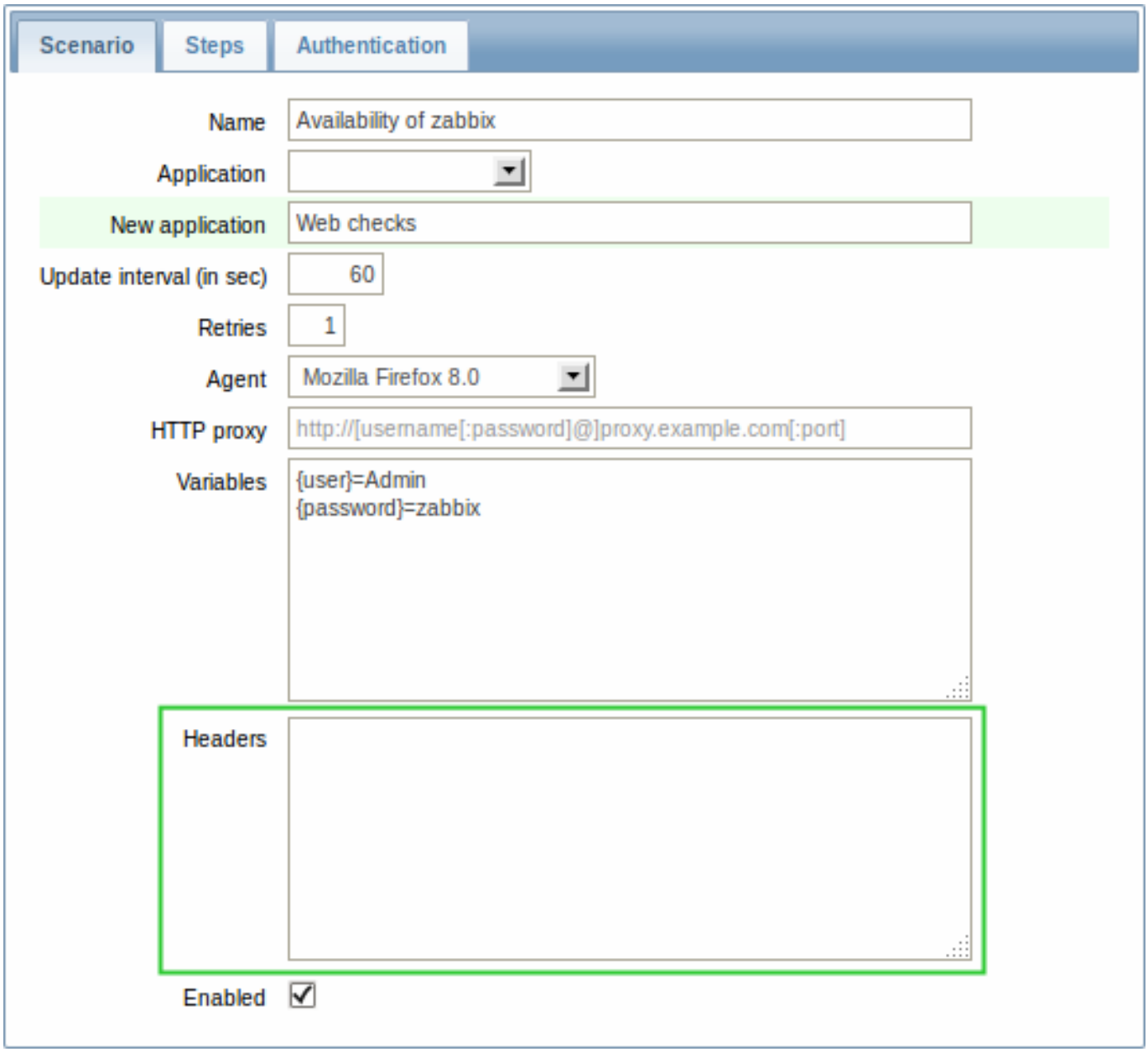
- # (not equal) is now expressed with **<>**
- & (logical AND) is now expressed with **and**
- | (logical OR) is now expressed with **or**

Note that the old operators are no longer supported, so the new ones have to be used instead. Note also that the new **and**, **or**, **not** operators are case-sensitive and must be surrounded by spaces or parentheses. For more details, refer to the trigger operator section.

In a related development, the support for newlines and tabs in trigger expressions has been added.

5.6 Web monitoring improvements Ability to specify custom headers

It is now possible to specify custom headers for HTTP requests in web monitoring:

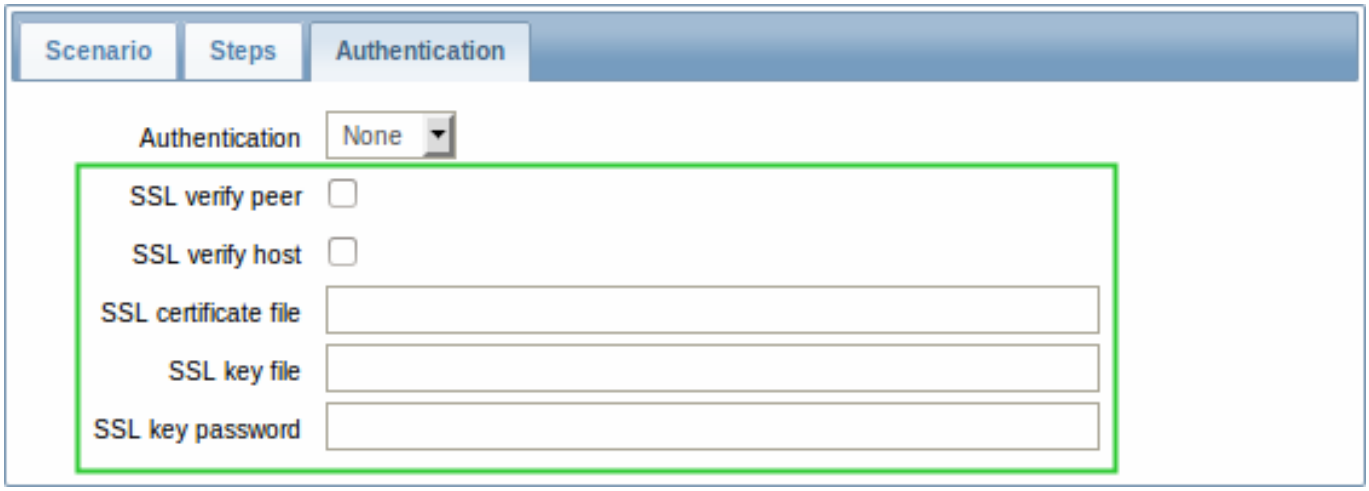


Custom headers are available on both the web scenario and scenario step levels. You can also request a page header only and optionally disable follow redirect functionality on the scenario step level.

In a related development, the maximum URL size limit has been increased from 255 to 2048 characters.

SSL verification and authentication options

A new *Authentication* tab has been added to the web scenario configuration form holding the already familiar *Authentication* dropdown and several new fields all related to SSL options:



Two of these options help authenticate the server to the client:

- SSL verify peer - will check that the certificate provided by the server is valid (trusted by a known certificate authority, not revoked, expired, etc.)
- SSL verify host - will check that the server name actually matches the name in the certificate

With the new SSLCALocation server parameter Zabbix also provides an option of specifying a separate directory for the certificates, which, if set, will override the system-wide directory.

Using SSL certificates is also a way of authenticating the client to the server. For this purpose three new options are available:

- SSL certificate file - you may specify the client certificate that is sent to the web server when establishing an SSL connection. Unless it contains the client private key the client private key file must be specified separately in the following field
- SSL key file - specify the client private key
- SSL key password - specify the client private key password if the key is encrypted

The new SSLCertLocation and SSLKeyLocation server parameters determine the path to client certificate and private key files.

Clearing history and trend data

A *Clear history and trends* button has been added to the web scenario configuration form.

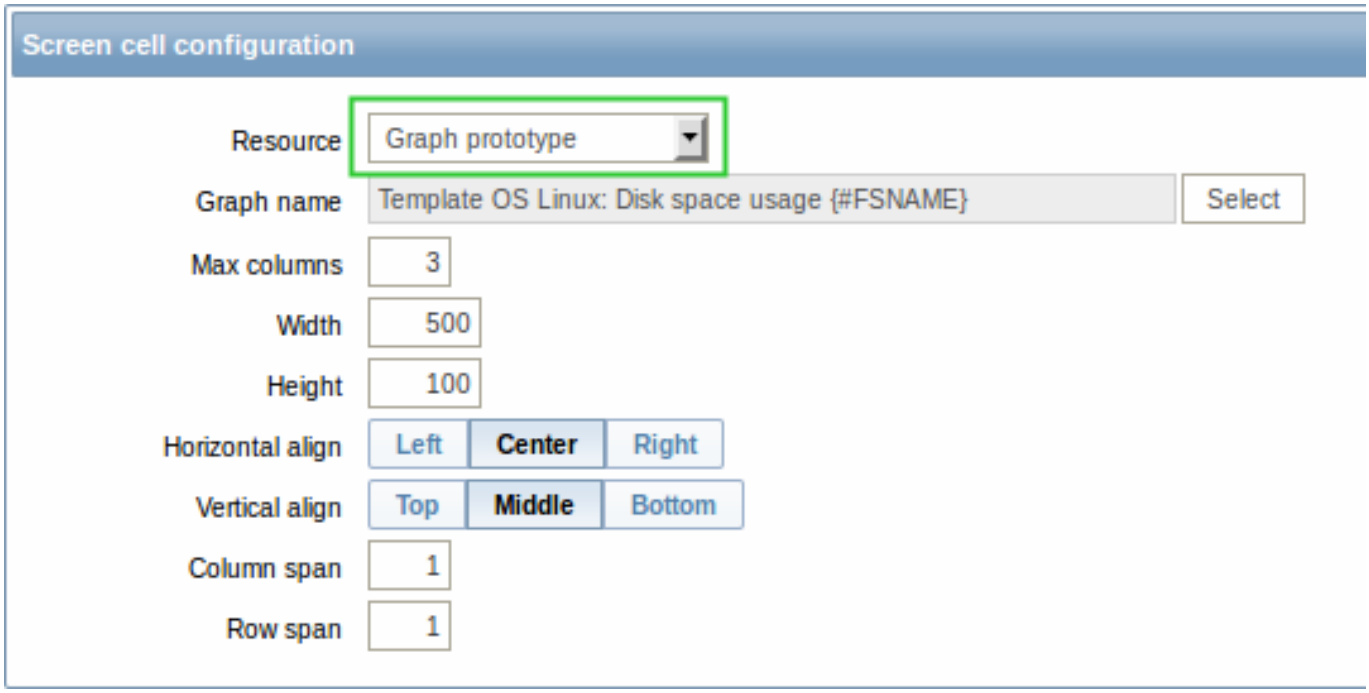
5.7 Optional SNMP bulk requests Bulk processing of SNMP requests was first introduced in Zabbix 2.2.3. While providing benefits in terms of reduced network traffic and load on SNMP devices, it also encountered problems with a number of devices that did not respond to these requests as expected.

As a result, SNMP bulk processing has been made optional in Zabbix 2.4.0. Enabled by default, it can be disabled per interface in host configuration by unchecking the respective option:

5.8 Graph prototypes supported in screens Zabbix screens have gained two new resources:

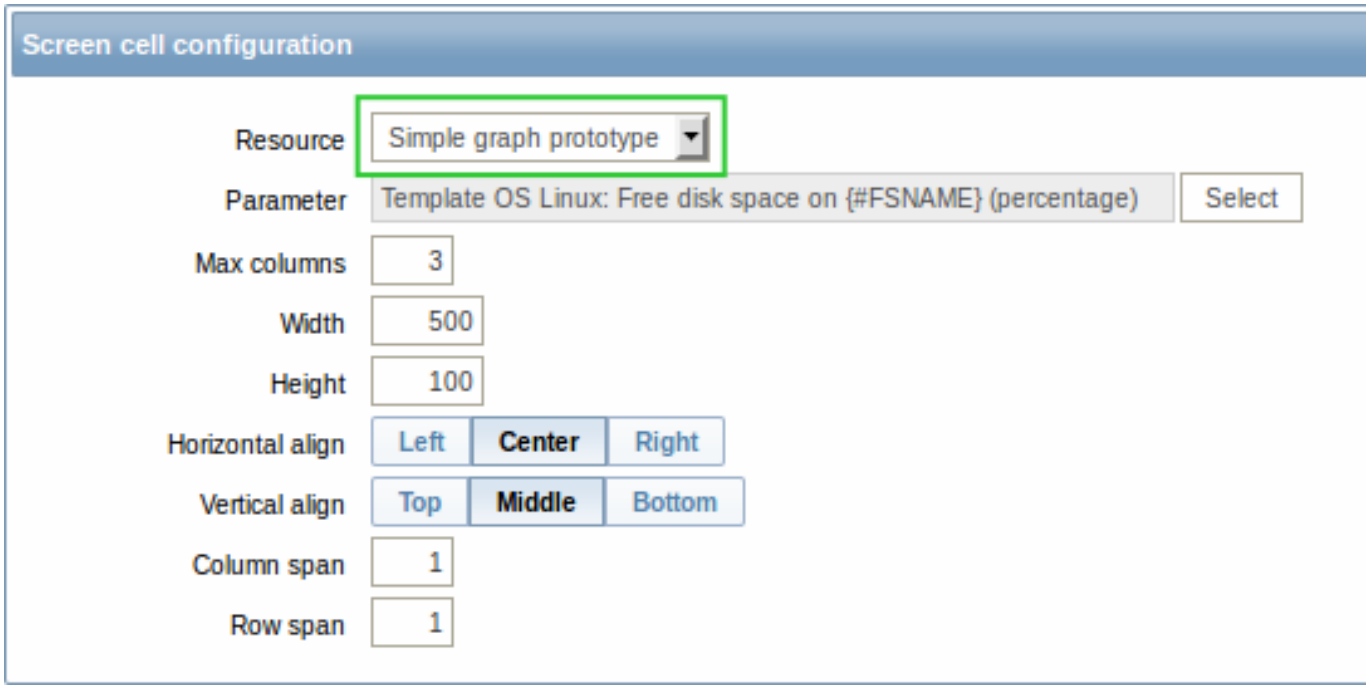
- Graph prototype;
- Simple graph prototype.

The *Graph prototype* resource is based on custom graph prototypes created in low-level discovery (LLD) rules.



In monitoring, the screen cell will display an LLD-generated graph as soon as it is generated. If the graph is not generated, nothing will be displayed.

A *Simple graph prototype* is based on item prototypes in low-level discovery. In monitoring, the screen cell will display a graph created from an LLD-generated item. If the item is not generated, nothing will be displayed.



The functionality is supported for host and template screens. With template screens, graph prototypes can be selected from the respective template only.

5.9 Option to remove missing resources when importing When using the XML import function before, it was possible to create new elements and update those that already existed. While that seemed to cover most needs, one option was missing.

For example, you would export a template, then update it by removing some items and triggers from it. However, when importing the same template back, the removed items and triggers would again be there since they are still on the original template. For this situation, Zabbix 2.4 offers a new *Delete missing* option for deleting resources that are not in the imported XML file. This option is implemented for host and template import.

Note that the host/template macros not present in the imported XML file will be deleted too.

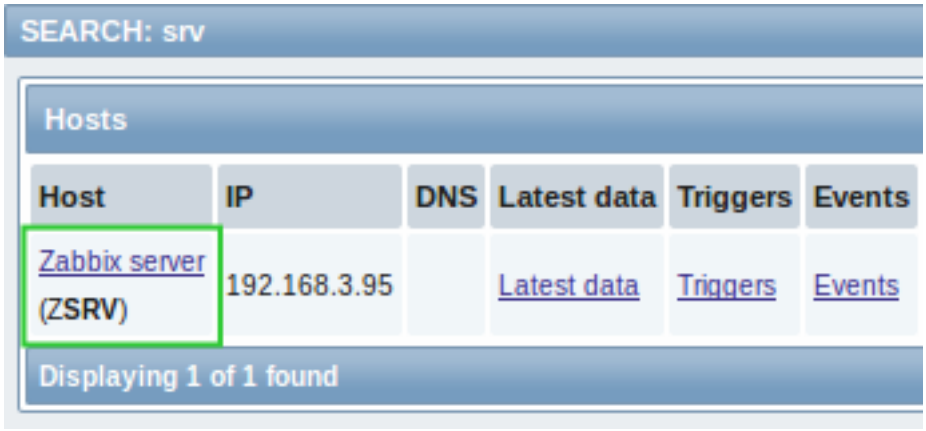
Additionally, the *Add missing* option has been renamed to *Create new*, to avoid confusion with the new option.

5.10 Frontend improvements 5.10.1 Unified date and time format

Date and time format has been unified across Zabbix frontend according to the ISO 8601 standard of **YYYY-MM-DD hh:mm:ss**. The new format is implemented everywhere except graphs.

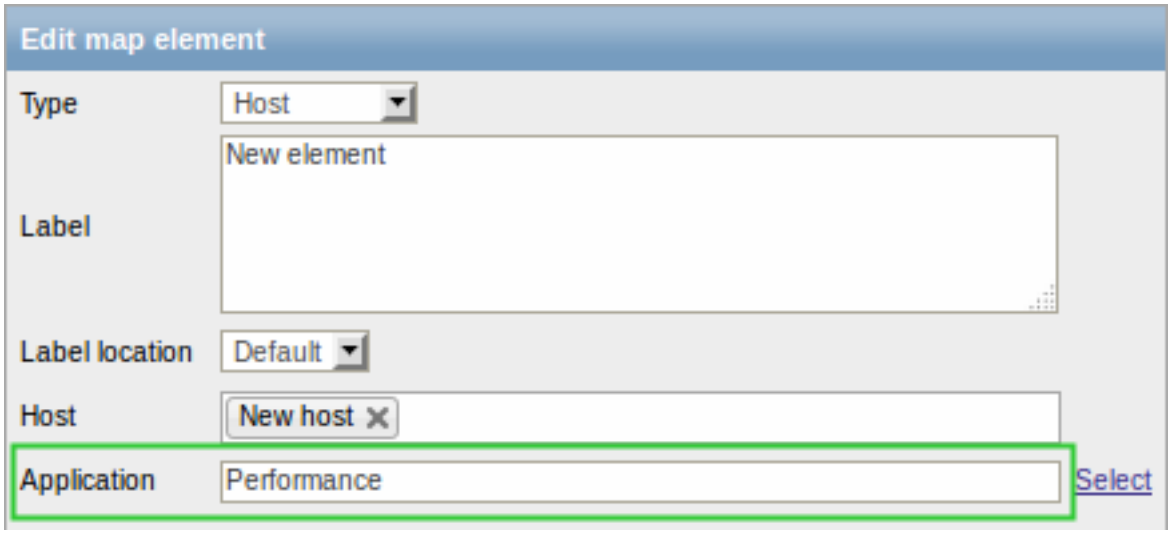
5.10.2 Global search by technical name

In recent Zabbix versions, hosts and templates could only be searched for by their visible name. Now they can be searched by technical name as well. If a match is found for the technical name, it is displayed in parenthesis below the visible name.



5.10.3 Application filter for maps

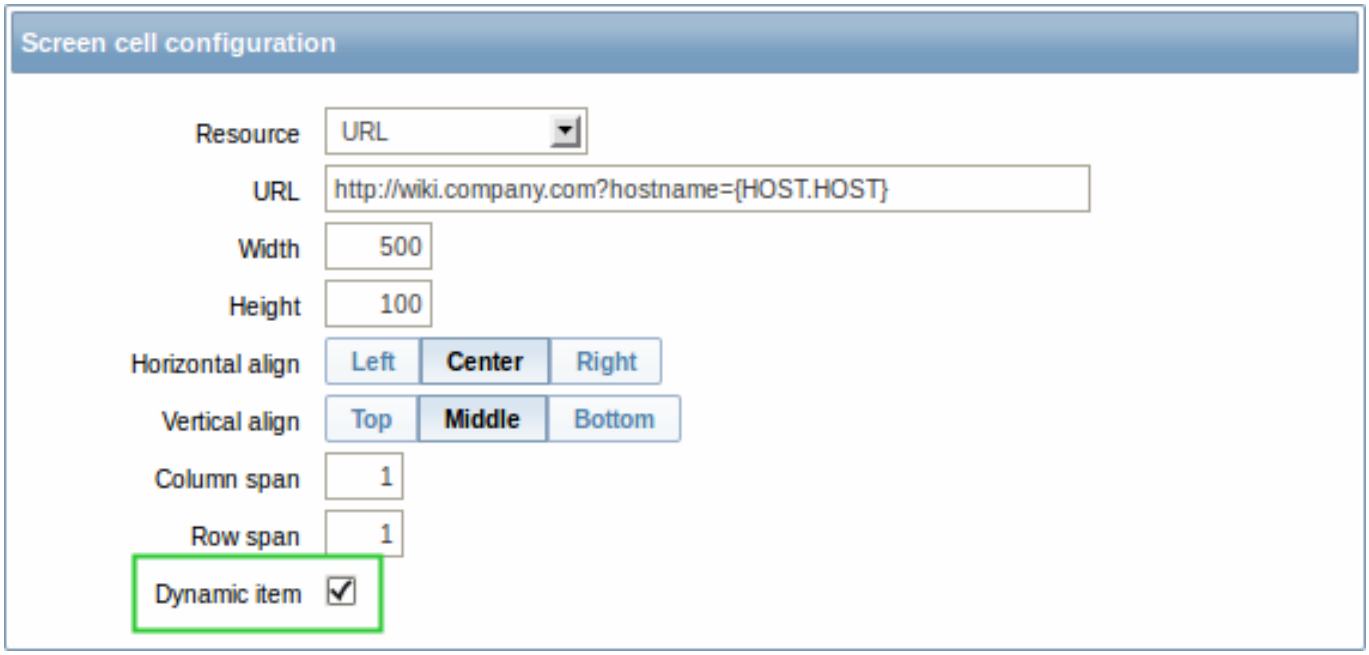
A new ”Application” filter has been added for host and host group map elements in maps.



The field can contain a name of an application and allows to only display problems of triggers that belong to the given application.

5.10.4 URL as dynamic screen element

In screens, URL now is a dynamic screen element:

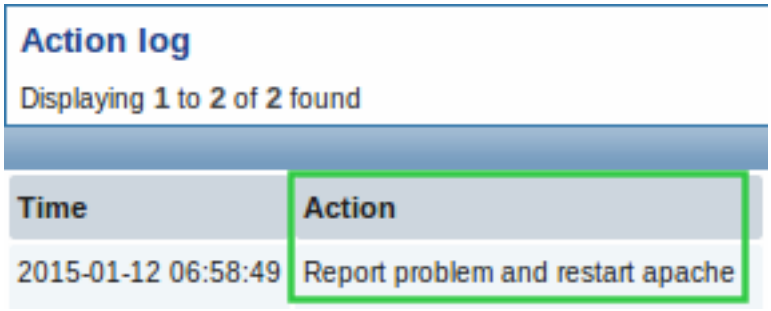


To support the new functionality, several macros are supported in the URL field: {HOST.CONN}, {HOST.DNS}, {HOST.ID}, {HOST.IP}, {HOST.HOST}, {HOST.NAME} and {$MACRO} user macro.

5.10.5 Responsible action shown in action log

The action log (displayed in the *Administration → Audit* section and in the *History of actions* screen element) would previously display various details about action operations (such as notifications sent, commands executed), however, the name of the responsible action was not displayed.

Now the Action log/Action log screen element have gained a new **Action** column showing the name of the responsible action as well.



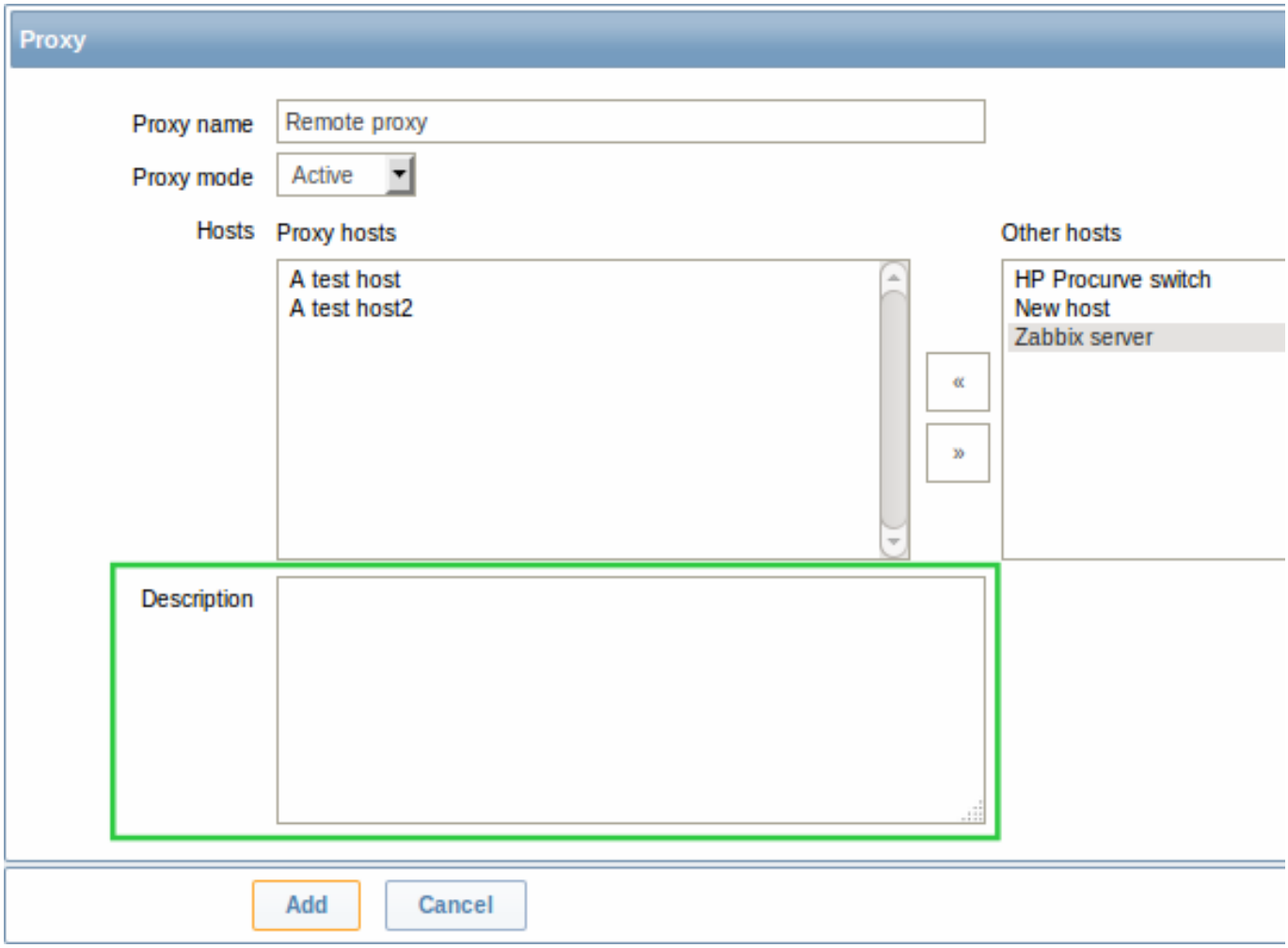
Other improvements include:

- renaming page titles in *Administration → Audit*:
  - 'Actions' to 'Action log'
  - 'Logs' to 'Audit log'
- renaming 'History of actions' screen element to 'Action log'
- showing user alias, name and surname with notification recipient information (in addition to the previously displayed e-mail address)
- merging action operation *Status* and *Retries left* columns into one *Status* column.

For more information, see the audit section documentation.

5.10.6 Description field added

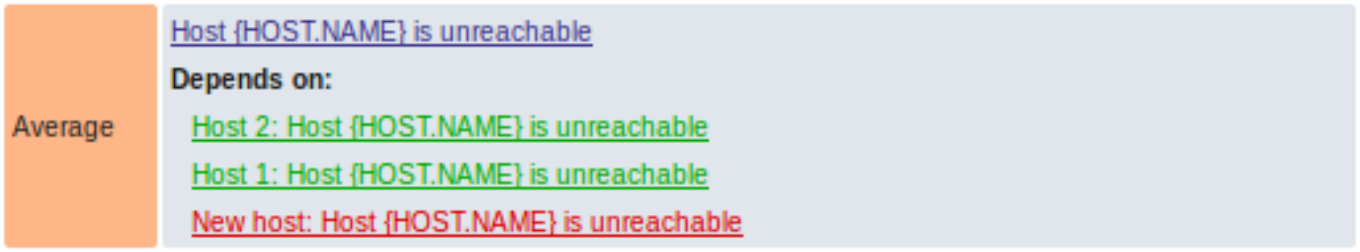
A new description field has been added to **host**, **template** and **proxy** configuration. The field may be used to provide details on how to install and use a template, have links to external resources, list user parameters, etc.



For hosts, the description field content is also visible in host inventory overview.

5.10.7 Trigger dependencies shown as links

Trigger dependencies previously were displayed as just plain text listing of trigger names. Now the trigger names are displayed as links leading to the trigger configuration. Links are green for enabled triggers, while red indicates that the trigger is disabled.

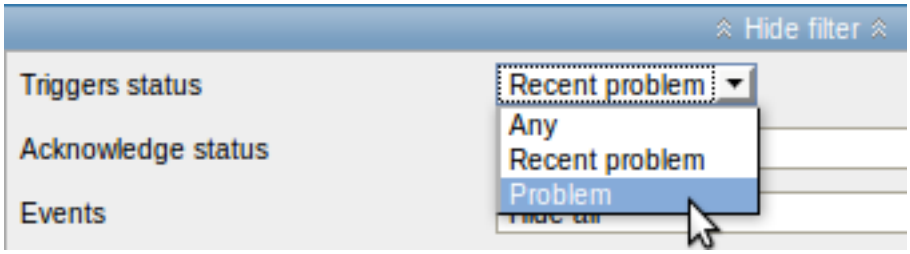


Dependencies are also shown as links (in blue) in trigger configuration and mass update forms. Additionally, a comma separated host list is displayed there if a trigger belongs to multiple hosts.

5.10.8 Option to see problems only in trigger status page

Filtering options in *Monitoring* → *Triggers* previously allowed to either show all triggers regardless of status, or to show problem triggers, which included those that have recently switched to OK.

The second option meant that if many triggers were switching to OK simultaneously, it became difficult to spot triggers that still remained in problem status. To deal with this situation, a third specific filtering option is now available showing only those triggers that still remain in problem status - this option now is called *Problem*.

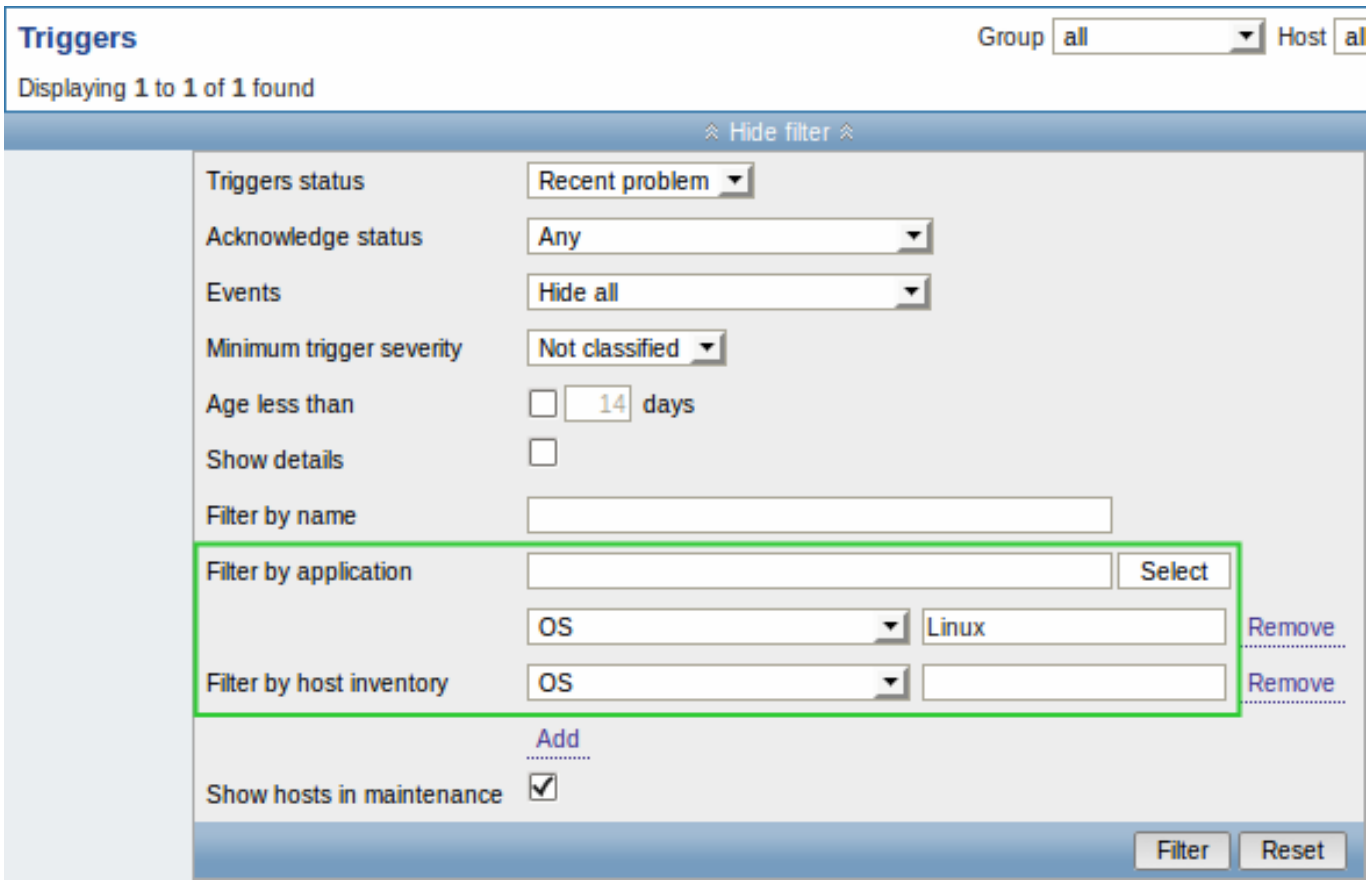


The first two options remain in place, however, the one that used to be called *Problem* now is called *Recent problem*.

5.10.9 More options for trigger filtering

Filtering in *Monitoring* → *Triggers* has gained a couple of new options:

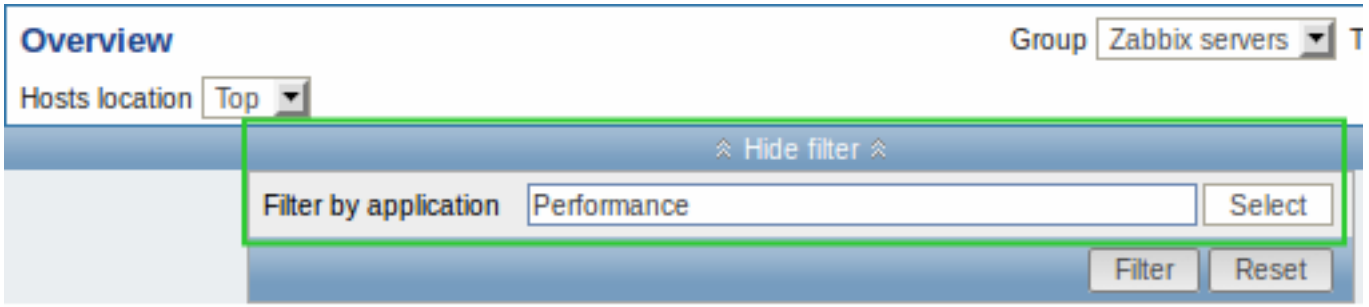
- *Filter by application*
- *Filter by host inventory* (multiple values can be used)



Additionally, the fairly large trigger filter now features in two frontend sections:

- *Monitoring* → *Triggers*
- *Monitoring* → *Overview* (with *Triggers* selected in the Type dropdown)

Note that in *Monitoring* → *Overview*, with *Data* selected in the Type dropdown, a small filter is displayed offering the possibility to filter data by application (an option, which was introduced in 2.2 version as an additional dropdown in the title bar).



5.10.10 Maintenance period sorting

Maintenance periods can now be sorted by two new columns added to the list - *Active since* and *Active till*:

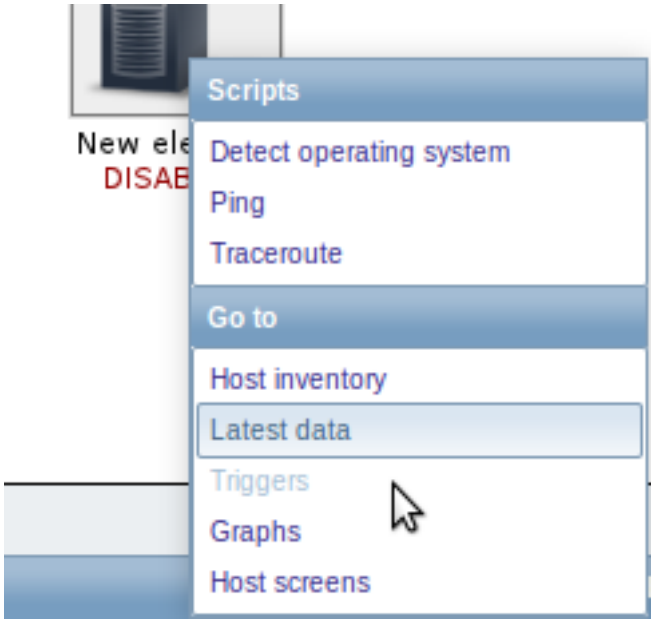
5.10.11 Host menu changes

Linking to graphs

The host menu has gained a new entry for quickly accessing host graphs.

Unavailable links shown

Unavailable links that previously were hidden now are shown as disabled - meaning greyed out and not clickable.



Synchronized in maps

The menu available for hosts in *Monitoring* → *Maps* has been synchronized with other host menus to display the same selection of links.

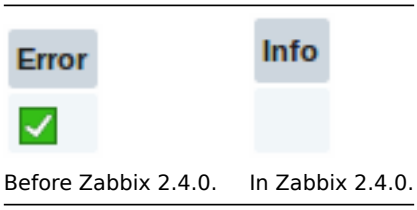
The host menu is accessible by clicking on a host in several frontend sections. See more details about the host menu.

5.10.12 Regular expression validation

Regular expressions are now validated before saving a global regular expression. For incorrect regular expressions errors are also displayed when testing.

5.10.13 Dropping green status icons in listings

Previously, green status icons were displayed in the last *Error* column of the item and trigger listings, for error-free entries, which could be misunderstood as if, on the contrary, the entries had errors. Now the green icons are displayed no more and, additionally, the column is renamed to *Info*.



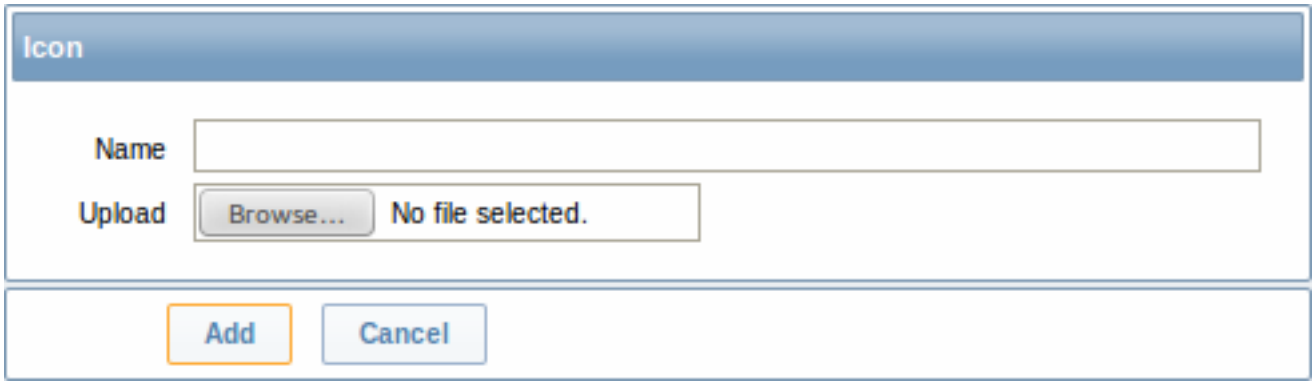
Before Zabbix 2.4.0. In Zabbix 2.4.0.

For items/triggers with problems, as before, a red square icon is displayed in this column.

5.10.14 Changing uploaded image type disallowed

The type of uploaded images previously could be freely changed from *Icon* to *Background* in the Images section of *Administration → General*.

That could be potentially dangerous since an icon, used in a network map, could be changed to *background* thus making the map unsaveable in the future. To avoid this situation, the option of changing image type has been removed and the image upload form has been separated into two forms - one for icon and one for background upload.



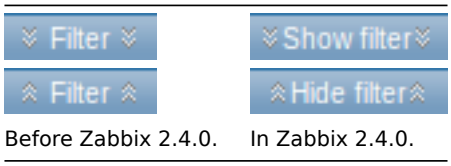
This is also indicated by different names of the upload buttons - *Create icon* and *Create background*.

5.10.15 Changes regarding multibyte string support

Zabbix frontend now displays fatal error if mbstring PHP functions are not enabled. There are no string wrapper functions anymore; they have been replaced by string and/or mbstring functions accordingly. Additionally, the `mbstring.func_overload` option from php.ini is required to be off, and a warning will be displayed in *Status of Zabbix* if it is enabled.

5.10.16 Miscellaneous improvements

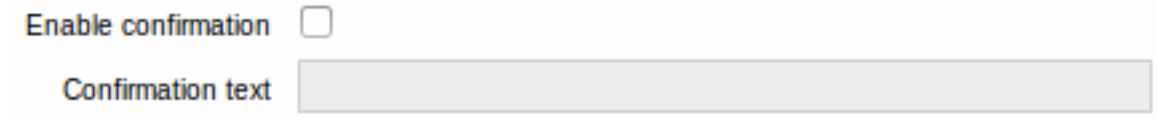
- *Save* buttons have been replaced by *Add* or *Update* depending on whether a new entry is created or an existing one updated.
- More intuitive naming for the use of filter has been introduced. If previously *Filter* stood for both show and hide functions (with only very tiny icons indicating the direction), now separate *Show filter* and *Hide filter* names have been introduced.



Before Zabbix 2.4.0. In Zabbix 2.4.0.

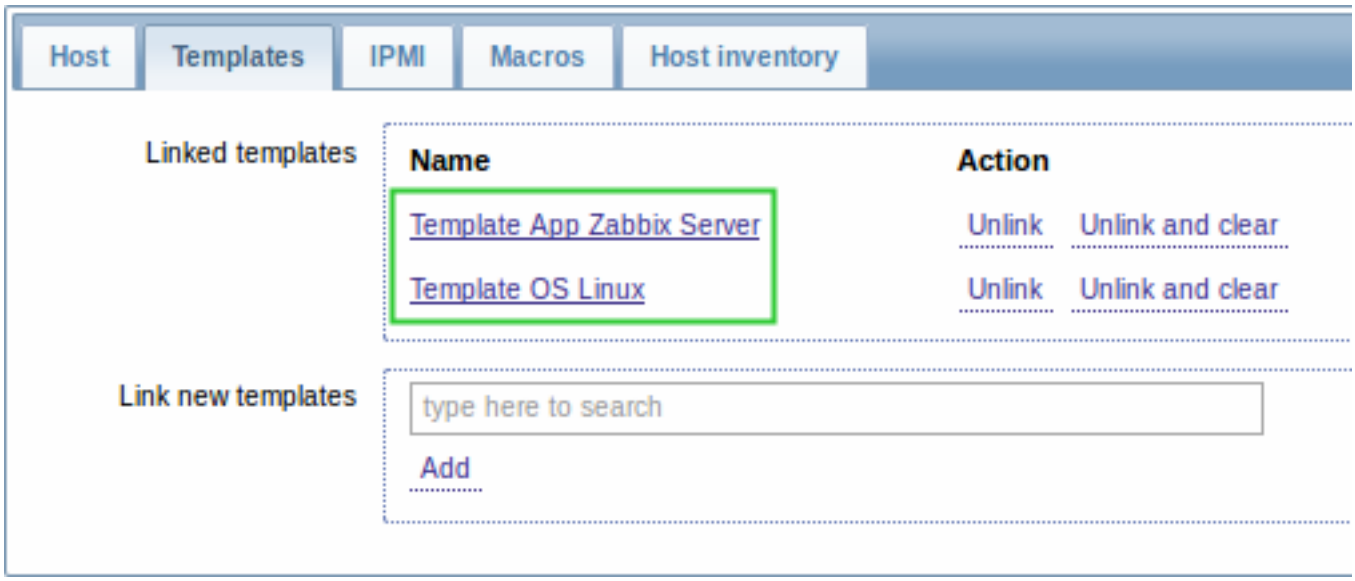
- Disabled input fields previously were not distinguished much, which could create confusion as to why they were not available for editing. Now these fields are distinguished with a grey background, for example, in host inventory, web scenario step or global script definition forms.

Before Zabbix 2.4.0.



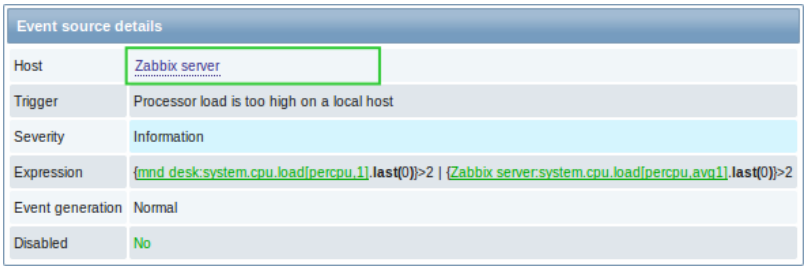
In Zabbix 2.4.0.

- The *Rows per page* setting from User profile previously was not applied to the *Availability report* section of the frontend, leading to difficulties when loading the page on installations with a lot of triggers. Now the setting is in force and can be used to limit the number of records displayed in one page.
- Templates listed in the host configuration have been made clickable. Clicking on a template name opens the template configuration form.

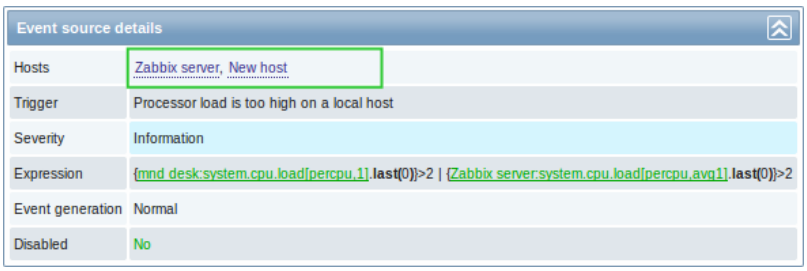


- Host groups listed in the *Web monitoring* widget of the frontend have been made clickable. The link leads to *Monitoring* → *Web* with scenarios of the respective host group selected.

- Previously, only one host was listed in event details, even if the trigger expression contained several. Now, all hosts from the expression are listed.



Before Zabbix 2.4.0.



In Zabbix 2.4.0.

- *Monitoring* → *Latest data* filter option *Show items without data* is now enabled by default
- *Cancel* button has been added to *Monitoring* → *Dashboard* filter
- In the user list, disabled groups in the *Group* column are now displayed in red
- In the user group list, *Status* column has been moved to the end
- The value of `Max count of elements to show inside table cell` from *Administration* → *General* → *GUI* now also applies to templates listed in the host list, users listed in the user group list and user groups listed in the user list
- When a trigger belongs to many hosts, hosts are now displayed in alphabetical order
- For numeric items with disabled history ('history' and 'trends' set to 0), the *Graph* link is hidden from view in *Monitoring* → *Latest data*. If history and trends are set to '0' globally in *Administration* → *General* → *Housekeeping*, then all *Graph/History* links are hidden in *Latest data*.
- Hyperlinks in the first column of *Inventory* → *Overview* now are displayed properly as links, not plain text.
- If evaluation for nodata() function fails due to lack of data on the server, a more informative message is displayed. Previous vague message of *Evaluation failed for function...* has been expanded to include *item does not have enough data after server start or item creation*.
- *Auto-login* and *Auto-logout* controls have been removed from 'guest' user profile, as unnecessary.
- Saving global regular expressions with a leading space (like " 1") now works correctly. Moreover, spaces and tabs are dealt with correctly before, after and within expression text.
- A forward slash (/) in global regular expressions 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.
- Proxy name is now displayed as host prefix in the item queue details page.
- The trigger filter will no longer be reset when selecting all hosts on the *Monitoring* → *Events* page.
- IT services in pop-up windows are now sorted by name.
- IT service dependencies in the configuration window are now sorted by name.

5.11 Macro improvements User macros {$MACRO} now are available in notifications and commands.

Two new macros - `{HOST.DESCRIPTION}` and `{PROXY.DESCRIPTION}` are supported for use in notifications and commands. `{HOST.DESCRIPTION}` can also be used in map labels.

Host level macros - `{HOST.HOST}`, `{HOST.NAME}`, `{HOST.IP}`, `{HOST.DNS}` and `{HOST.CONN}` along with user macros `{$MACRO}` are available in simple low-level discovery rule filter regexps.

For more details, see Macros supported by location.

5.12 Daemon improvements Zabbix application names appearing in syslog have been corrected. The new names are: `zabbix_server`, `zabbix_proxy`, `zabbix_agentd`, and `zabbix_agent`.

** Low-level discovery **

A new item key has been introduced (system.cpu.discovery) that allows enumerating processors and their states on the target machine. On *nix systems, the processors (or cores) will be reported in two states - as being "online" or "offline". On Windows a third, "unknown" state, is possible. A processor is considered to be in an unknown state when it has been detected, but no data has been collected from the performance counter.

** Internal items always processed **

Internal items previously were not processed if a host went into maintenance with 'no data collection'. Now internal items are always processed by server or proxy regardless of host maintenance status.

** Config file handling **

Any problem in the config file is checked for and reported before writing to the logfile. Error message is printed in the terminal window now.

`zabbix_agentd` no longer requires a configuration file to be present, loaded or specified when launched with options that don't require a configured setup on the Windows platform.

Alias parameter keys in the agent configuration file are now validated to be proper Zabbix item keys. These keys now also support colon symbols in their parameters.

A new configuration parameter (User) has been introduced for daemons which allows dropping privileges to the specified user if the daemon has been started by the root account.

When using the Include parameter in the agent configuration file for Windows it is now possible to include all files from a directory.

Server and proxy now refuse to start if **StartPollersUnreachable** configuration parameter is 0, but regular, IPMI or Java pollers are started. Otherwise, hosts that become unreachable would never be checked again.

** Ability to include specific files from Include directory **

Pattern matching has been introduced in the Include directives of Zabbix daemon configuration files, using the asterisk wildcard character. Thus, to only include relevant files in the specified directory, something like `/path/to/config/files/*.conf` may be used.

** JSON formatting **

The formatting of JSON objects (with tabs and new lines) has been removed, which allows the traffic to be reduced by 20-30% when data is sent between Zabbix services. Additionally, the escaping of forward slash or solidus (/) has also been removed.

** Log file handling **

At DebugLevel=0 only basic information is printed to the logfile/syslog. For agents it is Zabbix version, used config file, started/stopped Zabbix processes. For proxy and server - Zabbix version, enabled features, used config file, DB version, started/stopped Zabbix processes. The logging is also consistent on Linux and Windows machines now.

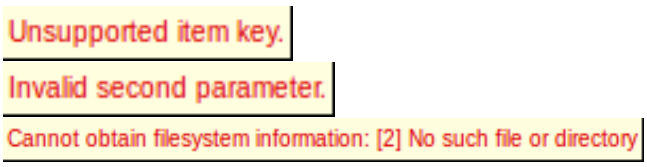
** Handling automatically discovered hosts **

Hosts discovered by a network discovery rule will be removed automatically from *Monitoring* → *Discovery* if a discovered entity is not in the rule's IP range any more. Hosts are removed immediately.

** More details about unsupported agent items **

Zabbix agents now provide detailed information on why items become not supported, instead of the generic "Not supported by Zabbix Agent". This was achieved by extending the passive agent and active agent protocols (see protocol documentation). However, new agents are compatible with older server and proxy, except the error message for passive agents will not be visible in the frontend. Querying values with `zabbix_get`, testing with `zabbix_agentd -p` and `zabbix_agentd -t` now also provides a detailed error message.

As a result, in *Configuration* → *Hosts* → *Items* of the frontend, when rolling the mouse over the error icon, you may expect to see more specific messages about why an item went unsuported.

	 ...
A single message before Zabbix 2.4.0.	More informative messages in Zabbix 2.4.0.

Additionally, maximum message length has been increased from 128 to 2048 symbols. This should reduce the possibility of error message being trimmed.

** Database related changes **

The DBSchema configuration parameter in both the `zabbix_server` and `zabbix_proxy` configuration now also applies to PostgreSQL databases in addition to IBM DB2. If this parameter is left undefinied, a default of `public` is used as the name of the DBSchema.

A check that prevents starting the proxy with a server database and vice versa has been added.

Support for PHP mutexes has been removed on the server side due to licensing issues. While it was not recommended to use Zabbix server and frontend with SQLite3 database before, this change makes it even less recommended, because simultaneous database access with Zabbix server and frontend may now corrupt the database. Note that using Zabbix proxy with SQLite3 database is still a perfectly valid solution.

** Housekeeper changes **

On `zabbix_server` and `zabbix_proxy` startup housekeeping is postponed for 30 minutes instead of running at once. This will lower startup load for both these processes.

** Script result changes **

User parameters, global scripts, `system.run[]` items and external checks now return standard error along with standard output. Previously only standard output was returned (standard error was discarded).
Note: Windows agents have this functionality since 1.8.5

Empty result is now allowed for `system.run[]` items configured with textual value type (character, log or text).

** Log level change at runtime **

Two additional runtime control options have been added on all Zabbix services - *log_level_increase* and *log_level_decrease*. Now, it is possible to change the log level of all or certain process(-es) without restarting a service. These two runtime control options accept parameters for target process selection. A target process can be selected by specifying PID or process type and process number.

5.13 Item changes/improvements **zabbix[host,„maintenance]** is a new internal item that returns the current maintenance status of a host. It is always processed by the server regardless of host location (on server or proxy).

The existing **zabbix[proxy,<proxyname>,”lastaccess”]** internal item is now always processed by the server, which means that the item will work correctly in cases when a host is monitored by proxy.

vfs.dev.read and **vfs.dev.write** agent items are now supported on AIX.

system.swap.size agent items on Windows now support a percentage free parameter (system.swap.size[,pfree]). The way swap sizes are calculated has changed due to issues discovered in Windows APIs reporting of these metrics. Swap size is now calculated from the total memory space available (physical and virtual memory combined). Although these results are more accurate than they were before, they still may return inaccurate data depending on various circumstances and setups. Commentary on the issues has been added to src/libs/zbxsysinfo/win32/swap.c

5.14 API improvements ** Dropped support of output ”refer” **

Support of the ”refer” value for get method ”output” parameter has been removed. Instead it’s recommended to use arrays of specific properties.

6 What’s new in Zabbix 2.4.1

6.1 Frontend improvements 6.1.1 Updated translations

- Brazilian Portuguese
- German
- Slovak

6.2 Daemon improvements

- Value cache requests have been optimized to better utilize database indexes. The improvement would be mostly noticeable with large databases.

7 What’s new in Zabbix 2.4.2

7.1 Frontend improvements 7.1.1 Updated translations

- Brazilian Portuguese
- Greek
- Italian
- Japanese
- Polish
- Slovak
- Spanish

7.2 Daemon improvements

- History cache has been optimized to better handle a situation when it's being flooded with hundreds of thousands of values from less than a thousand items.
- SNMP polling logic has been improved to always retry at least once. This should make Zabbix more resilient to network errors.
- SNMP values of type OID are now supported.
- Validation of received SNMP responses has been added to server and proxy. Now, upon receiving a malformed SNMP response server and proxy will log lines similar to the following:

  `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 a useful indicator that bulk SNMP requests should be turned off for this device.

8 What's new in Zabbix 2.4.3

8.1 Frontend improvements

- History related macros - {ITEM.VALUE}, {ITEM.LASTVALUE} and the {host:key.last()} functional macro - now obey the ZBX_HISTORY_PERIOD parameter. This limits the amount of data the macro has to sift through and results in better performance.

8.1.1 Updated translations

- Brazilian Portuguese
- Chinese (China)
- Italian
- Japanese
- Slovak

8.2 Daemon improvements

- SNMP validation error messages from Zabbix 2.4.2 have been improved by including the sent and received OIDs:

  `SNMP response from host "gateway" contains variable bindings that do not match the request: sent ".1.3.6.1.2.1.2.2.1.16.9", received ".1.3.6.1.2.1.2.2.1.16.9.0"`For bulk requests, these are logged at DebugLevel=3. For single-variable requests, these are logged at DebugLevel=4.
- If an IPMI device reports a threshold sensor and a discrete sensor under the same name, the threshold sensor is now preferred. This might fix strange readings (like "1" for fan RPM) or "not supported" errors.
- Message logging on IBM DB2 errors has been improved. Now additional information is printed to the log file - database name on connection errors and SQL query on failed queries.

9 What's new in Zabbix 2.4.4

9.1 Extended IP range option in network discovery Previously, IP range matching was possible by specifying a range like 192.168.1.1-255. That, however, was not sufficient for easily specifying multiple subnets, thus the range option has now been extended to also allow specifying a range like 192.168.1-10.1-255.

The extended option is supported in discovery rules and in discovery-based action conditions. The new range is limited by the total number of covered addresses (less than 64K) in network discovery rules, but not limited in action conditions.

9.2 VMware monitoring improvements VMware performance collector based statistics retrieval was separated from VMware data retrieval:

- VMware collector now sends fewer requests to VMware servers, greatly improving performance of configuration data and performance collector based statistics gathering.
- VMware performance collector based statistics retrieval is much faster and can be done more frequently than VMware configuration data retrieval. To avoid it being delayed by VMware configuration data retrieval it is recommended to enable more VMware collectors than monitored services in your Zabbix server/proxy configuration.
- **vmware.vm.perfcounter** and **vmware.hv.perfcounter** items were added to allow custom hypervisor and virtual machine performance counter monitoring.

A configurable timeout was added to VMware data requests. See `VMwareTimeout` option in server and proxy configuration documentation.

VMware data requests were optimized to reduce the amount of transferred data by half.

9.3 Frontend improvements 9.3.1 Updated translations

- Brazilian Portuguese
- Chinese (China)
- Japanese
- Korean
- Polish
- Russian
- Slovak

9.3.2 Miscellaneous improvements

- Width of the *Host* column in *Monitoring* → *Latest data* has been increased
- Chinese (China) translation is 100% completed and is now displayed in the language dropdown
- Audit records about failed login attempts have been added

9.4 Daemon improvements

- When monitoring Windows eventlog Zabbix agent will no longer set an item state to NOTSUPPORTED in case of error when formatting the message. Instead, an unformatted message will be used.
- Item proc_info on Windows was improved to get more information about the processes.
- The `SourceIP` configuration parameter is now respected in web monitoring, VMware monitoring, Ez Texting alerts and **net.tcp.service{.perf}[https]** items.

9.5 Miscellaneous improvements 9.5.1 Validation of global regular expressions in LLD rules

A check for valid reference has been added for global regular expressions in LLD rules. If entered reference is not valid, due to misspelling or missing referenced global regular expression, the respective LLD rule will become unsupported and appropriate error message will be displayed.

10 What's new in Zabbix 2.4.5

10.1 Frontend improvements 10.1.1 Updated translations

- Brazilian Portuguese
- Chinese (China)
- Italian
- Japanese
- Polish
- Slovak

10.2 Daemon improvements

- For Java gateway, it is now possible to specify timeout for JMX network operations using **TIMEOUT** configuration option in *startup.sh*.
- In actions, it is now possible to execute a custom script on the server if trigger expression contains multiple hosts.

11 What's new in Zabbix 2.4.6

11.1 Frontend improvements Czech translation is 100% completed and is now displayed in the language dropdown.

11.1.1 Updated translations

- Brazilian Portuguese
- Chinese (Taiwan)
- Czech
- Dutch

- Japanese
- Spanish

11.2 Daemon improvements

- While item `net.tcp.service[ntp]` has existed for a long time, it almost never worked, because it tried to probe NTP protocol over TCP. It was rewritten to work over UDP and it now works.

12 What's new in Zabbix 2.4.7

12.1 Frontend improvements

12.1.1 Updated translations

- Chinese (China)
- Chinese (Taiwan)
- Czech
- French
- Russian
- Turkish

12.2 Daemon improvements

12.2.1 Alias for active checks

Using an Alias will now work properly with active-only checks (**log**, **logrt**, **eventlog**).

12.2.2 system.cpu.discovery

When using system.cpu.discovery on Solaris agent will now return JSON with correct value of {#CPU.NUMBER} that is suitable for item prototype.

12.3 Miscellaneous improvements

The default error message displayed for unsupported items has been changed from "Item is not supported." and "ZBX_NOTSUPPORTED" to "Unknown error.". This message is used when an item check fails without setting error message.

Input file description in the zabbix_sender manpage has been improved by adding rules and examples.

Zabbix now tries to differentiate item timeouts from host timeouts. If another item check was successful between two failed checks of a problematic item, then the problematic item is marked as not supported after the second failed check without affecting host availability.

13 What's new in Zabbix 2.4.8

Warning:
Zabbix 2.4 is not supported anymore. See lifecycle and release policy page for more information.

13.1 Frontend improvements

13.1.1 Updated translations

- English (United States)
- French
- Japanese
- Korean
- Polish
- Romanian
- Russian
- Slovak
- Spanish
- Turkish
- Ukrainian
- Vietnamese

Enabled French, Greek, Korean, Romanian, Ukrainian, Vietnamese translations to be displayed by default.

13.1.2 Performance improvements

- Improved performance and memory usage in screens with a large amount of screen elements

13.1.3 Dashboard host status widget

- Previously, when using the dashbaord filter *Unacknowledged only* option, acknowledged problem triggers were displayed neither in *With problems* nor *Without problems* columns of the host status widget, resulting in a wrong host count in total. Now the acknowledged problem triggers are displayed in the *Without problems* column.

13.2 Daemon improvements

- Item key length limitation of 2KB has been removed on Zabbix server when sending item key to the agent
- Item key length limitation of 1KB has been removed from the -k option of zabbix_get
- **wmi.get** item was improved to accept UTF-8 encoded namespace, WQL query and encode returned string in UTF-8
- The detection of a single item failing with network/timeout error introduced in Zabbix 2.4.7 was removed because of inability to distinguish possible network errors.

13.2.1 Trend cache improvements

Trends of items that received no data during the first 55 minutes of the hour will be flushed to database and removed from the cache. This means no more Zabbix server restarts due to lack of free space in trend cache. Also trends of items with update interval longer than one hour will get into the database earlier.

13.3 Miscellaneous improvements

2. Zabbix concepts

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

1 Zabbix 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 off of a host, a metric of data.

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 auto-registration taking place

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

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 agent

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

Zabbix proxy

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

2 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.

Server process

If installed as package

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

```
shell> service zabbix-server start
```

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

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

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

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

Start up manually

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

```
shell> zabbix_server
```

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

```
-c --config <file>        absolute path to the configuration file (default is /usr/local/etc/zabbix_server
-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.	
log_level_increase[=<**target**>]	Increase log level, affects all processes if target is not specified.	**pid** - Process identifier (1 to 65535) **process type** - All processes of specified type (e.g., poller) **process type,N** - Process type and number (e.g., poller,3)
log_level_decrease[=<**target**>]	Decrease log level, affects all processes if target is not specified.	

Allowed range of PIDs for changing the log level of a single Zabbx process is from 1 to 65535. On systems with large PIDs <process type,N> target option can be used for changing the log level of a single process.

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 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.

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.

3 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 for:

- Linux
- IBM AIX
- FreeBSD
- NetBSD
- OpenBSD
- HP-UX
- Mac OS X
- Solaris: 9, 10, 11
- Windows: all desktop and server versions since 2000

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> service zabbix-agent start
```

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

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

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

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

Start up manually

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

```
shell> zabbix_agentd
```

Agent on Windows systems

Zabbix agent on Windows runs as a Windows service.

Preparation

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

```
C:\zabbix
```

Copy bin\win64\zabbix_agentd.exe and conf\zabbix_agentd.win.conf files to c:\zabbix.

Edit the c:\zabbix\zabbix_agentd.win.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.win.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>	Absolute 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
-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: `pid` - process identifier (1 to 65535) `process type` - all processes of specified type (e.g., poller) `process type,N` - process type and number (e.g., poller,3)
log_level_decrease[=<target>]	Decrease log level. If target is not specified, all processes are affected.	

Note that the usable range of PIDs for changing the log level of a single agent process is 1 to 65535. On systems with large PIDs, the <process type,N> target can be used for changing the log level of a single process.

Examples:

- increasing log level of all processes
- increasing log level of the second 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,2
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.

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.

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 or IBM DB2 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

Proxy process

If installed as package

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

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

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

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

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

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

Start up manually

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

```
shell> zabbix_proxy
```

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

```
-c --config <file>              absolute path to the configuration file
-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.	
log_level_increase[=<**target**>]	Increase log level, affects all processes if target is not specified.	**pid** - Process identifier (1 to 65535) **process type** - All processes of specified type (e.g., poller) **process type,N** - Process type and number (e.g., poller,3)
log_level_decrease[=<**target**>]	Decrease log level, affects all processes if target is not specified.	

Allowed range of PIDs for changing the log level of a single Zabbx process is from 1 to 65535. On systems with large PIDs <process type,N> target option can be used for changing the log level of a single process.

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 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.

Supported platforms

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

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, since Zabbix 2.0.15, Zabbix 2.2.10 and Zabbix 2.4.5 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.

Sections below describe 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 GUI that correspond to particular JMX counters.

1 Getting Java gateway

There are two ways to get Java gateway. One is to download Java gateway package from Zabbix website and the other is to compile Java gateway from source.

1.1 Downloading from Zabbix website

Zabbix Java gateway packages (RHEL, Debian, Ubuntu) are available for download at http://www.zabbix.com/download.php.

1.2 Compiling from source

In order to compile Java gateway, you first run `./configure` script with `--enable-java` option. It is advisable that you specify `--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 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
```

2 Overview of files in Java gateway distribution

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

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

Java gateway JAR file itself.

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

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

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

Configuration files for Logback.

```
shutdown.sh
startup.sh
```

Convenience scripts for starting and stopping Java gateway.

`settings.sh`

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

3 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.

4 Configuring server for use with Java gateway

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

5 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.

6 Sender

Overview

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

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

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

Running Zabbix sender

An example of running Zabbix UNIX sender:

```
shell> cd bin
shell> ./zabbix_sender -z zabbix -s "Linux DB3" -k db.connections -o 43
```

where:

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

Attention:
If objects have whitespaces, these objects 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.

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

Zabbix sender on Windows can be run similarly:

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

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

If sending many values from an input file, Zabbix sender will batch them at 250 values in one go (all values will be processed), for example:

```
# zabbix_sender -z 127.0.0.1 -i /tmp/traptest.txt
Info from server: "Processed 250 Failed 0 Total 250 Seconds spent 0.002668"
Info from server: "Processed 50 Failed 0 Total 50 Seconds spent 0.000540"
sent: 300; skipped: 0; total: 300
```

All entries from an input file are sent in a sequential order top-down.

If the target item has triggers referencing it, all timestamps in an input file must be in an increasing order, otherwise event calculation will not be correct.

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

7 Get

Overview

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

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

Running Zabbix get

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

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

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

```
shell> cd bin
shell> ./zabbix_get -s 192.168.1.1 -p 10050 -k "web.page.regexp[www.zabbix.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.
```

Zabbix get on Windows can be run similarly:

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

3. Installation

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

1 Getting Zabbix

Overview

There are three ways of getting Zabbix:

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

To download the latest sources or the virtual appliance, go to the Zabbix download page, where direct links to latest versions are provided. To download older versions, see the link below stable version downloads.

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. It is highly recommended to run the database on a separate box for large installations.

Supported platforms

Due to security requirements and mission-critical nature of 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 is tested on the following platforms:

- Linux
- IBM AIX
- FreeBSD
- NetBSD
- OpenBSD
- HP-UX
- Mac OS X
- Solaris
- Windows: all desktop and server versions since 2000 (Zabbix agent only)

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

Software

Zabbix is built around a modern Apache web server, leading database engines, and PHP scripting language.

Database management system

Software	Version	Comments
MySQL	5.0.3 or later	Required if MySQL is used as Zabbix backend database. InnoDB engine is required.
Oracle	10g or later	Required if Oracle is used as Zabbix backend database.
PostgreSQL	8.1 or later	Required if PostgreSQL is used as Zabbix backend database. It is suggested to use at least PostgreSQL 8.3, which introduced much better VACUUM performance.
SQLite	3.3.5 or later	Required if SQLite is used as Zabbix backend database.
IBM DB2	9.7 or later	Required if IBM DB2 is used as Zabbix backend database.

Attention:
IBM DB2 support is experimental!

Attention:
While SQLite3 can be used with Zabbix proxies without any problems, using SQLite3 with Zabbix server is not recommended. Since Zabbix 2.4.0, simultaneous database access with server and frontend may even lead to database corruption!

Frontend

The following software is required to run Zabbix frontend:

Software	Version	Comments
Apache	1.3.12 or later	
PHP	5.3.0 or later	PHP v7 is not supported.
PHP extensions:		
gd	2.0 or later	PHP GD extension must support PNG images (*--with-png-dir*), JPEG (*--with-jpeg-dir*) images and FreeType 2 (*--with-freetype-dir*).
bcmath		php-bcmath (*--enable-bcmath*)
ctype		php-ctype (*--enable-ctype*)
libXML	2.6.15 or later	php-xml or php5-dom, 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.
ibm_db2		Required if IBM DB2 is used as Zabbix backend database.

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.
sqlite3		Required if SQLite 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 in Monitoring → Overview header and other locations. 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 versions of Google Chrome, Mozilla Firefox, Microsoft Internet Explorer and Opera are supported. Other browsers (Apple Safari, Konqueror) may work with Zabbix as well.

Server

Requirement	Description
OpenIPMI	Required for IPMI support.
libssh2	Required for SSH support. Version 1.0 or higher.
fping	Required for ICMP ping items.
libcurl	Required for web monitoring and VMware monitoring.
libiksemel	Required for Jabber support.
libxml2	Required for VMware monitoring.
net-snmp	Required for SNMP support.

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	Comments
logback-core-0.9.27.jar	EPL 1.0, LGPL 2.1	http://logback.qos.ch/	Tested with 0.9.27, 1.0.13, and 1.1.1.
logback-classic-0.9.27.jar	EPL 1.0, LGPL 2.1	http://logback.qos.ch/	Tested with 0.9.27, 1.0.13, and 1.1.1.
slf4j-api-1.6.1.jar	MIT License	http://www.slf4j.org/	Tested with 1.6.1, 1.6.6, and 1.7.6.

Library	License	Website	Comments
android-json-4.3_r3.1.jar	Apache License 2.0	https://android.googlesource.com/platform/libcore/+/master/json	Tested with 2.3.3_r1.1 and 4.3_r3.1. See src/zabbix_java/lib/README for instructions on creating a JAR file.

Java gateway compiles and runs with Java 1.6 and above. It is recommended that those who provide a precompiled version of the gateway for others use Java 1.6 for compilation, so that it runs on all versions of Java up to the latest one.

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. 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 customised.

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 170 bytes of disk space. 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.

It means that if we want to keep 3 years of events, this would require **3***365*24*3600* **170** = **15GB**

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.

Parameter	Formula for required disk space (in 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 ~170 bytes.

Note:
Average values such as ~90 bytes for numeric items, ~170 bytes for events have been gathered from real-life statistics using a MySQL backend database.

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 synchronisation

It is very important to have precise system date 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 synchronised system date on all systems Zabbix components are running on.

If the time is not synchronised Zabbix will convert timestamps of the gathered data into Zabbix server time by taking client/server timestamps after establishing data connection and adjusting the received item value timestamps by the client-server time difference. To keep it simple and avoid possible complications the connection latency is ignored. Because of that the connection latency is added to the timestamps of data acquired from active connections (active agent, active proxy, sender) and subtracted from the timestamps of data acquired from passive connections (passive proxy). All other checks are done in server time and their timestamps are not adjusted.

3 Installation from packages

From distribution packages Several popular OS distributions have Zabbix packages provided. You can use these packages to install Zabbix.

Note:
OS distributions may lack the latest version of Zabbix in their repositiories.

From Zabbix official repository Zabbix SIA provides official RPM and DEB packages for Red Hat Enterprise Linux, Debian and Ubuntu LTS.

Package files are available at *repo.zabbix.com*. yum and apt repositories are also available on the server. A step-by-step tutorial for installing Zabbix from packages is provided here.

Red Hat Enterprise Linux / CentOS Supported for versions: RHEL 5, RHEL 6, RHEL 7, Oracle Linux 5, Oracle Linux 6, Oracle Linux 7, CentOS 5, CentOS 6, CentOS 7

Installing repository configuration package

Install the repository configuration package. This package contains yum configuration files.

Zabbix 2.4 for RHEL5, Oracle Linux 5, CentOS 5:

```
# rpm -ivh http://repo.zabbix.com/zabbix/2.4/rhel/5/x86_64/zabbix-release-2.4-1.el5.noarch.rpm
```

Zabbix 2.4 for RHEL6, Oracle Linux 6, CentOS 6:

```
# rpm -ivh http://repo.zabbix.com/zabbix/2.4/rhel/6/x86_64/zabbix-release-2.4-1.el6.noarch.rpm
```

Zabbix 2.4 for RHEL7, Oracle Linux 7, CentOS 7:

```
# rpm -ivh http://repo.zabbix.com/zabbix/2.4/rhel/7/x86_64/zabbix-release-2.4-1.el7.noarch.rpm
```

Installing Zabbix packages

Install Zabbix packages. Example for Zabbix server and web frontend with mysql database.

> **Note:**
> Zabbix official repository provides fping, iksemel, libssh2 packages as well. These packages are located in the *non-supported* directory.

```
# yum install zabbix-server-mysql zabbix-web-mysql
```

Example for installing Zabbix agent only.

```
# yum install zabbix-agent
```

Creating initial database

Create zabbix database and user on MySQL. See database creation scripts for MySQL.

Import initial schema and data.

```
# cd /usr/share/doc/zabbix-server-mysql-2.4.0/create
# mysql -uroot zabbix < schema.sql
# mysql -uroot zabbix < images.sql
# mysql -uroot zabbix < data.sql
```

Starting Zabbix server process

Edit database configuration in zabbix_server.conf

```
# vi /etc/zabbix/zabbix_server.conf
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=zabbix
```

Start Zabbix server process.

```
# service zabbix-server start
```

Editing PHP configuration for Zabbix frontend

Apache configuration file for Zabbix frontend is located in /etc/httpd/conf.d/zabbix.conf. Some PHP settings are already configured.

```
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 date.timezone Europe/Riga
```

It's necessary to uncomment the "date.timezone" setting and set the right timezone for you. After changing the configuration file restart the apache web server.

```
# service httpd restart
```

Zabbix frontend is available at http://zabbix-frontend-hostname/zabbix in the browser. Default username/password is Admin/zabbix.

Debian / Ubuntu Supported for version: Debian 7 (Wheezy), Ubuntu 14.04 LTS (Trusty Tahr)

Installing repository configuration package

Install the repository configuration package. This package contains apt configuration files.

Zabbix 2.4 for Debian 7:

```
# wget http://repo.zabbix.com/zabbix/2.4/debian/pool/main/z/zabbix-release/zabbix-release_2.4-1+wheezy_all
# dpkg -i zabbix-release_2.4-1+wheezy_all.deb
# apt-get update
```

Zabbix 2.4 for Ubuntu 14.04 LTS:

```
# wget http://repo.zabbix.com/zabbix/2.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_2.4-1+trusty_all
# dpkg -i zabbix-release_2.4-1+trusty_all.deb
# apt-get update
```

Installing Zabbix packages

Install Zabbix packages. dbconfig-common will create the database and populate the initial schema and data automatically. If backend db is located on a different server, please set dbc_remote_questions_default='true' in /etc/dbconfig-common/config.

Example for Zabbix server and web frontend with mysql database.

```
# apt-get install zabbix-server-mysql zabbix-frontend-php
```

> **Note:**
> The zabbix-frontend-php package, during installation, will configure a font, which is used on generated images. If you updated the package from any other repository and text is empty on graphs or maps, please check if a "ttf-dejavu-core" package is installed and try to execute "dpkg-reconfigure zabbix-frontend-php" command.

Example for installing Zabbix agent only.

```
# apt-get install zabbix-agent
```

Editing PHP configuration for Zabbix frontend

Apache configuration file for Zabbix frontend is located in /etc/apache2/conf.d/zabbix (For ubuntu 14.04, the file is located in /etc/apache2/conf-available/zabbix.conf). Some PHP settings are already configured.

```
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 date.timezone Europe/Riga
```

It's necessary to uncomment the "date.timezone" setting and set the correct timezone for you. After changing the configuration file restart the apache web server.

```
# service apache2 restart
```

Zabbix frontend is available at http://zabbix-frontend-hostname/zabbix in the browser. Default username/password is Admin/zabbix.

Troubleshooting See the section on installation-specific issue troubleshooting.

4 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-2.4.0.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 Linux systems, run:

```
groupadd zabbix
useradd -g zabbix 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
```

Note:
--with-libxml2 configuration option is required for virtual machine monitoring, supported since Zabbix 2.2.0.

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
```

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.

Attention:
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.

Note:
Command-line utilities zabbix_get and zabbix_sender are compiled if --enable-agent option is used.

Note:
Use --with-ibm-db2 flag to specify location of the CLI API.
Use --with-oracle flag to specify location of the OCI API.

5 Make and install everything

Note:
If installing from SVN, 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).

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

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

- if you have installed a Zabbix proxy, edit the proxy configuration file **/usr/local/etc/zabbix_proxy.conf**

You must specify the server IP address and proxy hostname (must be known to the server), as well as the database name, user and password (if using any).

> **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.
> See the "See also" section at the bottom of this page to find out how to configure shared memory.

Run zabbix_agentd on all the monitored machines.

```
shell> zabbix_agentd
```

> **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 frontends/php 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 (Fedora, RHEL, CentOS)
- /var/www (Debian, Ubuntu)

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 frontends/php
cp -a . <htdocs>/zabbix
```

If installing from SVN and planning to use any other language than English, you must generate translation files. To do so, run:

```
locale/make_mo.sh
```

`msgfmt` utility from gettext package is required.

Note:
Additionally, to use any other language than English, its locale should be installed on the web server. See the "See also" section in the "User profile" page to find out how to install it if required.

Installing frontend

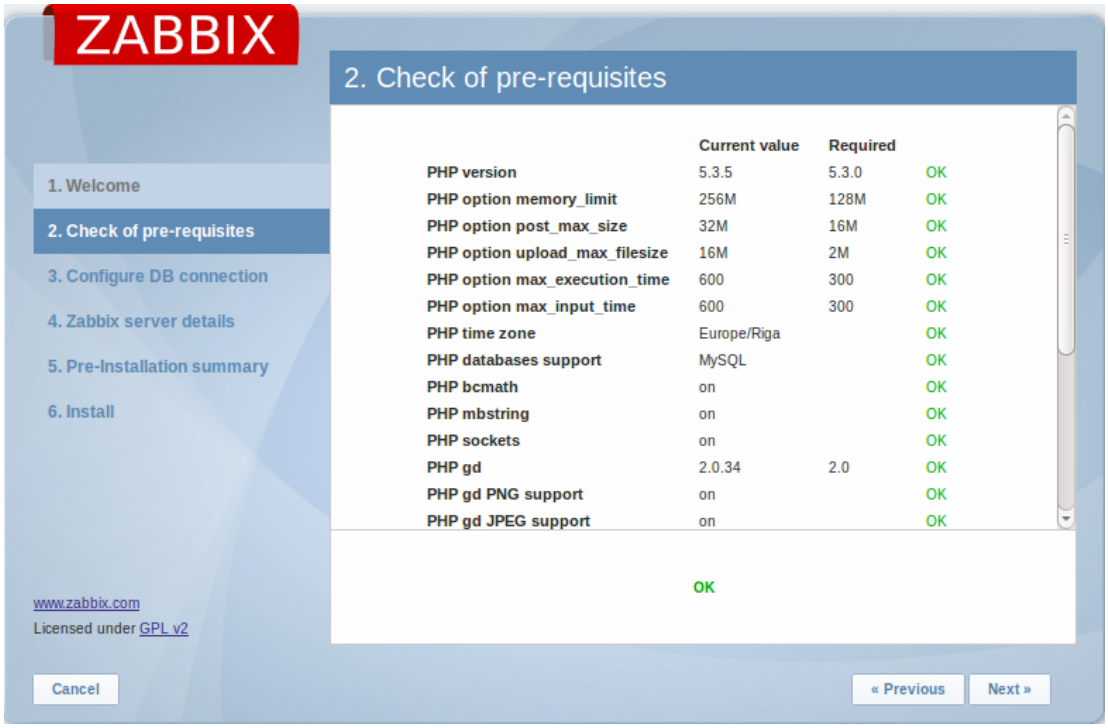
Step 1

In your browser, open Zabbix URL: http://<server_ip_or_name>/zabbix

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

Step 2

Make sure that all software prerequisites are met.



Pre-requisite	Minimum value	Description
PHP version	5.3.0	
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	In php.ini: max_execution_time = 300
PHP max_input_time option	300 seconds	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: IBM DB2, MySQL, Oracle, PostgreSQL, SQLite	One of the following modules must be installed: ibm_db2, mysql, oci8, pgsql, sqlite3
bcmath		php-bcmath
mbstring		php-mbstring
PHP mbstring.func_overload option	must be disabled	In php.ini: mbstring.func_overload = 0.
PHP always_populate_raw_post_data option	must be disabled	Required only for PHP versions 5.6.0 or newer. In php.ini: always_populate_raw_post_data = -1
sockets		php-net-socket. Required for user script support.
gd	2.0 or higher	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 or php5-dom
xmlwriter		php-xmlwriter
xmlreader		php-xmlreader
ctype		php-ctype
session		php-session

Pre-requisite	Minimum value	Description
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.

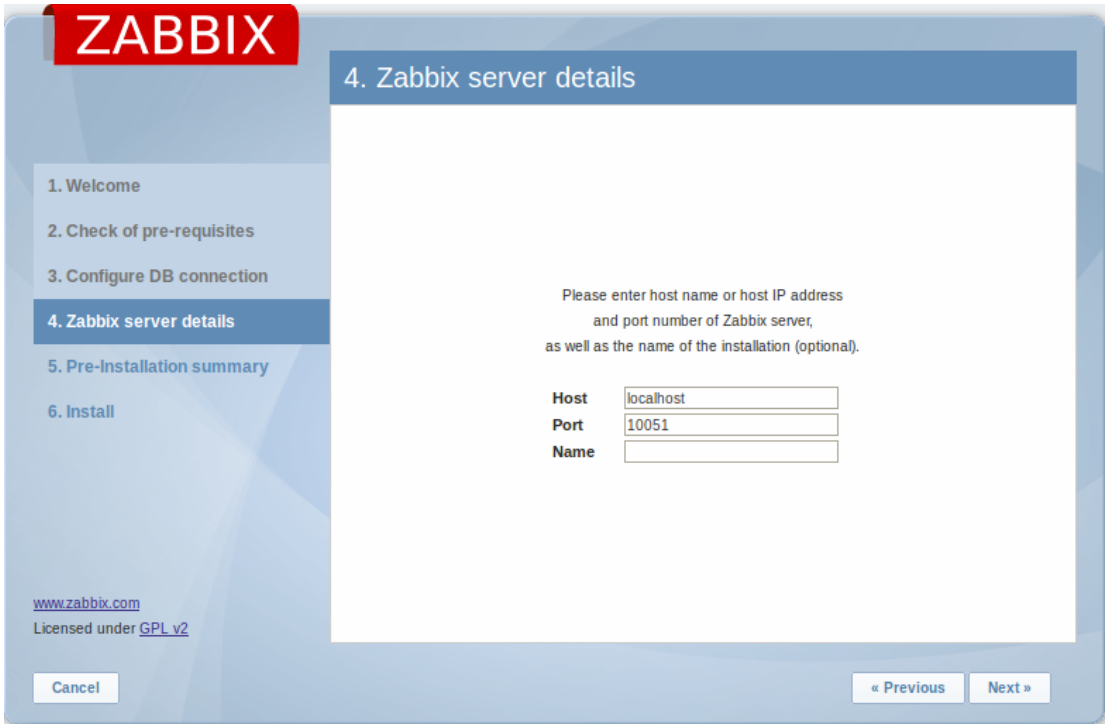
Attention:
If there is a need to change the Apache user or user group, permissions to the session folder must be verified. Otherwise Zabbix setup may be unable to continue.

Step 3

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

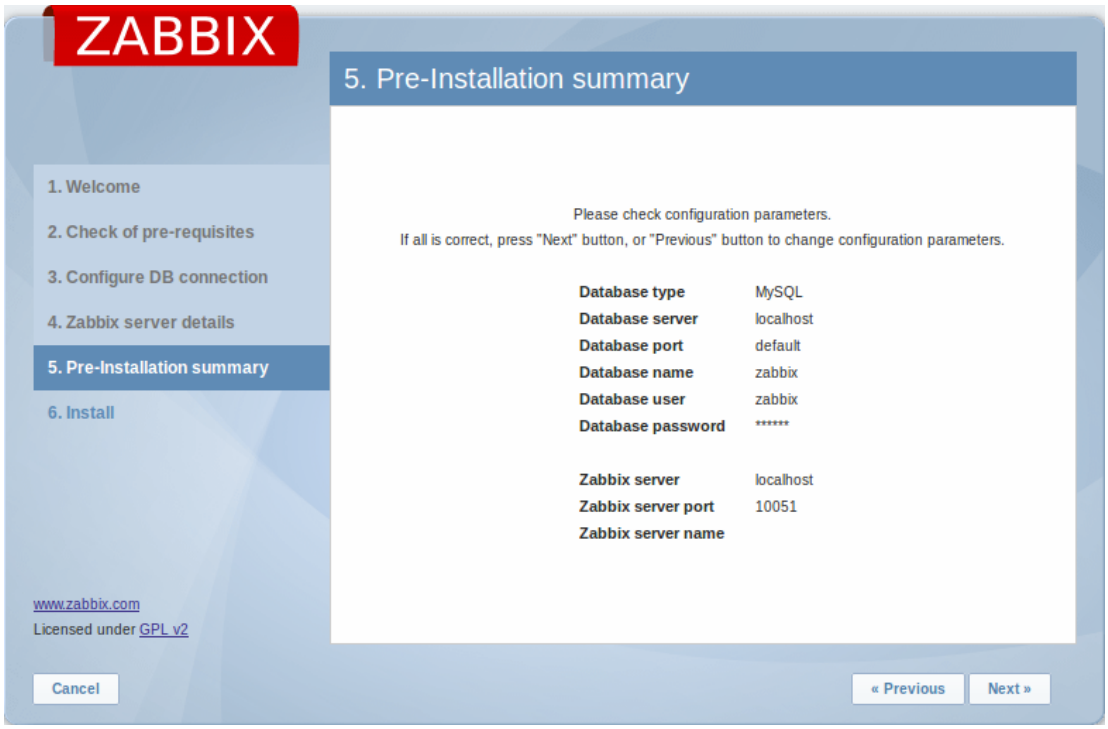
Step 4

Enter Zabbix server details.



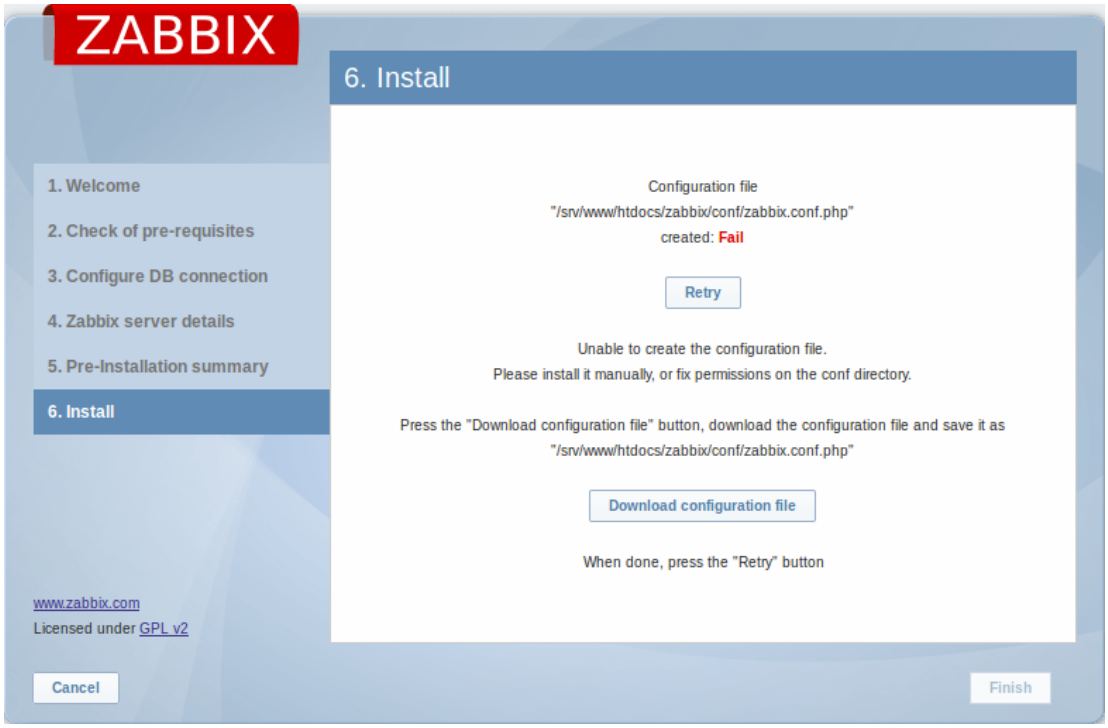
Step 5

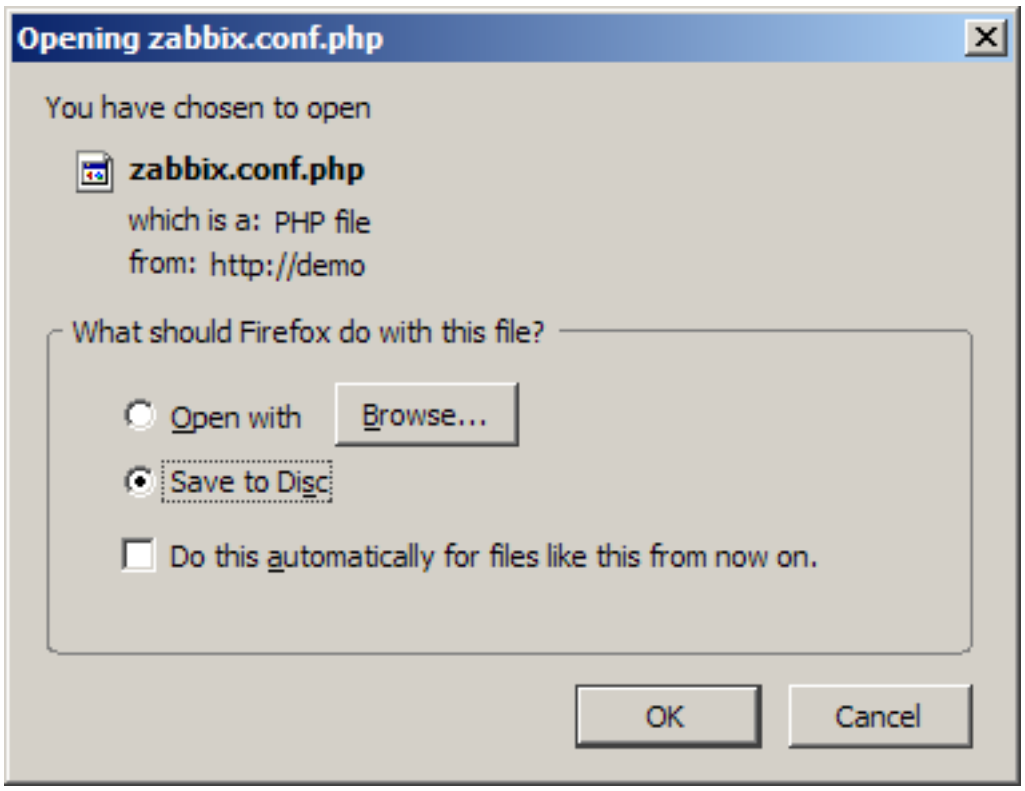
Review a summary of settings.



Step 6

Download the configuration file and place it under conf/.

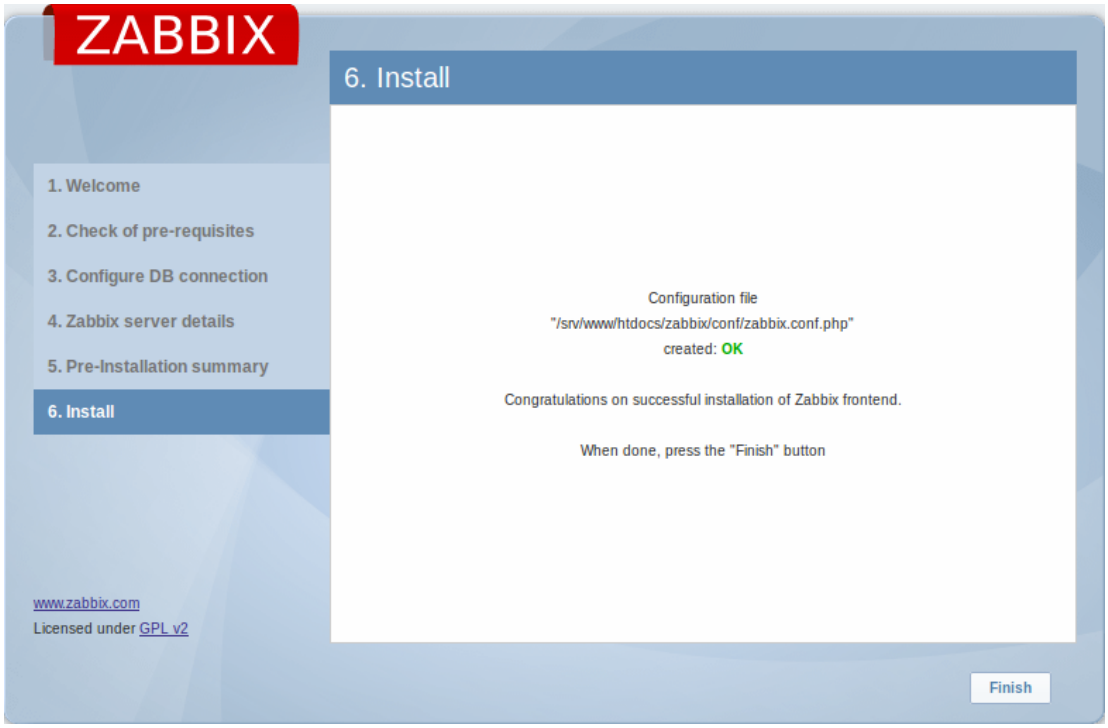




Note:
Providing the webserver user has write access to conf/ directory the configuration file would be saved automatically and it would be possible to proceed to the next step right away.

Step 7

Finish the installation.



Step 8

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

Proceed to getting started with Zabbix.

Troubleshooting See the section on installation-specific issue troubleshooting.

See also

1. How to configure shared memory for Zabbix daemons

5 Upgrade procedure

Overview

This section provides the steps required for a successful upgrade from Zabbix 2.2.x to 2.4.

Database upgrade to version 2.4 may take a long time.

> **Attention:**
> Make sure to read upgrade notes before proceeding with the upgrade.

1 Stop Zabbix 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 PHP files.

4 Install new server binaries

You may use pre-compiled binaries or compile your own.

5 Review server configuration parameters

Some parameters of zabbix_server.conf might have changed from 2.2, new parameters added. You may want to review them.

> **Attention:**
> Housekeeper is disabled after upgrading to Zabbix 2.4 from versions before 2.2. The desired housekeeper functionality should be checked and enabled manually in Administration → General → Housekeeper, if necessary.

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.

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.

Zabbix server will automatically upgrade the database only from Zabbix 2.0.x to 2.4. For upgrading from earlier versions consult Zabbix documentation for 2.0 and earlier.

7 Install new Zabbix web interface

Follow installation instructions.

6 Known issues

Problems with pressing Enter in configuration forms

Pressing Enter in a text field of a configuration form is known to result in various problems.

For instance, if you open the configuration form of a host with linked templates, then press Enter in any text field and update the form, template linkage is removed (items from the template remain).

IPMI checks

IPMI checks will not work with the standard OpenIPMI library from Debian/Ubuntu package. To fix, 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.

ODBC checks

Zabbix server or proxy that uses MySQL as its database may or may not work correctly with MySQL ODBC library due to an upstream bug. Please see ZBX-7665 for more information and available workarounds.

HTTPS checks

Web scenarios using the https protocol and Zabbix agent checks `net.tcp.service[https...]` and `net.tcp.service.perf[https...]` may fail if the target server is configured to disallow TLS v1.0 protocol or below. Please see ZBX-9879 for more information and available workarounds.

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.

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.

Web monitoring

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. Please see ZBX-10486 for more information and available workarounds.

Slow MySQL queries

Zabbix server generates slow select queries in case of non-existing values for items. This is caused by a known issue in MySQL 5.6/5.7 versions. A workaround to this is disabling the index_condition_pushdown optimizer in MySQL. For an extended discussion, see ZBX-10652.

7 Template changes

This page lists all changes to the stock templates that are shipped with Zabbix. It is suggested to modify these templates in existing installations - depending on the changes, it can be done either by importing the latest version or by performing the change manually.

Template changes in 2.4.0

Due to removal of node-based distributed monitoring, several changes have been made to *Template App Zabbix Server*:

- *Zabbix busy node watcher processes, in %* internal item has been removed;
- *Zabbix node watcher processes more than 75% busy* trigger has been removed;
- *Zabbix busy node watcher processes, in %* internal item has been removed from *Zabbix internal process busy %* graph.

Template changes in 2.4.4

Disk device discovery transfer rate item prototype names now correctly identify item value as bytes per second rather than kilobytes per second in *Template Virt VMware Guest*. The affected items are `vmware.vm.vfs.dev.read[{$URL},{HOST.HOST},{#DISKNAME},bps` and `vmware.vm.vfs.dev.write[{$URL},{HOST.HOST},{#DISKNAME},bps]`.

Template changes in 2.4.5

Value type was changed from "Numeric (unsigned)" to "Numeric (float)" for items `system.stat[kthr,b]` and `system.stat[kthr,r]` in *Template OS AIX*. Both items were also added to "Performance" application.

Template changes in 2.4.7

Item **vm.memory.size[total]** moved from "Filesystems" to "Memory" application in *Template OS Windows*.

8 Upgrade notes for 2.4.0

Node-based distributed monitoring removed

In Zabbix 2.4.0 the support of node-based distributed monitoring has been discontinued due to existing limitations and potential problems with reliability in large environments. While a new version of distributed monitoring may be considered and developed in the future, for the time being using proxies becomes the main distributed monitoring option in Zabbix.

For the remaining node-based DM users, during an upgrade to Zabbix 2.4.0, each upgraded node will be switched to a standalone Zabbix server keeping all configuration and history information from the local node and from the child nodes as well.

To maintain uniqueness of data from non-local nodes, all the fields will be prefixes with N<nodeid>_. If the length of the new value exceeds max field size, it will be trimmed. Highly unlikely, but if the new value already exists in the database, the situation will be caught by a check on duplicates. Global macros will be processed in a special way by adding N<nodeid>_ after the dollar sign, for example, {$N123_MACRO}.

Support of the "node watcher" second parameter has been removed from the `zabbix[process]` internal item. This item has also been removed from *Template App Zabbix Server*. In order to fix it, either manually delete the item or import the new template and mark *Delete missing* option for items.

Case-sensitive MySQL database

A case-sensitive MySQL database is required for proper server work. It is recommended to create a case-sensitive MySQL database during new installations. If you created a MySQL database with the utf8 character set previously, in order to support case sensitiveness of stored data, you need to convert the charset to utf8_bin.

Handling automatically discovered hosts

Hosts created by a network discovery rule will be deleted automatically if a discovered entity is not in the rule's IP range anymore. Hosts are deleted immediately.

Internal items always processed

Internal items previously were not processed if a host went into maintenance with 'no data collection'. Now internal items are always processed by server or proxy regardless of host maintenance status.

Item changes

`zabbix[proxy,<proxyname>,"lastaccess"]` internal item is now always processed by Zabbix server regardless of host location (on server or proxy).

JSON formatting

The formatting of JSON objects (with tabs and new lines) has been removed, which allows the traffic to be reduced by 20-30% when data is sent between Zabbix services. Additionally, the escaping of forward slash or solidus (/) has also been removed. JSON, formatted as in the previous versions of Zabbix, would still be accepted as valid.

Log file handling

Now, at DebugLevel=0 only basic information is printed to the logfile/syslog. For agents it is Zabbix version, used config file, started/stopped Zabbix processes. For proxy and server - Zabbix version, enabled features, used config file, DB version, started/stopped Zabbix processes. The logging for Zabbix agent is also consistent on Unix and Windows machines now.

Daemon changes

Disabled hosts, items and triggers are stored in configuration cache now. Adjusting **CacheSize** configuration parameter might be needed due to increased memory usage.

Server and proxy now refuse to start if **StartPollersUnreachable** configuration parameter is 0, but regular, IPMI or Java pollers are started. Otherwise, hosts that become unreachable would never be checked again.

Proxy now refuses to start with a server database. In some older versions a proxy could be started with a server database (e.g. by mistake) and somehow it worked. After upgrading to Zabbix 2.4 this is not the case anymore. Follow instructions to create a new, valid proxy database, if necessary. A similar check has been added to server to prevent starting it with a proxy database.

Support for PHP mutexes has been removed on the server side due to licensing issues. While it was not recommended to use Zabbix server and frontend with SQLite3 database before, this change makes it even less recommended, because simultaneous database access with Zabbix server and frontend may now corrupt the database. Note that using Zabbix proxy with SQLite3 database is still a perfectly valid solution.

Carriage return symbols (CR) are not stripped from notification messages anymore. Any custom scripts used to process action messages must be updated to support the new newline format (CRLF).

Housekeeper changes

On `zabbix_server` and `zabbix_proxy` startup housekeeping is postponed for 30 minutes instead of running at once. This will lower startup load for both these processes.

Script result changes

User parameters, global scripts, system.run[] items and external checks will now return standard error along with standard output. In the corner case of standard error as numeric value a numeric item will change state from NOTSUPPORTED to normal.
Note: Windows agent has this functionality since 1.8.5

Empty result is now allowed for `system.run[]`. `system.run` item configured with textual value type (character, log or text) will change state from unsupported to normal if the command returns empty result.

Changes regarding multibyte string support

Zabbix frontend now displays fatal error if mbstring PHP functions are not enabled. There are no string wrapper functions anymore; they have been replaced by string and/or mbstring functions accordingly. Additionally, the `mbstring.func_overload` option from php.ini is required to be off, and a warning will be displayed in *Status of Zabbix* if it is enabled.

Unified date and time format

Date and time format has been unified across Zabbix frontend according to the ISO 8601 standard of **YYYY-MM-DD hh:mm:ss**. The new format is implemented everywhere except graphs.

Changes in latest data page

The latest data section no longer has the host and host group selection dropdowns. Instead, those two choices can be made within an expanded filter section, which can be used flexibly for displaying the exact set of items you want.

Dropping green status icons in item list

Previously, green status icons were displayed in the last *Error* column of the item list, for error-free items, which could be misunderstood as if, on the contrary, the items had errors. Now the green icons are displayed no more and, additionally, the column is renamed to *Info*. For items with problems, as before, a red square icon is displayed in this column.

The same change is applied to the trigger list as well.

Graph palette changes

The default colour palette for graphs has been changed to use green colour first instead of red. Red will be used second. The bar report palette nr. 4 has also been slightly changed. This won't change existing custom graphs, but the colours will be different when creating new custom graphs.

Converting triggers to the new syntax

The database upgrade patches will automatically convert the existing triggers to the new syntax.

Note that along with the syntax change, unary **-** operator associativity has changed from right-associative to non-associative, meaning that **-(-1)** should now be used instead of **--1**.

Dropped support of round-off constants

The descriptions of ZBX_UNITS_ROUNDOFF_THRESHOLD, ZBX_UNITS_ROUNDOFF_UPPER_LIMIT, ZBX_UNITS_ROUNDOFF_MIDDLE_LIMIT and ZBX_UNITS_ROUNDOFF_LOWER_LIMIT definitions have been removed from the documentation since their functionality no longer matches their intended purpose. They are still present in the code and any changes made to them will remain, but their modification may cause unexpected results.

9 Upgrade notes for 2.4.1

Bug in converting triggers and calculated items to the new syntax

The database upgrade patches in 2.4.0 automatically converted the existing triggers and calculated items to the new syntax.

However, these patches had a bug, namely replacing "#" in low-level discovery macros with "<>". Thus, `{#MACRO}` would be replaced with `{<>MACRO}`.

This only affected macros which were an independent part of the expression, not part of the key. For instance, the expession `{host:snmp.oid[{#SNMPINDEX}].last()} # {#SNMPVALUE}` would be transformed into `{host:snmp.oid[{#SNMPINDEX}].last()} <> {<>SNMPVALUE}`.

In 2.4.1, the database upgrade patches have been fixed, so an upgrade from 2.2 to 2.4.1 should now work fine. However, they do not fix expressions that have already been broken by the upgrade from 2.2 to 2.4.0.

If you wish to fix these broken expressions, you can start by finding which expressions were affected by using the following queries:

```
select triggerid, expression from triggers where expression like '%{<>%';
select itemid, key_, params from items where type=15 and params like '%{<>%';
```

You should then replace "{<>" in these expressions with "{#".

10 Upgrade notes for 2.4.2

PHP 5.6

If your system has PHP verion 5.6 or later, a *php.ini* core directive **always_populate_raw_post_data** must be set to **-1**.

Daemon changes

SNMP polling logic has been changed to always retry at least once. This should reduce the number of network errors, and might affect poller and network load.

With validation of SNMP responses in place, bad single-variable responses with mismatching OIDs are not accepted by Zabbix server and proxy, and will make related SNMP items go not supported. This makes it impossible to monitor very non-conformant SNMP devices. This has later been fixed in Zabbix 2.4.3.

11 Upgrade notes for 2.4.3

Daemon changes

Strict validation of SNMP responses has been turned off for single-variable SNMP requests. Items on misbehaving devices will now be monitored normally, but messages about such responses will be logged at DebugLevel=4.

If an IPMI device reports a threshold sensor and a discrete sensor under the same name, the threshold sensor is now preferred. This might fix strange readings (like "1" for fan RPM) or "not supported" errors.

Frontend changes

History related macros - {ITEM.VALUE}, {ITEM.LASTVALUE} and the {host:key.last()} functional macro - now obey the ZBX_HISTORY_PERIOD constant. This limits the amount of data the macro has to sift through and results in better performance.

12 Upgrade notes for 2.4.4

Daemon changes

Previously, if Zabbix could not send ICMP ping packets to a particular host, all ICMP ping items would attain a value of 0 in some cases. Now, they always become unsupported.

Monitoring of Windows processes has been improved. After upgrading to 2.4.4, Zabbix agent may report a different number of processes than before when using the **proc.num** item, e.g.:

Before:`c:\> zabbix_agentd.exe -c \zabbix_agentd.conf -t proc.num[zabbix_agentd.exe] proc.num[zabbix_agentd.exe,,,,u|1]`

After:`c:\> zabbix_agentd.exe -c \zabbix_agentd.conf -t proc.num[zabbix_agentd.exe] proc.num[zabbix_agentd.exe,,,,u|4]`

The `SourceIP` configuration parameter is now respected in web monitoring, VMware monitoring, Ez Texting alerts and **net.tcp.service{.perf}[https]** items. This may lead to checks being done from a different IP address than before the upgrade.

Validation of global regular expressions in LLD rules

A check for valid reference has been added for global regular expressions in LLD rules. If entered reference is not valid, due to misspelling or missing referenced global regular expression, the respective LLD rule will become unsupported and appropriate error message will be displayed.

VMware monitoring changes

VMware performance collector based statistics retrieval was separated from VMware data retrieval. Therefore it is recommended to enable more collectors than monitored VMware services (`StartVMwareCollectors=<N>`). Otherwise retrieval of VMware performance collector based statistics might be delayed by retrieval of VMware configuration data (which takes a while for large installations).

A new configuration option `VMwarePerfFrequency` was added to configure statistics data retrieval period.

The *bps* mode value of the following items are now correctly reported in *bytes per second* instead of *kilobytes per second* as before:

- `vmware.hv.network.in`
- `vmware.hv.network.out`
- `vmware.vm.net.if.in`
- `vmware.vm.net.if.out`
- `vmware.vm.vfs.dev.read`
- `vmware.vm.vfs.dev.write`

Please see VMware configuration parameters description for more details on how to configure Zabbix server/proxy for VMware monitoring.

Template changes

Disk device discovery transfer rate item prototype names were fixed for *Template Virt VMware Guest*. The hypervisor network interface, virtual machine network interface and virtual machine disk device transfer rates were incorrectly reported in kilobytes rather than bytes. Now they will be correctly reported in bytes per second.

In order to fix it, import this template from https://www.zabbix.org/wiki/Zabbix_Templates/Official_Templates.

13 Upgrade notes for 2.4.5

Daemon changes

Template Changes

Value type was changed from ”Numeric (unsigned)” to ”Numeric (float)” for items **system.stat[kthr,b]** and **system.stat[kthr,r]** in *Template OS AIX*. Both items were also added to ”Performance” application.

In order to fix it, import this template from https://www.zabbix.org/wiki/Zabbix_Templates/Official_Templates.

14 Upgrade notes for 2.4.6

This minor version does not have any upgrade notes.

15 Upgrade notes for 2.4.7

Daemon changes

The default error message set for unsupported items was changed from ”Item is not supported.” and ”ZBX_NOTSUPPORTED” to ”Unknown error.”. This message is used when item check fails without setting error message.

Monitoring of Windows protected processes was improved. Therefore in some cases on Windows (2008 Server and later) **proc.num** may return more found processes than previously.

Zabbix now tries to differentiate item timeouts from host timeouts. If another item check was successful between two failed checks of a problematic item, then the problematic item is marked as not supported after the second failed check without affecting host availability.

Template changes

Item **vm.memory.size[total]** moved from ”Filesystems” to ”Memory” application in *Template OS Windows*.

In order to fix it, import this template from https://www.zabbix.org/wiki/Zabbix_Templates/Official_Templates.

16 Upgrade notes for 2.4.8

Item changes

Correct resolution of low-level discovery macros has been improved in calculated item formulas. Function parameters now will be quoted if, after resolving low-level discovery macros, they contain `,`, `)` characters or start with `"`, `<space>` characters.

Dashboard host status widget

Previously, when using the dashboard filter *Unacknowledged only* option, acknowledged problem triggers were displayed neither in *With problems* nor *Without problems* columns of the host status widget, resulting in a wrong host count in total. Now the acknowledged problem triggers are displayed in the *Without problems* column.

Daemon changes

The detection of a single item failing with network/timeout error introduced in Zabbix 2.4.7 was removed because of inability to distinguish possible network errors.

4. Quickstart

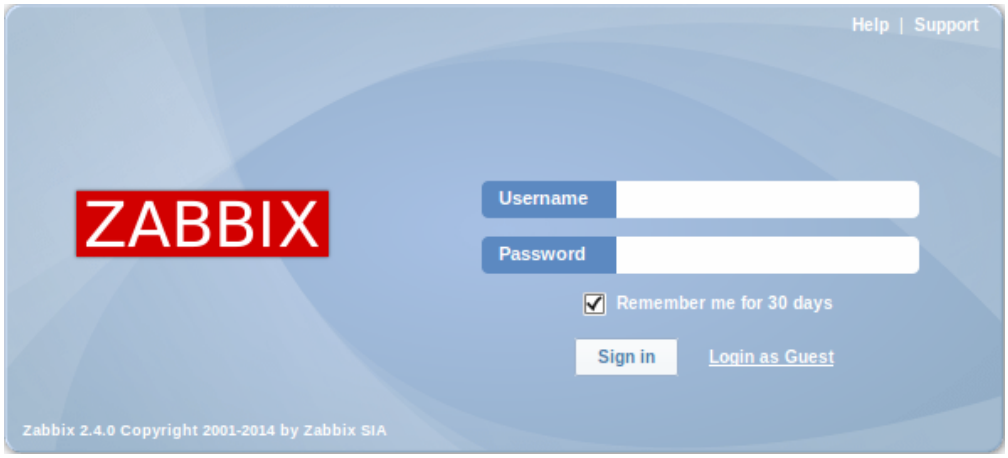
Please use the sidebar to access content in the Quickstart section.

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.

When logged in, you will see 'Connected as Admin' in the lower right corner of the page. 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* and select *Users* in the dropdown.

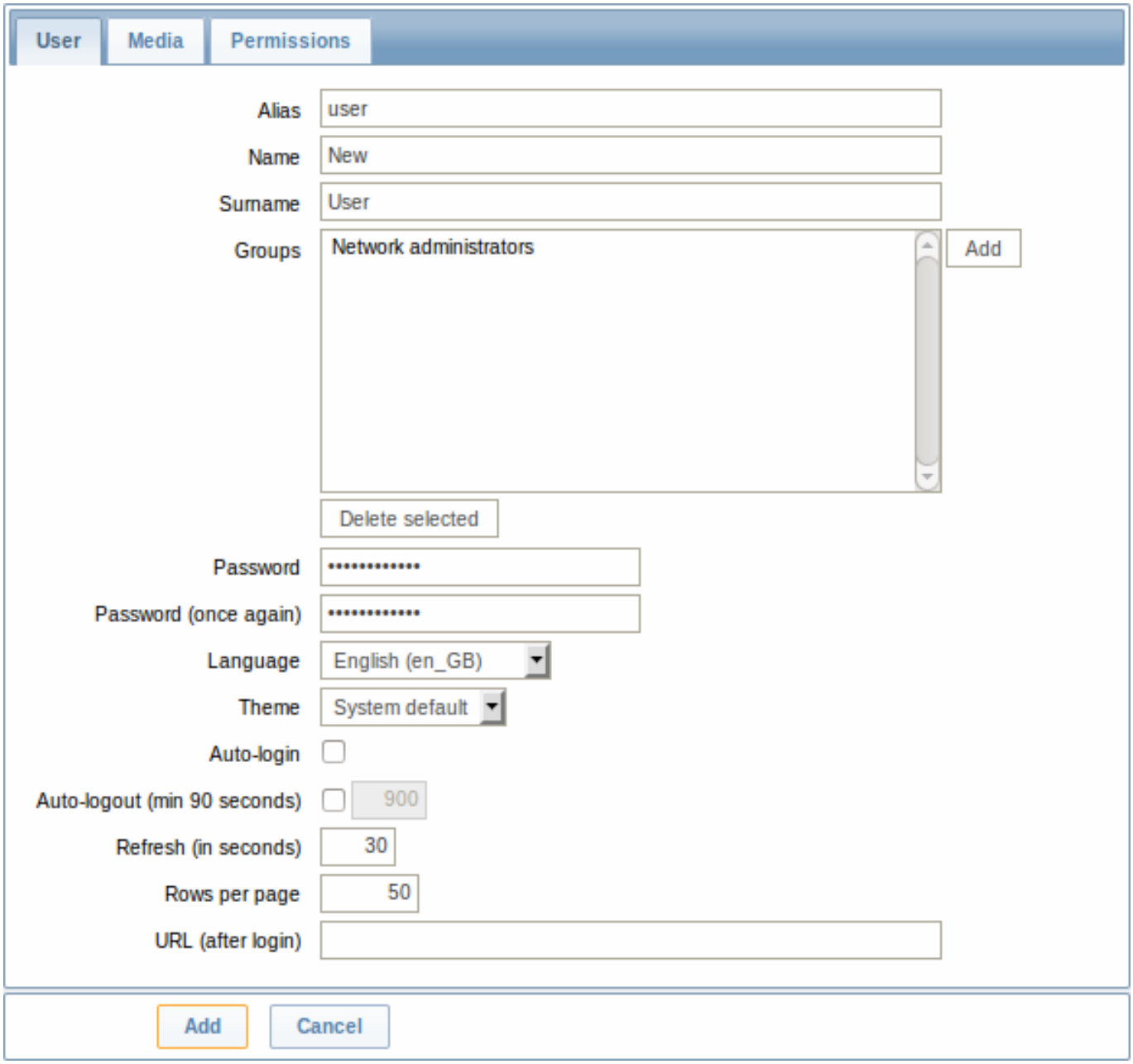
Alias	Name	Surname	User type	Groups	Is online?	Login	Frontend access	Debug mode	Status
Admin	Name	Surname	Zabbix Super Admin	Zabbix administrators	Yes (2015-01-12 07:16:21)	Ok	System default	Disabled	Enabled
guest			Zabbix User	Guests	Yes (2015-01-12 07:15:48)	Ok	System default	Disabled	Enabled

Initially there are only two users defined in Zabbix.

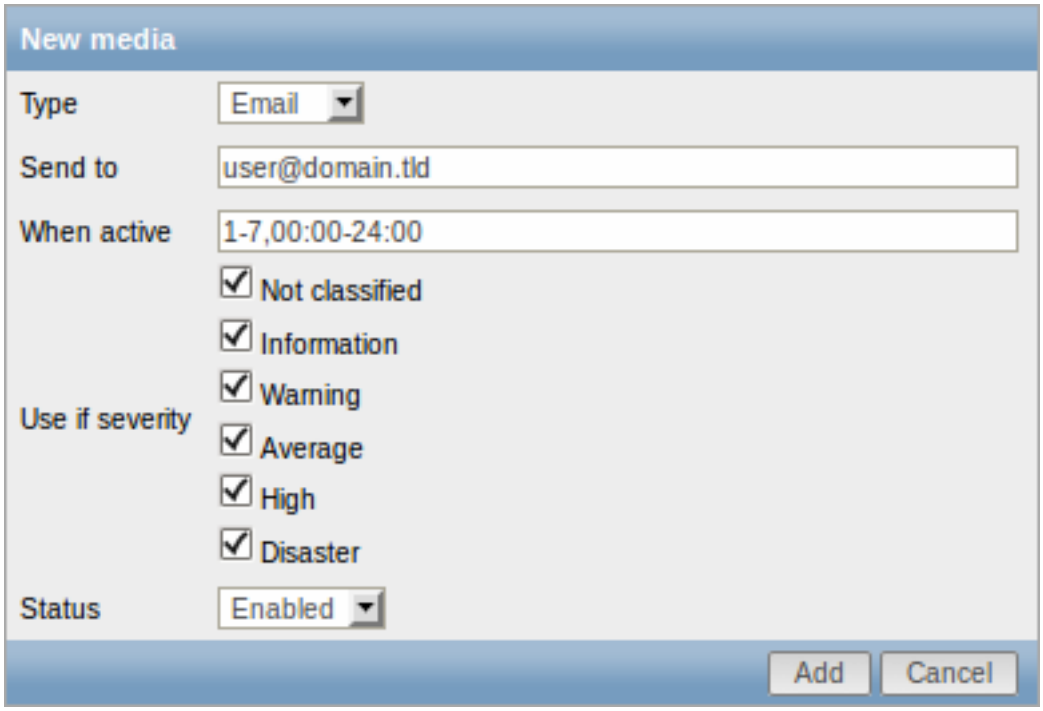
- 'Admin' user is a Zabbix superuser, which has full permissions.
- 'Guest' user is a special default user. If you are not logged in, you are accessing Zabbix with "guest" permissions. By default, "guest" has no permissions on Zabbix objects.

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 'Network administrators'.



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 customise trigger severity levels for which the medium will be active, but leave all of them enabled for now.

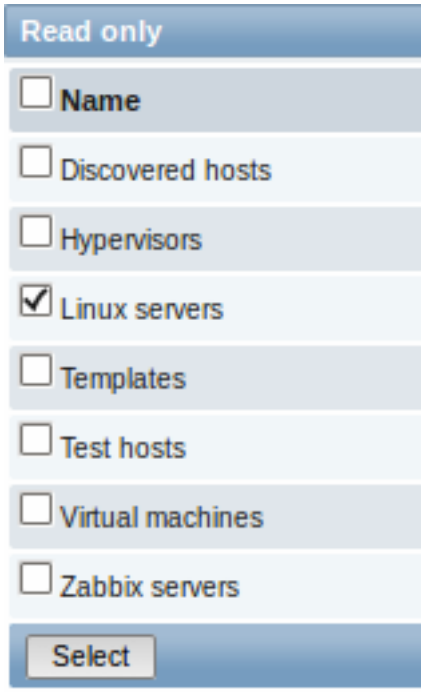
Click on *Add*, then click *Add* in the user properties form. The new user appears in the userlist.

Alias	Name	Surname	User type	Groups	Is online?	Login	Frontend access	Debug mode	Status
Admin	Name	Surname	Zabbix Super Admin	Zabbix administrators	Yes (2015-01-12 07:23:29)	Ok	System default	Disabled	Enabled
guest			Zabbix User	Guests	Yes (2015-01-12 07:22:57)	Ok	System default	Disabled	Enabled
user	New	User	Zabbix User	Network administrators	No	Ok	System default	Disabled	Enabled

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 - 'Network 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 *Add* below the 'Read only' listbox.



In this pop-up, mark the checkbox next to 'Linux servers', then click *Select*. *Linux servers* should be displayed in the respective box. 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

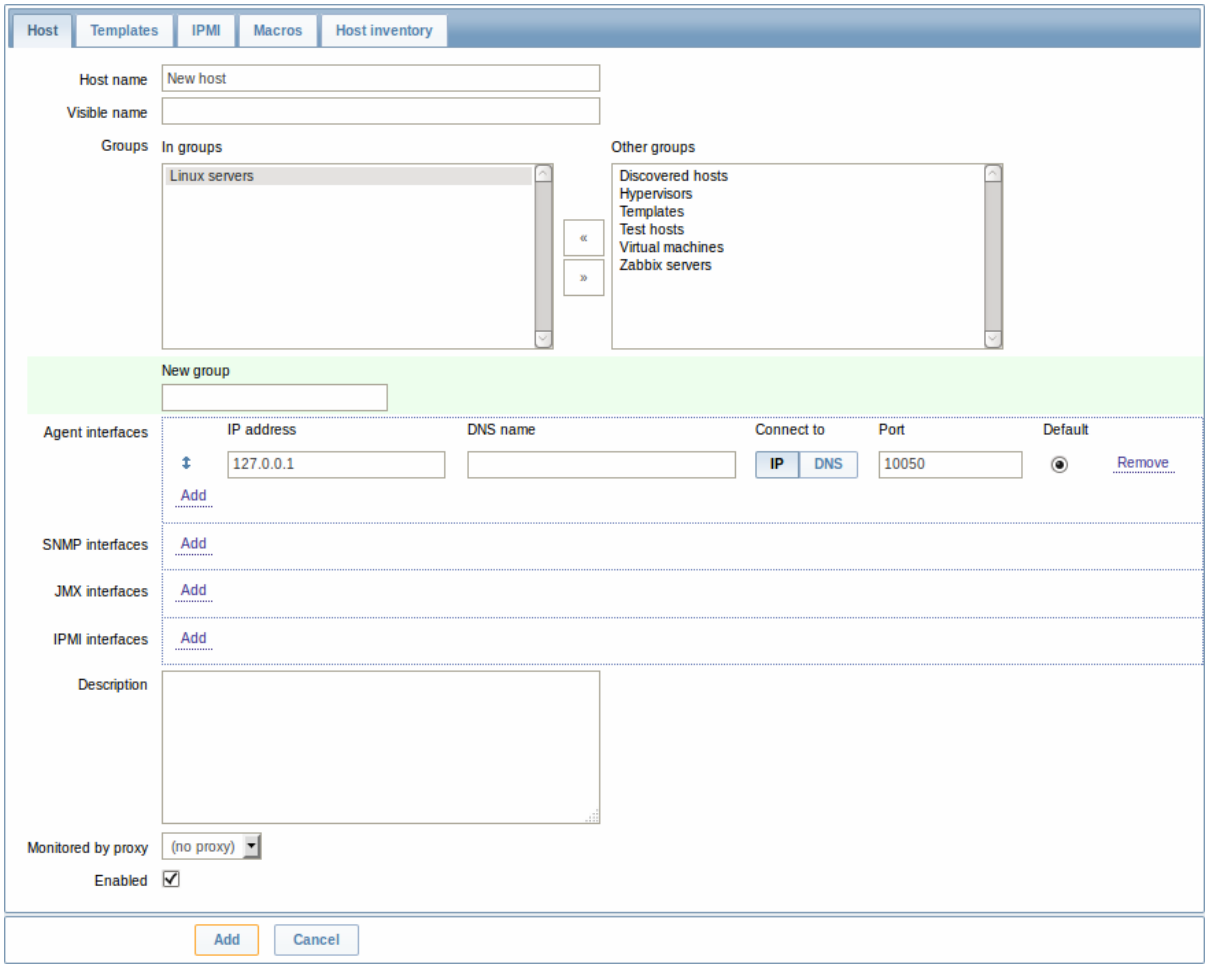
In this section you will learn how to set up a new host.

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*. This will present us with a host configuration form.



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 groups from the right hand side selectbox and click on « to move them to the 'In groups' selectbox.

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 hostlist.

Note:
If the *Z* icon in the *Availability* column is red, there is some error with communication - move your mouse cursor over it to see the error message. If that icon is gray, no status update has happened so far. Check that Zabbix server is running, and try refreshing the page later as well.

3 New item

Overview

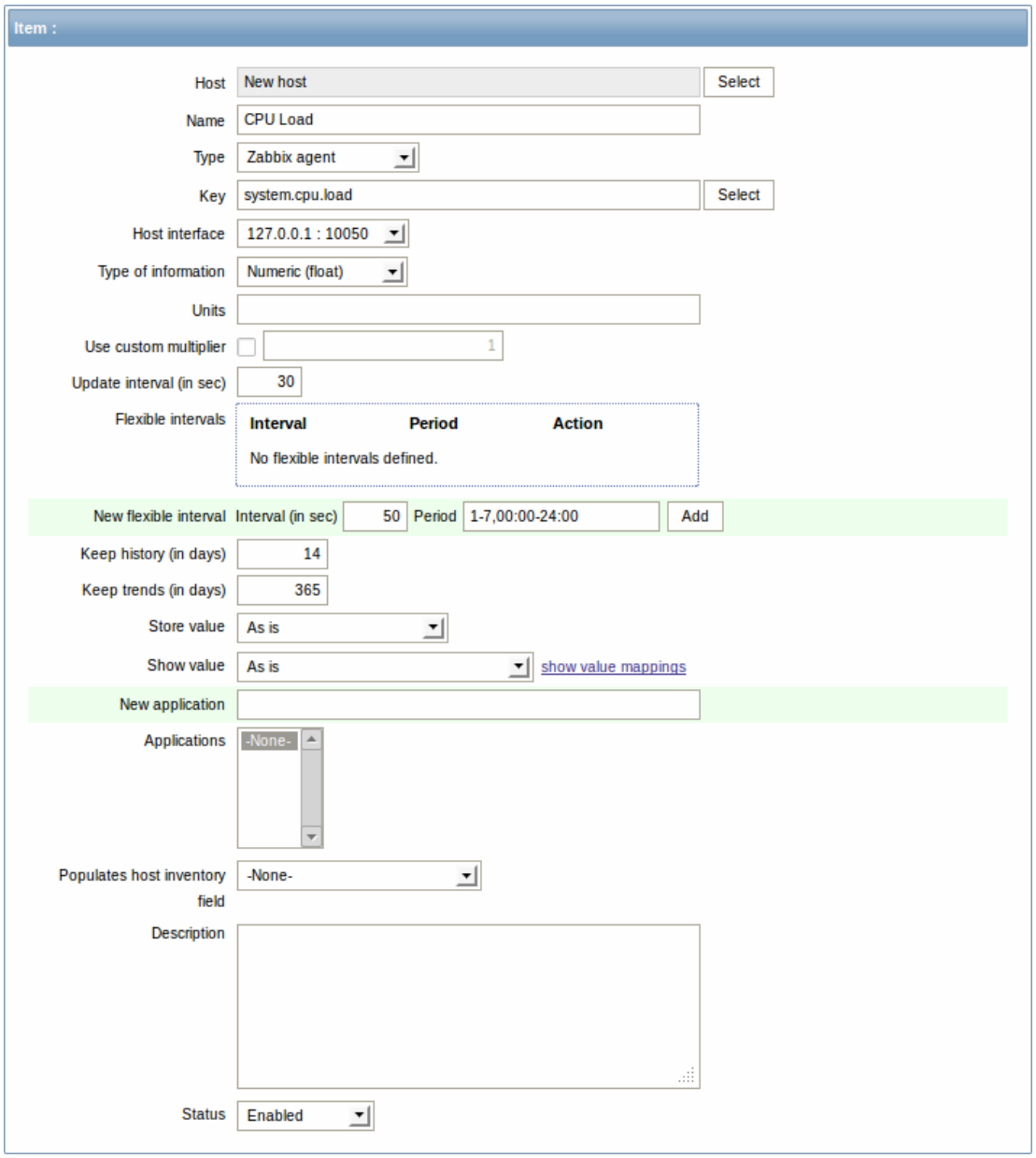
In this section you will learn how to set up an item.

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.

The *Items* link in the row of 'New host' should display a count of '0'. Click on the link, and then click on *Create item*. This will present us with an item definition form.



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

- 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.

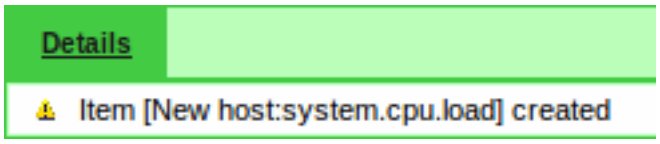
Type of information

- 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.

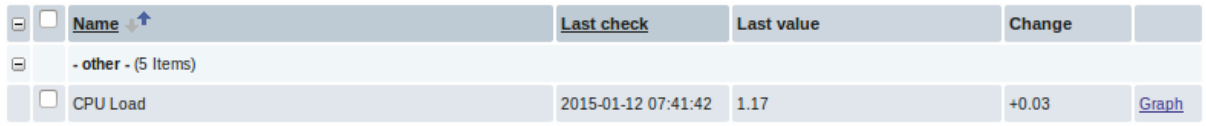
Other options will suit us with their defaults for now.

When done, click *Add*. The new item should appear in the itemlist. Click on *Details* above the list to view what exactly was done.



Seeing data

With an item defined, you might be curious if it is actually gathering data. For that, go to *Monitoring* → *Latest data*, 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.

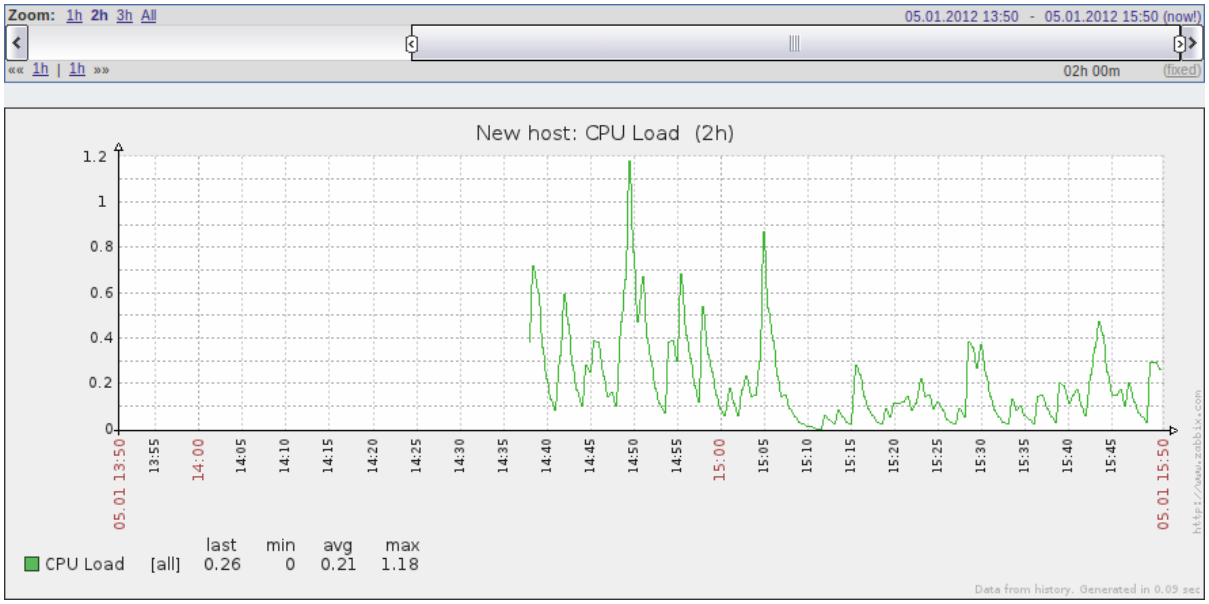
If you do not see information about the item as in the screenshot, make sure that:

- 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

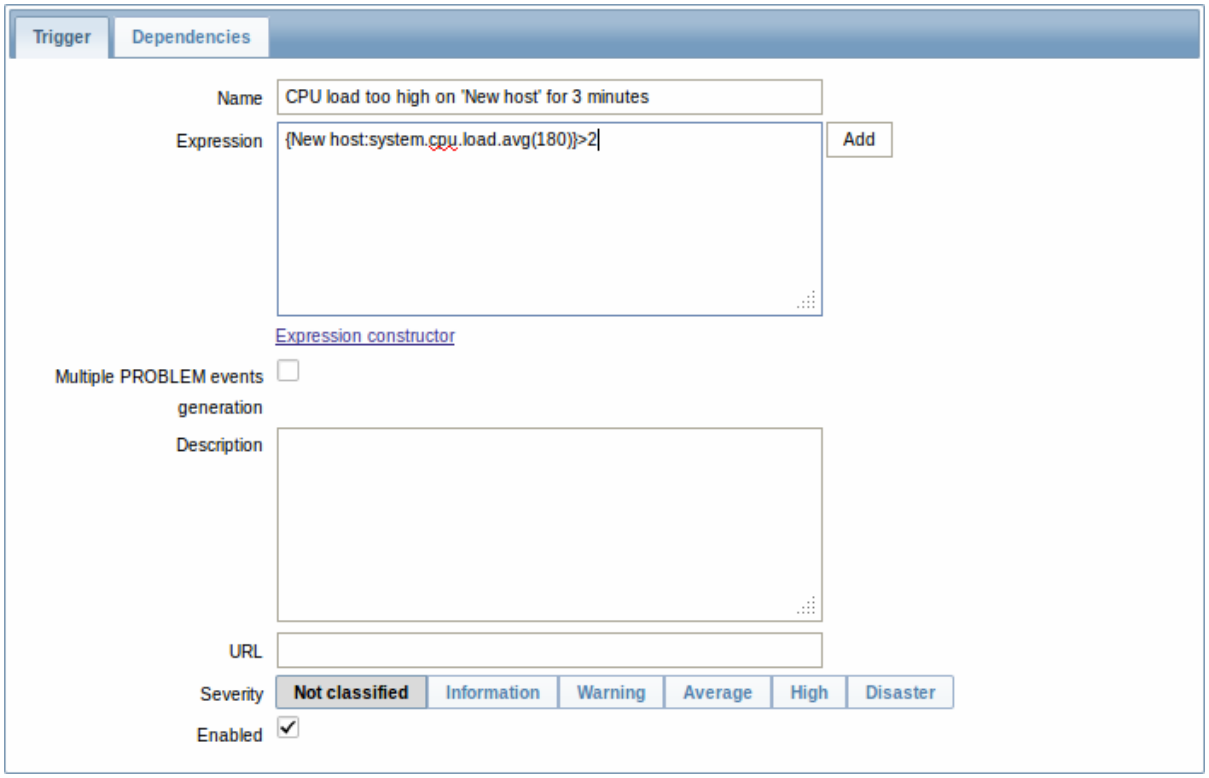
In this section you will learn how to set up a trigger.

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(180)}>2

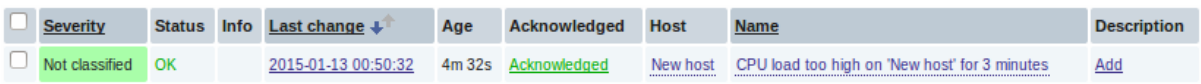
This is the trigger expression. Make sure that the expression is entered right, down to the last symbol. The item key here (system.cpu.load) is used to refer to the item. This particular expression basically says that the problem threshold is exceeded when the CPU load average value for 3 minutes is 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.

For that, go to *Monitoring* → *Triggers*. After 3 minutes or so (we asked to evaluate a 3-minute average after all) your trigger should appear there, presumably with a green 'OK' flashing in the 'Status' column.



The flashing indicates a recent change of trigger status, one that has taken place in the last 30 minutes.

If a red 'PROBLEM' is flashing there, then obviously the CPU load has exceeded the threshold level you defined in the trigger.

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.

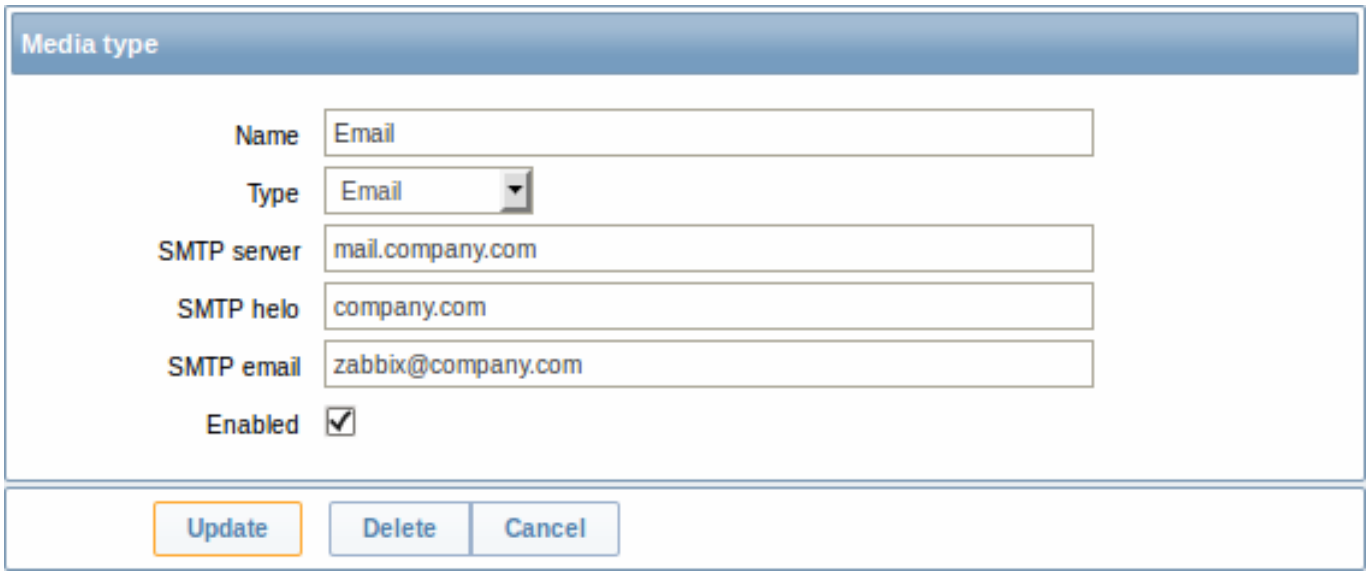
This is what notifications do. E-mail being the most popular delivery method for problem notifications, we will learn how to set up an e-mail notification.

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.



Set the values of SMTP server, SMTP helo and SMTP e-mail to the appropriate for your environment.

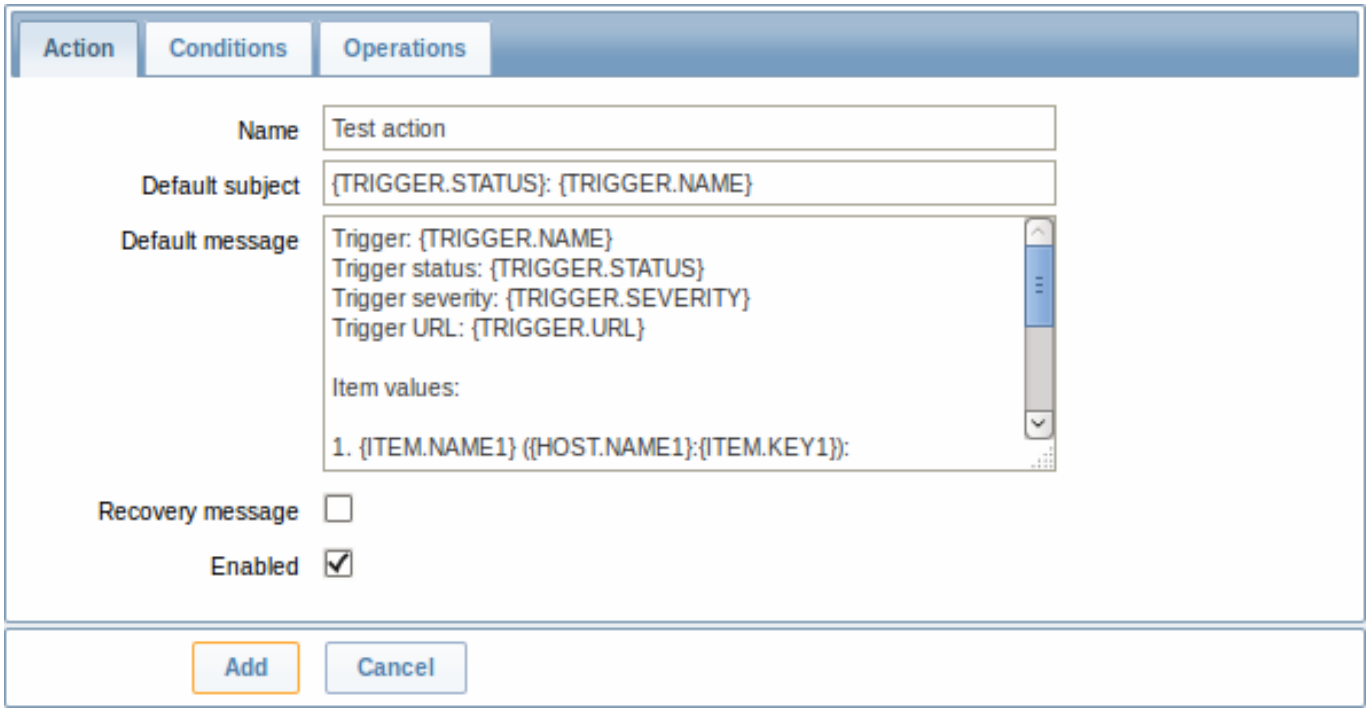
Note:
’SMTP email’ will be used as the ’From’ address for the notifications sent from Zabbix.

Press *Update* when ready.

Now you have configured ’Email’ as a working media type. A media type must be linked to users by defining specific delivery addresses (like we did when configuring a new user), otherwise it will not be used.

New action

Delivering notifications is one of the things actions do in Zabbix. Therefore, to set up a notification, go to *Configuration* → *Actions* and click on *Create action*.

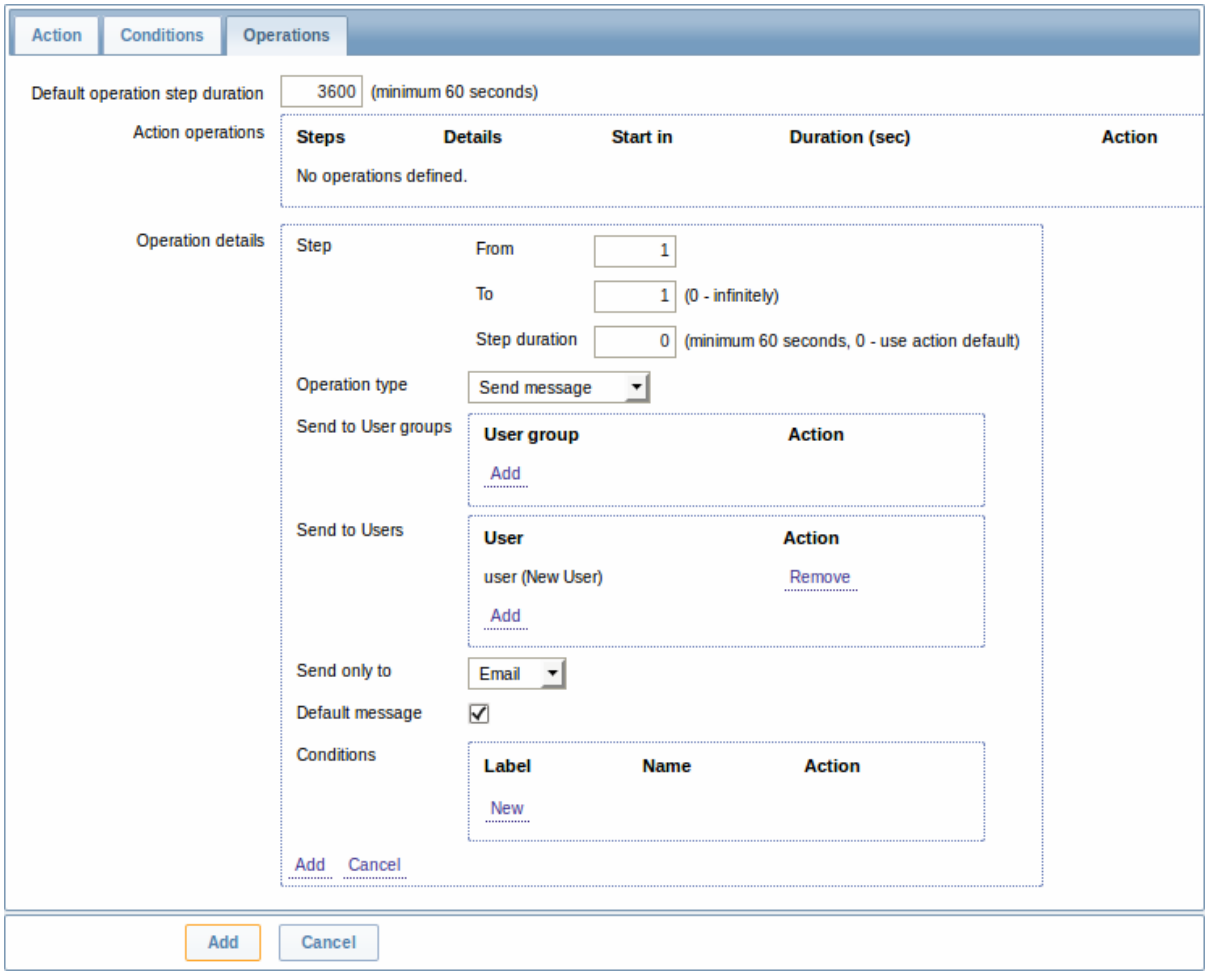


In this form, enter a name for the action.

{TRIGGER.STATUS} and {TRIGGER.NAME} macros (or variables), visible in the *Default subject* and *Default message* fields, will be replaced with the actual trigger status and trigger name values.

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 *New* in there, which opens a new operation form.



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* in the operation detail block.

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* → *Triggers* 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 'Network 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 *Administration* → *Audit*, and choosing *Actions* in the dropdown, located in the upper right corner.

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.

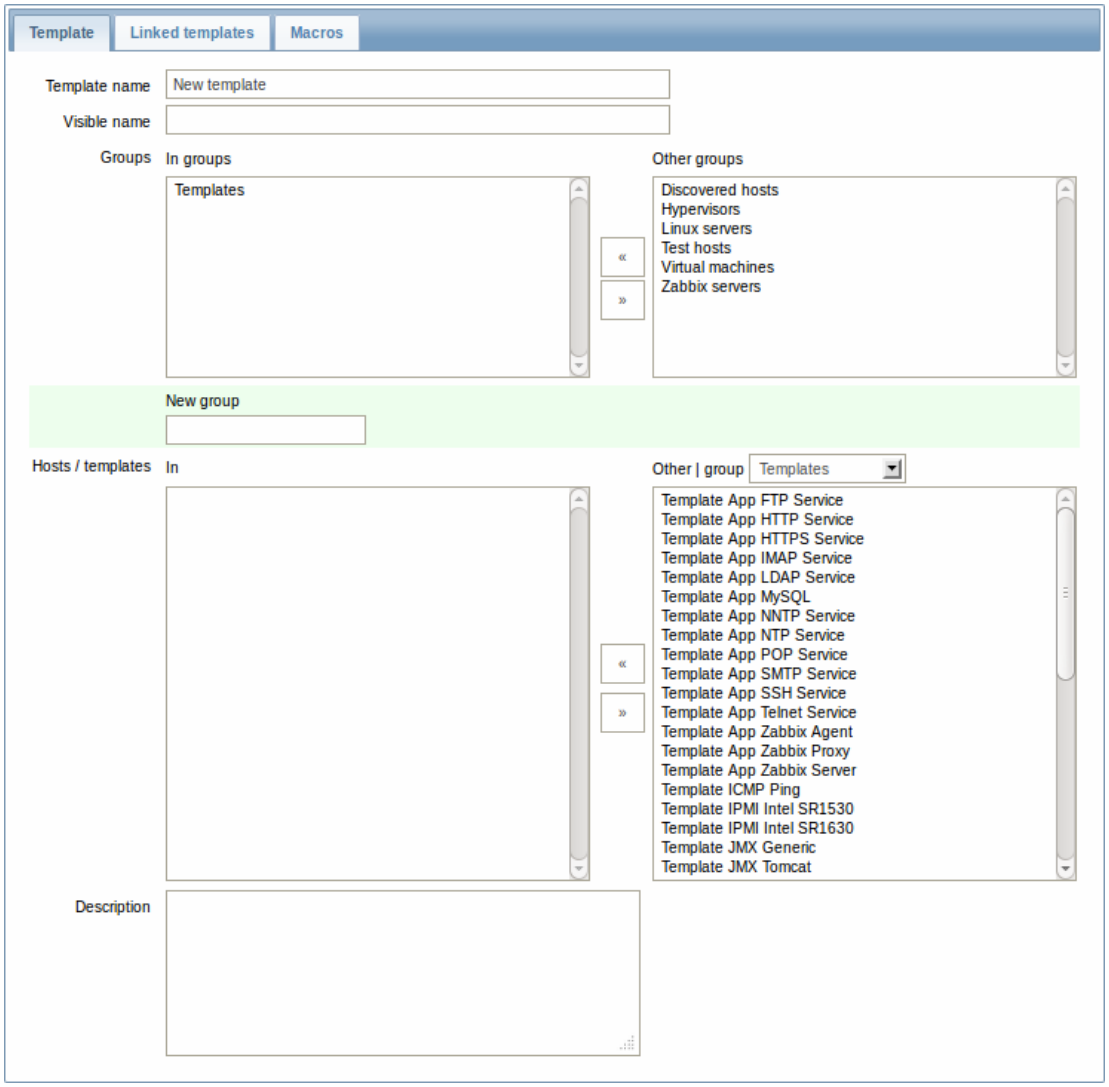
While all of these steps offer a great deal of flexibility in themselves, it may appear like a lot of steps to take if needed for, say, a thousand hosts. Some automation would be handy.

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.



The required parameters to enter here are:

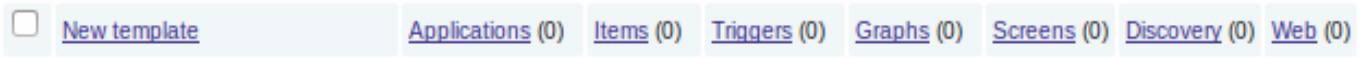
Template name

- Enter a template name. Alpha-numericals, spaces and underscores are allowed.

Groups

- Select one or several groups from the right hand side selectbox and click on « to move them to the 'In groups' selectbox. The template must belong to a group.

When done, click *Add*. Your new template should be visible in the list of templates.



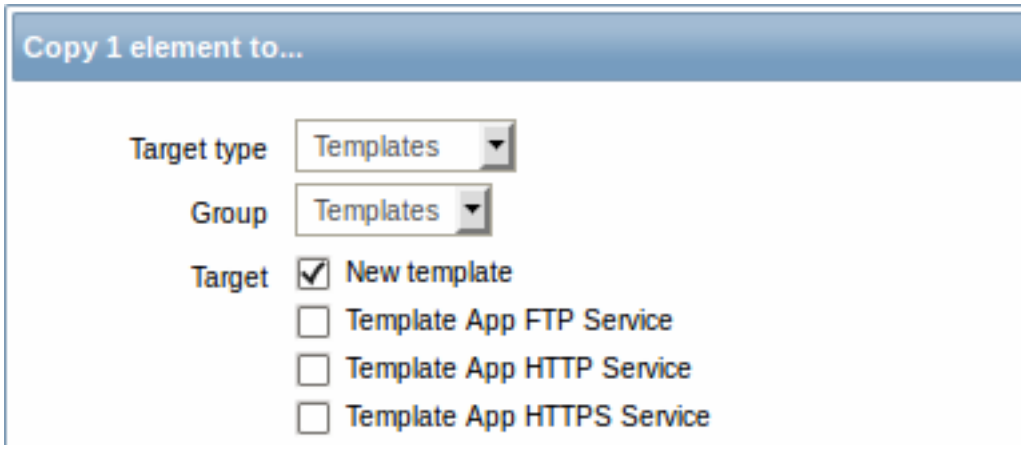
As you may see, the template is there, but it holds nothing in it - no items, triggers or other entities.

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
- select *Copy selected to...* in the dropdown below the list and click on *Go*
- select the template to copy item to



- click on *Copy*

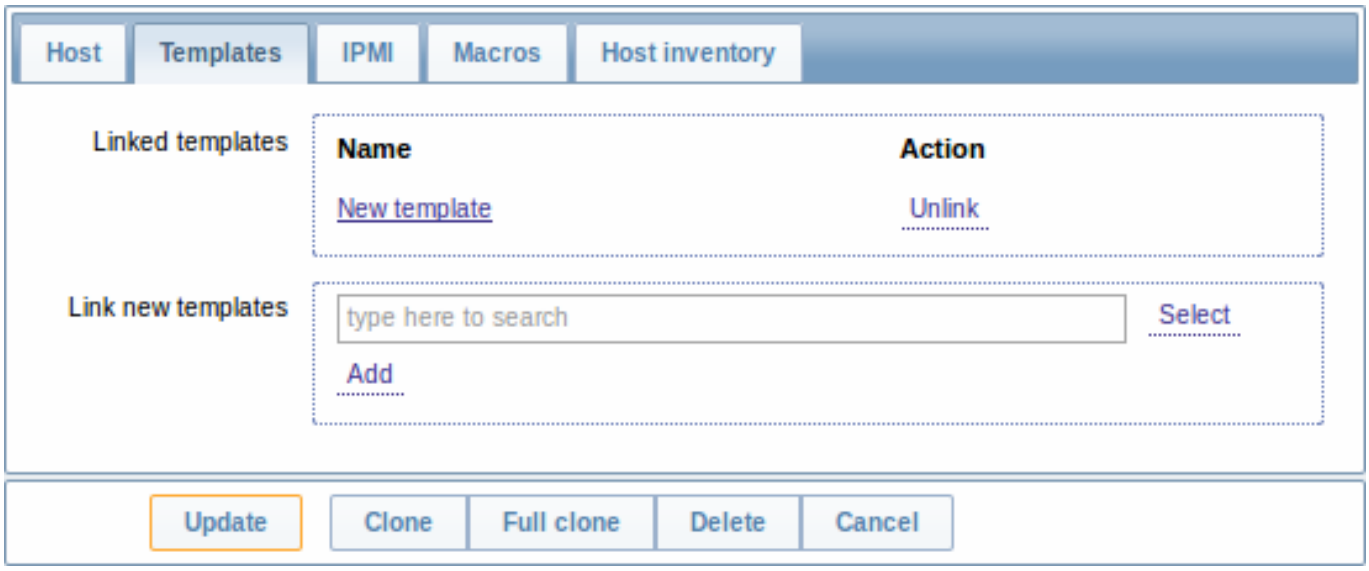
If you now go to *Configuration* → *Templates*, 'New template' should have one new item in it.

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, click on *Select* next to *Link new templates*. In the pop-up window click on the name of template we have created ('New template'). As it appears in the *Link new templates* field, click on *Add*. The template should appear in the *Linked templates* list.



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.

5. Zabbix appliance

As an alternative to setting up manually or reusing existing server for Zabbix, users may download Zabbix appliance.

To get started, boot the appliance and point your browser at the IP it has received over DHCP.

Attention:
Starting with appliance version 2.4.6, allow for a few minutes after the first boot for the Zabbix installation to be set up.

|<| |<| |-|

Zabbix appliance versions are based upon the following OpenSUSE versions:

Zabbix appliance version	OpenSUSE version
2.4.1	13.1
2.4.5	
2.4.6	

It is available in the following formats:

- vmdk (VMware/Virtualbox)
- OVF (Open Virtualisation Format)
- KVM
- CD ISO
- HDD/flash image
- Preload ISO
- Xen guest
- Microsoft VHD (Azure)
- Microsoft VHD (Hyper-V)
- Preload USB

It has Zabbix server configured and running on MySQL, as well as frontend available.

The appliance has been built using SUSE Studio.

Note:
It is not required to download full appliance to get the latest Zabbix version. Zabbix packages may be upgraded. See the #upgrading section for instructions.

1 Changes from appliance version 2.2 Most notably, all utility names and file names have been changed to use a dash instead of an underscore to comply with distribution guidelines. See changes to Zabbix configuration for more information.

2 Changes to SUSE configuration There are some changed applied to the base OpenSUSE configuration.

2.1 MySQL configuration changes

- Binary log is enabled and set to expire in one day
- Maximum packet size is changed from 1MB to 2MB

2.2 Using a static IP address

By default the appliance uses DHCP to obtain IP address. To specify a static IP address:

- Log in as root user;
- Open file */etc/sysconfig/network/ifcfg-<interface>* in your favourite editor;
- Set **BOOTPROTO** variable to **static**;
- Set **IPADDR**, **NETMASK** and any other parameters as required for your network;
- Create file */etc/sysconfig/network/routes*. For the default route, use **default 192.168.1.1 - -** (replacing with your gateway address).
- Run the command **rcnetwork restart**.

To configure DNS, add nameserver entries in */etc/resolv.conf*, specifying each nameserver on its own line: **nameserver 192.168.1.2**.

Alternatively, just use **yast** configuration utility to update network settings.

2.3 Changing time zone

By default the appliance uses UTC for the system clock. To change the time zone, copy appropriate file from */usr/share/zoneinfo* to */etc/localtime*, for example:

```
cp /usr/share/zoneinfo/Europe/Riga /etc/localtime
```

2.4 Other changes

- Network is configured to use DHCP to obtain IP address;
- Utility **fping** is set to have permissions 4710 and is owned by group **zabbix** - suid and only alowed to be used by zabbix group;
- ntpd configured to synchronise to the public pool servers;
- Various basic utilities have been added that could make working with Zabbix and monitoring in general easier.

3 Zabbix configuration Appliance Zabbix setup has the following passwords and other configuration changes:

3.1 Passwords

System:

- root:zabbix
- zabbix:zabbix

Database:

- root:zabbix
- zabbix:zabbix

Zabbix frontend:

- Admin:zabbix

> **Attention:**
> If you change frontend password, do not forget to update password setting web monitoring (*Configuration → Hosts*, *Web* for host ”Zabbix server”).

To change the database user password it has to be changed in the following locations:

- MySQL;
- zabbix-server.conf;
- zabbix.conf.php.

3.2 File locations

- Configuration files are placed in **/etc/zabbix**.
- Zabbix agent logfiles are placed in **/var/log/zabbix**.
- Zabbix server logfiles are placed in **/var/log/zabbixs**.
- Zabbix frontend is placed in **/usr/share/zabbix**.
- Home directory for user **zabbix** is **/var/lib/zabbix**.

3.3 Changes to Zabbix configuration

- Utility names and file names have underscore replaced with a dash to match distribution guidelines. For example, **zabbix-get** is to be used instead of zabbix_get.
- Server name for Zabbix frontend set to ”Zabbix 2.4 Appliance”;
- Frontend timezone is set to Europe/Riga, Zabbix home (this can be modified in **/etc/php5/apache2/php.ini**);
- Disabled triggers and web scenarios are shown by default to reduce confusion.

3.4 Preserving configuration

If you are running live CD version of the appliance or for some other reason can’t have persistent storage, you can create a backup of whole database, including all configuration and gathered data.

To create the backup, run:

```
mysqldump zabbix | bzip2 -9 > dbdump.bz2
```

Now you can transfer file **dbdump.bz2** to another machine.

To restore from the backup, transfer it to the appliance and execute:

```
bzcat dbdump.bz2 | mysql zabbix
```

> **Attention:**
> Make sure that Zabbix server is stopped while performing the restore.

4 Frontend access Access to frontend by default is allowed from:

- 127.0.0.1
- 192.168.0.0/16
- 10.0.0.0/8
- ::1

Root (/) is redirected to /zabbix on the webserver, thus frontend can be accessed both as *http://<host>* and *http://<host>/zabbix*.

This can be customised in **/etc/apache2/conf.d/zabbix.conf**. You have to restart webserver after modifying this file. To do so, log in using SSH as **root** user and execute:

```
service apache2 restart
```

5 Firewall By default, only ports 22 (SSH), 80 (HTTP) and 443 (HTTPS) are open. To open additional ports - for example, Zabbix server and agent ports - modify iptables rules with **SuSEfirewall2** utility:

```
SuSEfirewall2 open EXT TCP zabbix-trapper zabbix-agent
```

Then reload the firewall rules:

```
SuSEfirewall2 start
```

6 Monitoring capabilities Zabbix server is compiled with support for the following:

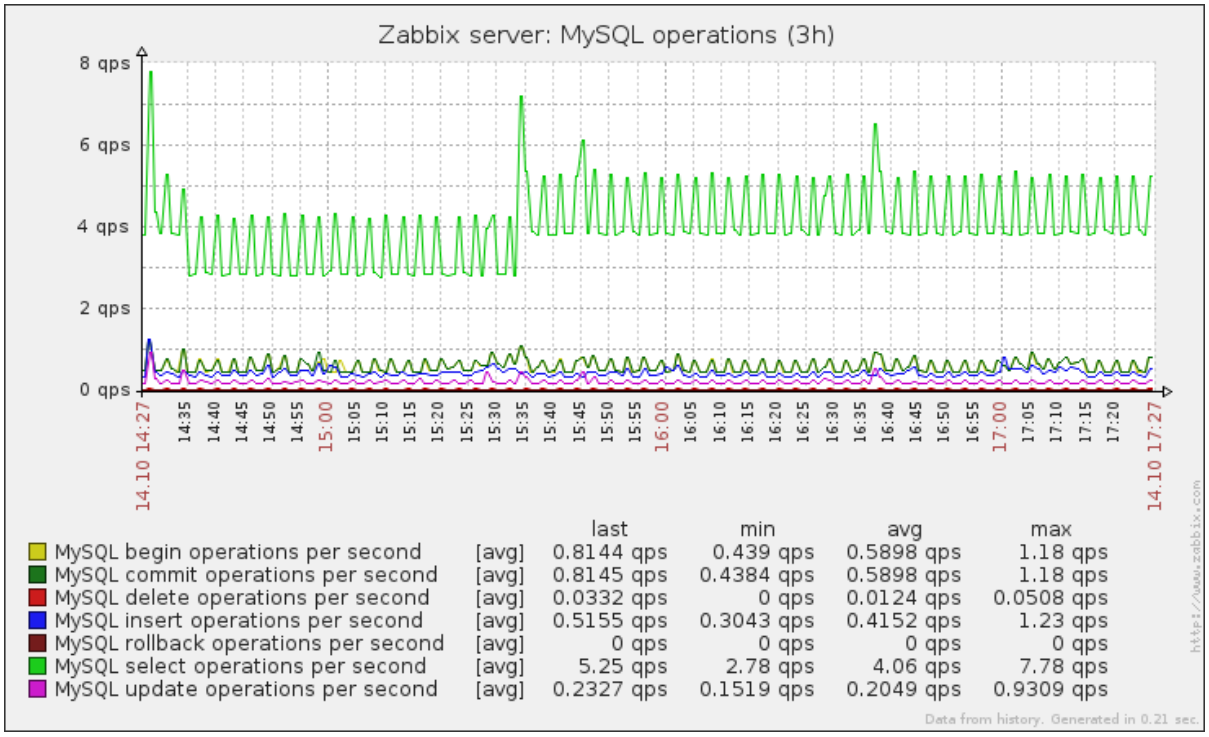
- SNMP
- IPMI
- Web monitoring
- VMware monitoring
- Jabber notifications
- EZ Texting notifications
- ODBC
- SSH2
- IPv6

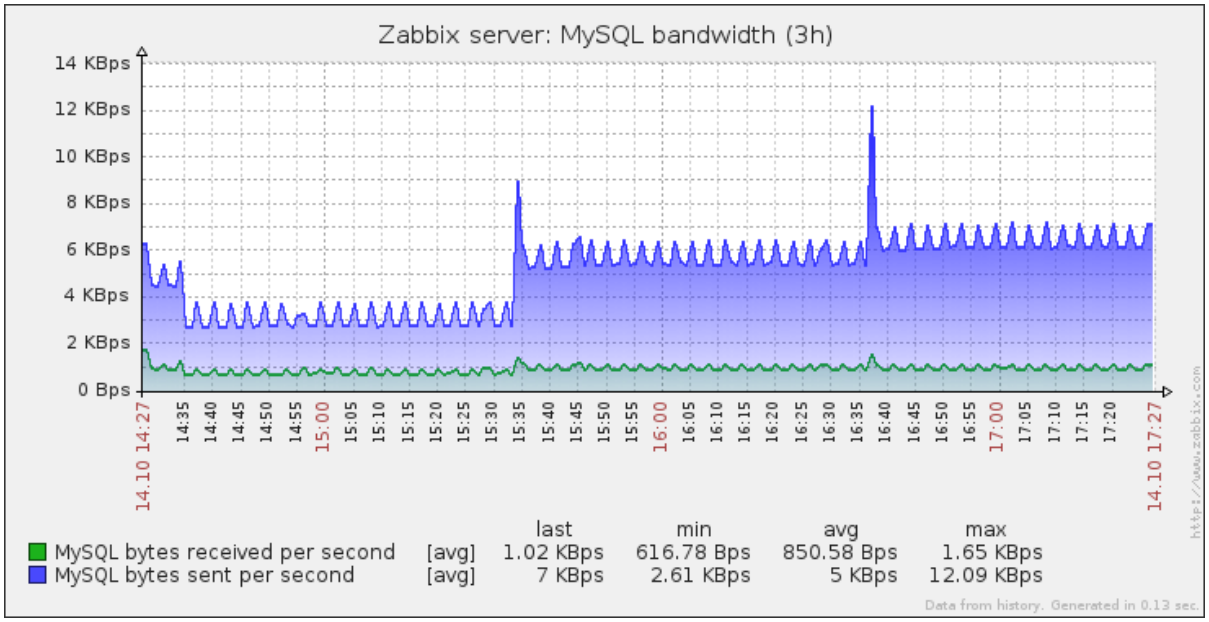
In the provided configuration Zabbix server itself is monitored with the help of locally installed agent for some base parameters, additionally Zabbix frontend is monitored as well using web monitoring.

|<| |<| |-|

Note:
Note that web frontend monitoring logs in - this can add lots of entries to the audit log.

Basic MySQL monitoring is demonstrated using user parameters. Operations per second and bandwidth graphs are provided.





7 Upgrading Starting with Zabbix 2.4, the appliance packages may be upgraded. To do so, run:

```
zypper ref
zypper up
```

The last command will prompt to accept repository signing key (this has to be done only once):

```
New repository or package signing key received:
Key ID: 6F1385F1A0875A36
Key Name: home:zabbix OBS Project <home:zabbix@build.opensuse.org>
Key Fingerprint: DF41712BF09B250CF033AC2D6F1385F1A0875A36
Key Created: Thu 11 Sep 2014 12:36:12 EEST
Key Expires: Sat 19 Nov 2016 11:36:12 EET
Repository: home:zabbix_openSUSE_13_1

Do you want to reject the key, trust temporarily, or trust always? [r/t/a/? shows all options] (r): a
```

Accept the key by entering 'a'. Confirm package upgrade. After the upgrade, it is necessary to manually restart Zabbix daemons. To do so, run:

```
rczabbix-server restart
rczabbix-agentd restart
```

7.1 Bash completion

Bash completion for Zabbix daemons and utilities has been included since the appliance version 2.4.6, but was not included with the previous versions. To install it, run:

```
zypper in zabbix24-bash-completion
```

Confirm the installation and close your login session, then log in again.

8 Naming, init and other scripts Appropriate init scripts are provided. To control Zabbix server, use any of these:

```
service zabbix-server status
rczabbix-server status
```

Replace **server** with **agentd** for Zabbix agent daemon.

8.1 Increasing available diskspace

> **Warning:**
> Create a backup of all data before attempting any of the steps.

Available diskspace on the appliance might not be sufficient. In that case it is possible to expand the disk. To do so, first expand the block device in your virtualisation environment, then follow these steps.

Start *fdisk* to change the partition size. As *root*, execute:

```
fdisk /dev/sda
```

This will start *fdisk* on disk *sda*. Next, switch to sectors by issuing:

```
u
```

> **Attention:**
> Don't disable DOS compatibility mode by entering **c**. Proceeding with it disabled will damage the partition.

Then delete the existing partition and create new one with desired size. In majority of cases you will accept the available maximum, which will expand the filesystem to whatever size you made available for the virtual disk. To do so, enter the following sequence in fdisk prompt:

```
d
n
p
1
(accept default 63)
(accept default max)
```

If you wish to leave some space for additional partitions (swap etc), you can enter another value for *last sector*. When done, save the changes by issuing:

```
w
```

Reboot the virtual machine (as the partition we modified is in use currently). After reboot, filesystem resizing can take place.

```
resize2fs /dev/sda1
```

That's it, filesystem should be grown to the partition size now.

9 Format-specific notes 9.1 Xen

To use images in Xen server, run:

```
xm create -c file-with-suffix.xenconfig
```

See the following pages for more information on using Xen images:

- http://en.opensuse.org/openSUSE:How_to_use_downloaded_SUSE_Studio_appliances#Using_Xen_guests
- http://old-en.opensuse.org/SUSE_Studio_Xen_Howtos

Converting image for XenServer

To use Xen images with Citrix Xenserver you have to convert the disk image. To do so:

- Create a virtual disk which is at least as large as the image
- Find out the UUID for this disk

```
xe vdi-list params=all
```

- If there are lots of disks, they can be filtered by name parameter *name-label*, as assigned when creating the virtual disk
- Import the image

```
xe vdi-import filename="image.raw" uuid="<UUID>"
```

Instructions from Brian Radford blog.

9.2 VMware

The images in *vmdk* format are usable directly in VMware Player, Server and Workstation products. For use in ESX, ESXi and vSphere they must be converted using VMware converter.

9.3 HDD/flash image (raw)

See http://en.opensuse.org/openSUSE:SUSE_Studio_Disc_Image_Howtos for more information on disk images.

10 Known issues 10.1 Extracting on Windows

Windows archive management software is known to mishandle the appliance archives. If extraction fails, try different software. Open source tool 7-zip might work.

10.2 Connectivity problems with IPv6

In some environments, the appliance might obtain IPv6 addresses (for example, for operating system updates), but be unable to use IPv6. To disable IPv6, add *net.ipv6.conf.all.disable_ipv6 = 1* in */etc/sysctl.conf* and restart the appliance.

10.3 Appliance fails to boot

If appliance seems to stop booting while loading initrd, more memory might be needed - 256MB has been shown to be too little. At least 512MB of memory is suggested.

6. Configuration

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

1 Configuring a template

Overview

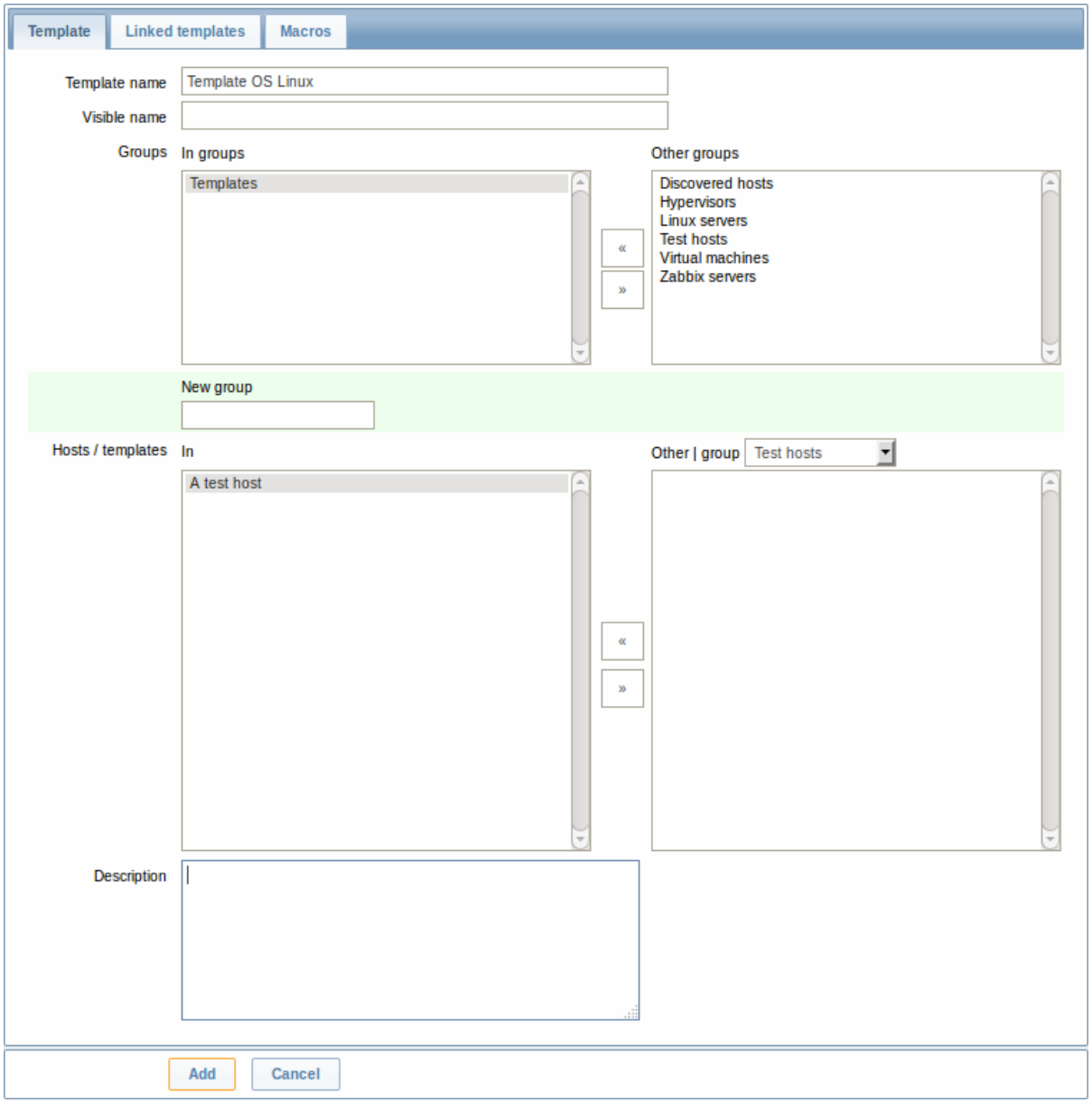
Configuring a template requires that you first create a template by defining its general parameters and then you add entities (items, triggers, graphs etc.) to it.

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.



Template attributes:

Parameter	Description
Template name	Unique template name.
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.
New group	A new group can be created to hold the template. Ignored, if empty.
Hosts/Templates	List of hosts/templates the template is applied 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 in the *Link new templates* field until a list of templates corresponding to the entered letter(s) appear. Scroll down to select. When all templates to be linked are selected, click on *Add*.

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 **Macros** tab allows you to define template-level user macros.

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

To add items to the template, do the following:

- Go to *Configuration* → *Hosts* (or *Templates*)
- Click on *Items* in the row of the required host/template
- Mark the checkboxes of items you want add to the template
- Select *Copy selected to...* below the item list and click on *Go*
- 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.

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
- Click on *Add* next to *Link new templates*
- Select one or several templates in the 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 grey 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

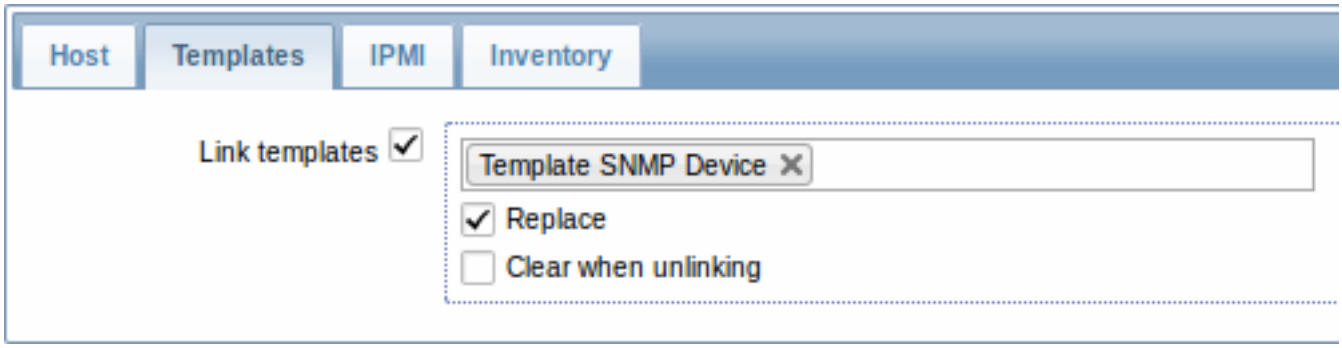
Linking templates to several hosts

There are some ways of mass-applying templates (to many hosts at once):

- To link a template to many hosts, in *Configuration* → *Templates*, click on the template, then select hosts from the respective group in the *Other* box, click on « and update the template.

Vice versa, if you select the linked hosts in the *In* box, click on » and update the template, you unlink the template from these hosts (while the hosts will still inherit the items, triggers, graphs etc. from the template).

- To update template linkage of many hosts, in *Configuration* → *Hosts* select some hosts by marking their checkboxes, then choose **Mass update** below the list, click on *Go* 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 *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.

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 on individual templates entities for various services, applications etc. you may end up with quite a few templates all of which may need to be linked to quite a few hosts. To simplify the picture, it is possible to link some templates together, in one ”nested” template.

The benefit of nesting is that then you have to link only the one template to the host and the host will inherit all entities of the linked templates automatically.

Configuring a nested template

If you want to link some templates, to begin with you can take an existing template or a new one, then:

- Open the template properties form
- Look for the *Linked templates* tab
- Click on *Select* to select templates in the popup window
- Click on *Add* to list selected templates

- Click on *Add/Update* in the template properties form

Now the template should have all the entities (items, triggers, custom graphs etc.) of the linked templates.

To unlink any of the linked templates, in the same form use the *Unlink* or *Unlink and clear* buttons and click on *Update*.

Choosing the *Unlink* option will simply remove the association with the other template, while not removing all its entities (items, triggers, graphs etc).

Choosing the *Unlink and clear* option will remove both the association with the other template and all its entities (items, triggers, graphs etc).

Permission issues

- You may have a setup where an Admin level user has *Read-write* access to some Template A while not having *Read-write* access to Template B that holds Template A in a nested setup. In this case, an item created on Template A, while inherited by the hosts of Template A, **will not** be inherited by the hosts of Template B. Thus, creating a trigger for such an item will fail altogether, because of missing corresponding items on hosts of Template B.

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:

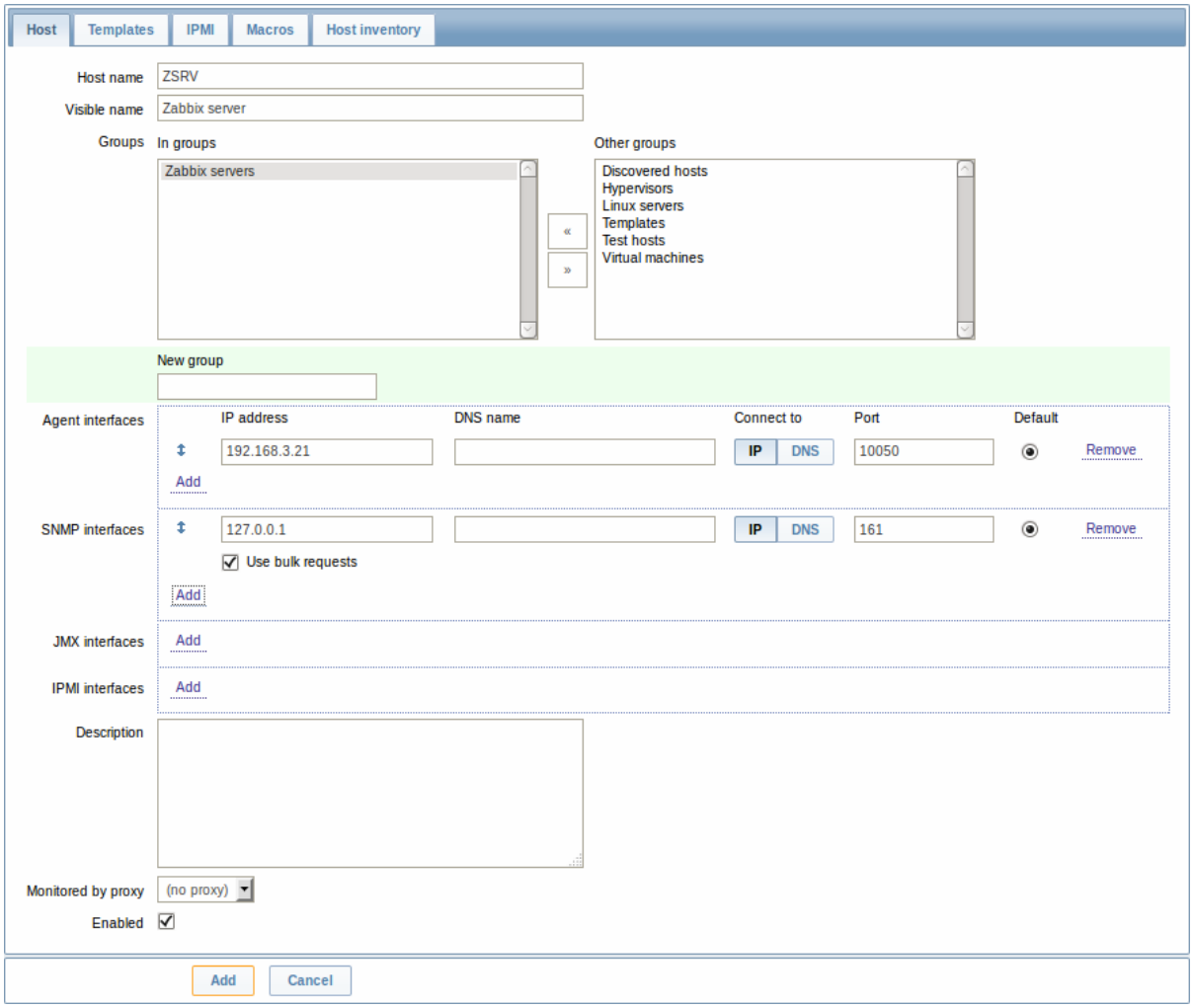
- Go to: *Configuration* → *Hosts*
- Click on *Create host* to the right (or on the host name to edit an existing host)
- Enter parameters of the host in the form

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:



Parameter	Description
Host name	Enter a unique host name. Alphanumerics, spaces, dots, dashes and underscores are allowed. *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	If you set this name, it will be the one visible in lists, maps, etc. This attribute has UTF-8 support.
Groups	Select host groups the host belongs to. A host must belong to at least one host group.
New host group	A new group can be created and linked to the host. Ignored, if empty.
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 greyed out for them. *Use bulk requests* option for SNMP interfaces allows to enable/disable bulk processing of SNMP requests per interface.
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.

Parameter	Description
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 in the *Link new templates* field until a list of matching templates appear. Scroll down to select. When all templates to be linked are selected, click on *Add*.

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.

See also known issues about template linkage.

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.
Password	Password for authentication.

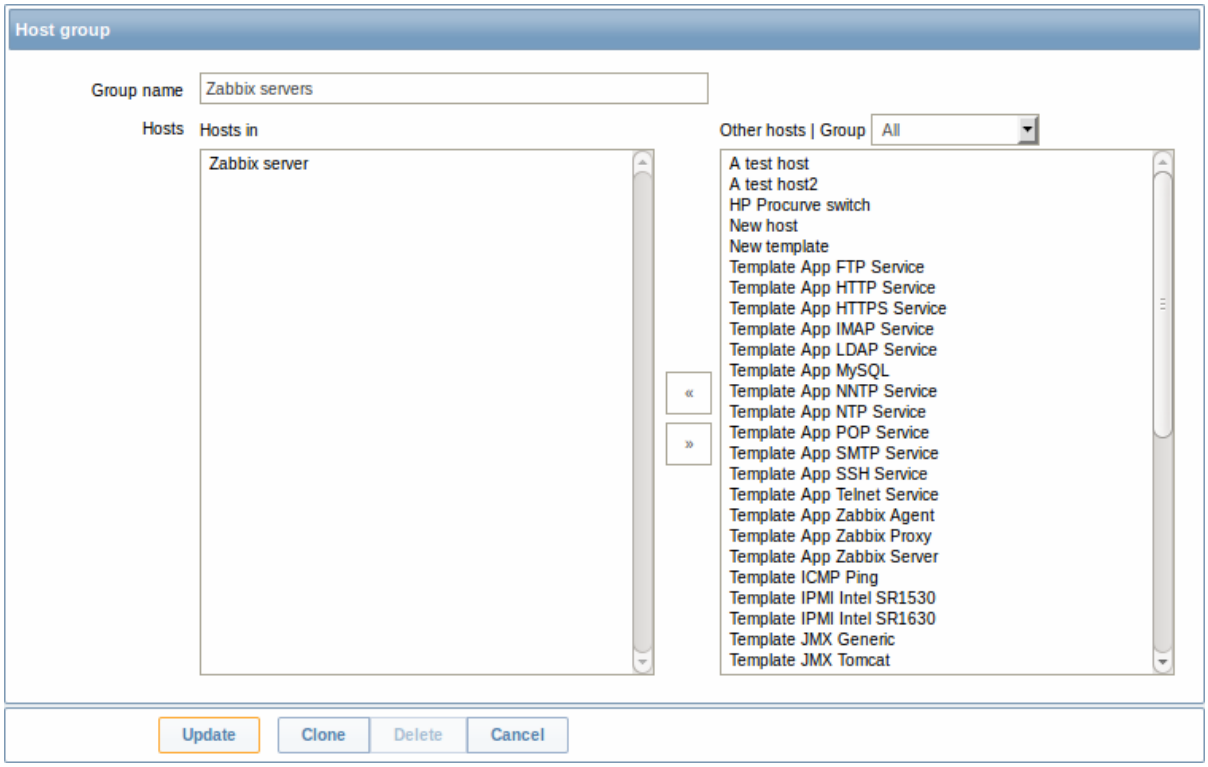
The **Macros** tab allows you to define host-level user macros.

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.

Configuring a host group

To configure 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



Parameter	Description
Group name	Enter a unique host group name.
Hosts	Select hosts, members of the group. A host group may have zero, one or more hosts.

2 Inventory

Overview

You can keep the inventory of networked devices in Zabbix.

There is a special *Inventory* menu in the Zabbix frontend. However, you will not see any data there initially and it is not where you enter data. Building inventory data is done manually when configuring a host or automatically by using some automatic population options.

Building inventory

Manual mode

When configuring a host, in the *Host inventory* tab you can enter such details as the type of device, serial number, location, responsible person, etc - data that will populate inventory information.

If a URL is included in host inventory information and it starts with 'http' or 'https', it will result in a clickable link in the *Inventory* section.

Automatic mode

Host inventory can also be populated automatically. For that to work, when configuring a host the inventory mode in the *Host inventory* tab must be set to *Automatic*.

Then you can configure host items to populate any host inventory field with their value, indicating the destination field with the respective attribute (called *Item will populate host inventory field*) in item configuration.

Items that are especially useful for automated inventory data collection:

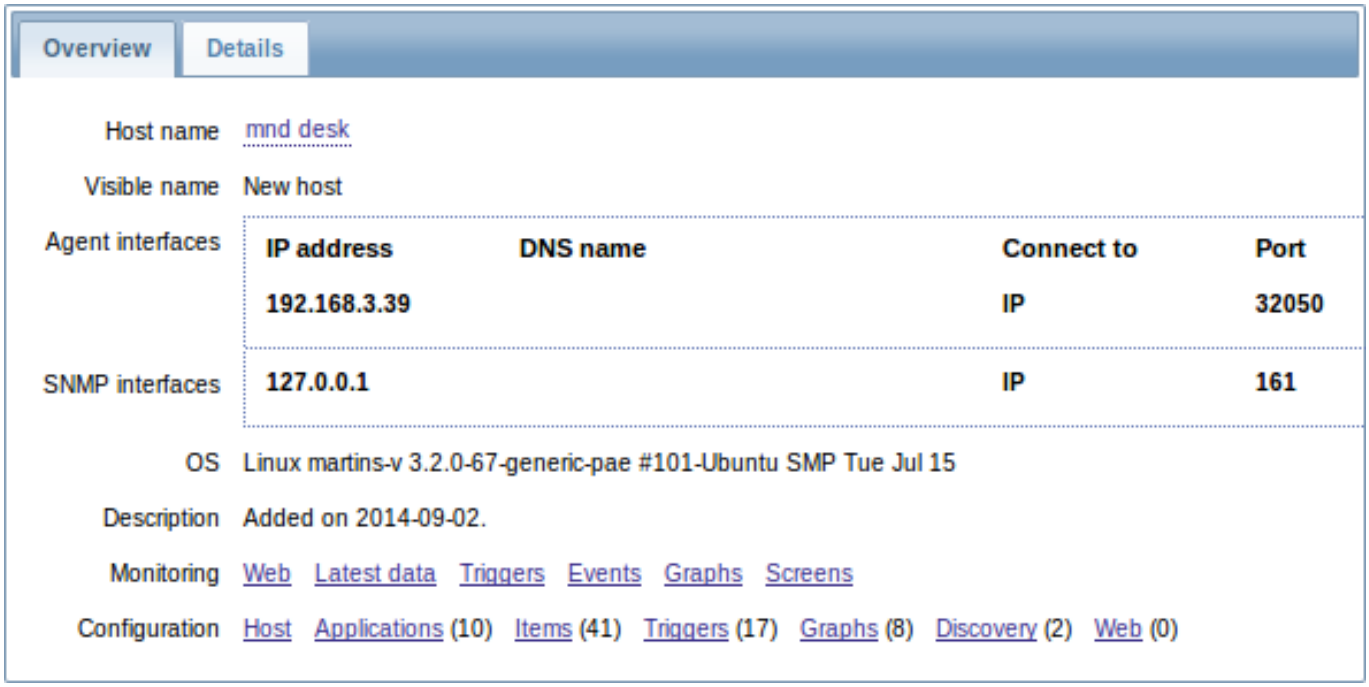
- system.hw.chassis[full|type|vendor|model|serial] - default is [full], root permissions needed
- system.hw.cpu[all|cpunum,full|maxfreq|vendor|model|curfreq] - default is [all,full]
- system.hw.devices[pci|usb] - default is [pci]
- system.hw.macaddr[interface,short|full] - default is [all,full], interface is regexp
- system.sw.arch
- system.sw.os[name|short|full] - default is [name]
- system.sw.packages[package,manager,short|full] - default is [all,all,full], package is regexp

Inventory 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*, *Events*, *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}."

Attention:
{PROFILE.*} macros from previous Zabbix versions are still supported but it's highly recommended to change those to {INVENTORY.*}

For more details, see the Macros supported by location page.

3 Mass update

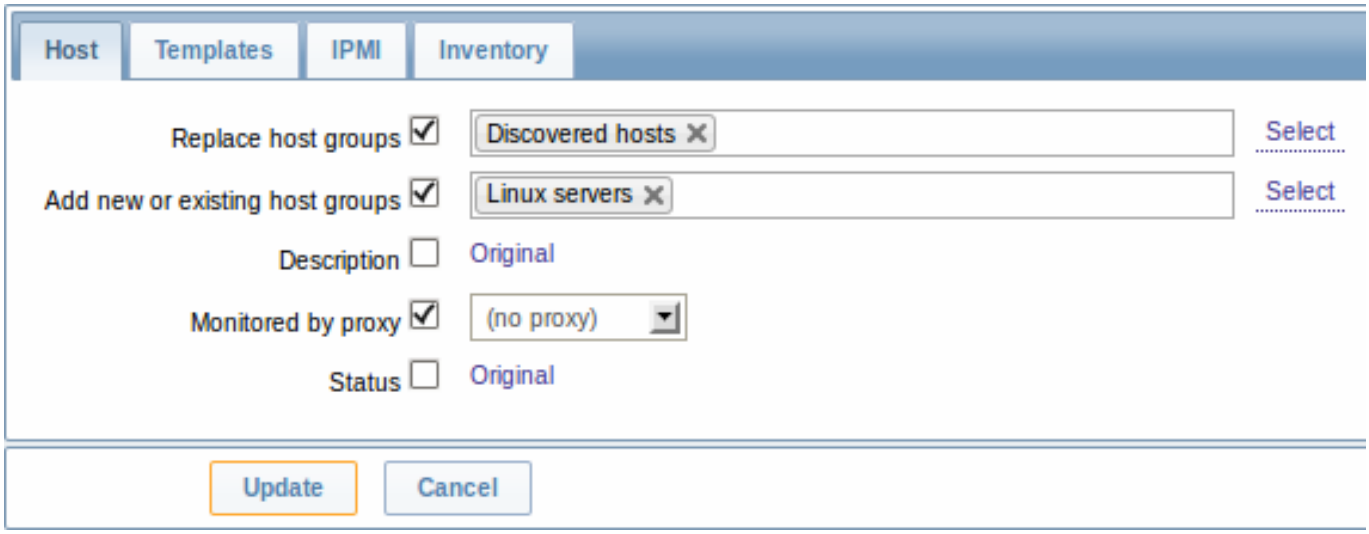
Overview

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:

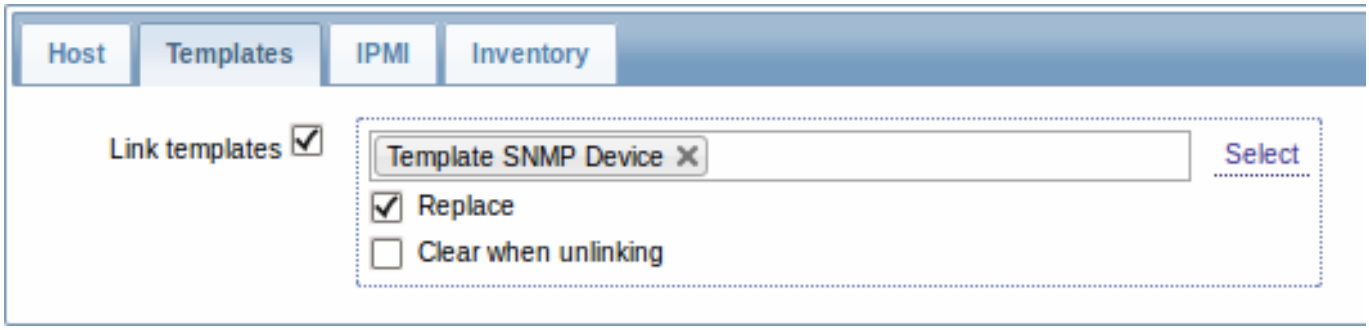
- Mark the checkboxes before the hosts you want to update in the host list
- Select *Mass update* from the dropdown below and click on *Go*
- Navigate to the desired tab of attributes (*Host*, *Templates*, *IPMI* or *Inventory*)
- Mark the checkboxes of any attribute to update and enter a new value for them



Replace host groups will remove the host from any existing host groups and replace those with the one(s) specified in this field.

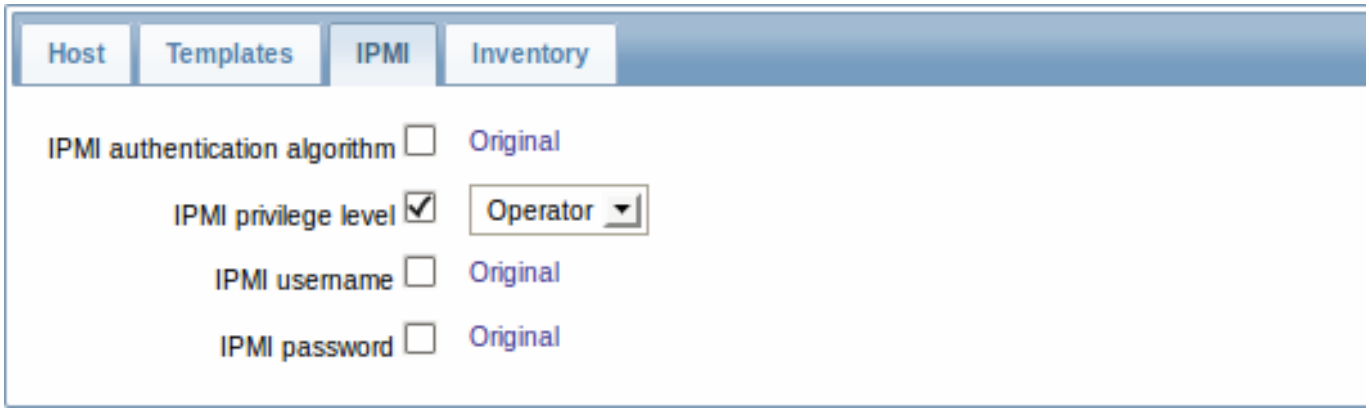
Add new or existing host groups allows to specify additional host groups from the existing ones or enter completely new host groups for the hosts.

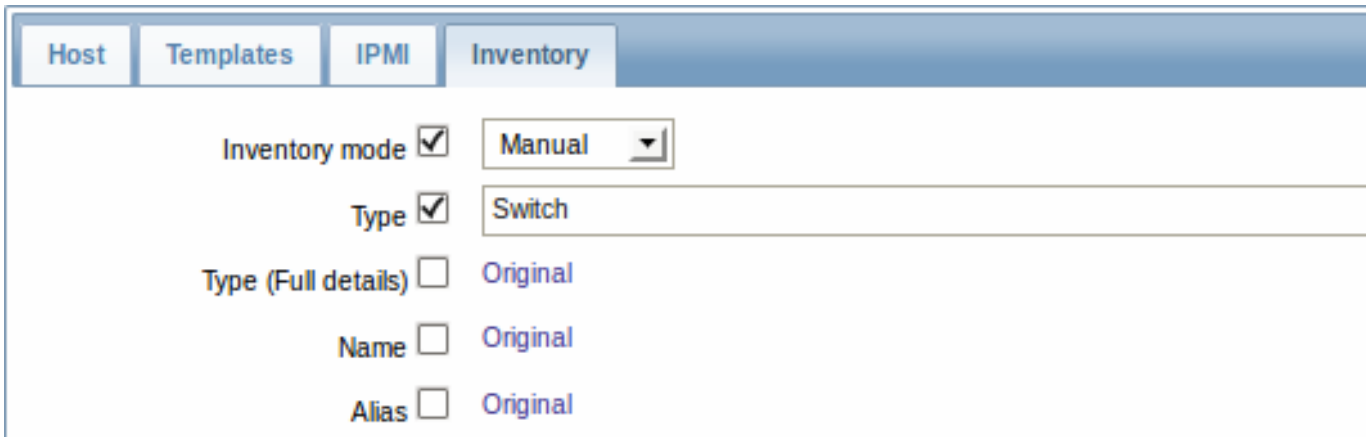
Both these fields are auto-complete - starting to type in them offers a dropdown of matching host groups. If the host group is new, it also appears in the dropdown and it is indicated by *(new)* after the string. Just scroll down to select.



To update template linkage in the **Templates** tab, 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 *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.).





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

Items are the ones that gather data from a host.

Once you have configured a host, you need to add some monitoring items to start getting actual data.

An item is an individual metric. One way of quickly adding many items is to attach one of the predefined templates to a host. For optimized system performance though, you may need to fine-tune the templates to have only as many items and as frequent monitoring as is really necessary.

In an individual item you specify what sort of data will be gathered from the host.

For that purpose you use the item key. Thus an item with the key name **system.cpu.load** will gather data of the processor load, while an item with the key name **net.if.in** will gather incoming traffic information.

To specify further parameters with the key, you include those in square brackets after the key name. Thus, system.cpu.load**[avg5]** will return processor load average for the last 5 minutes, while net.if.in**[eth0]** will show incoming traffic in the interface eth0.

Note:
For all supported item types and item keys, see individual sections of item types.

Proceed to creating and configuring an item.

1 Creating an item

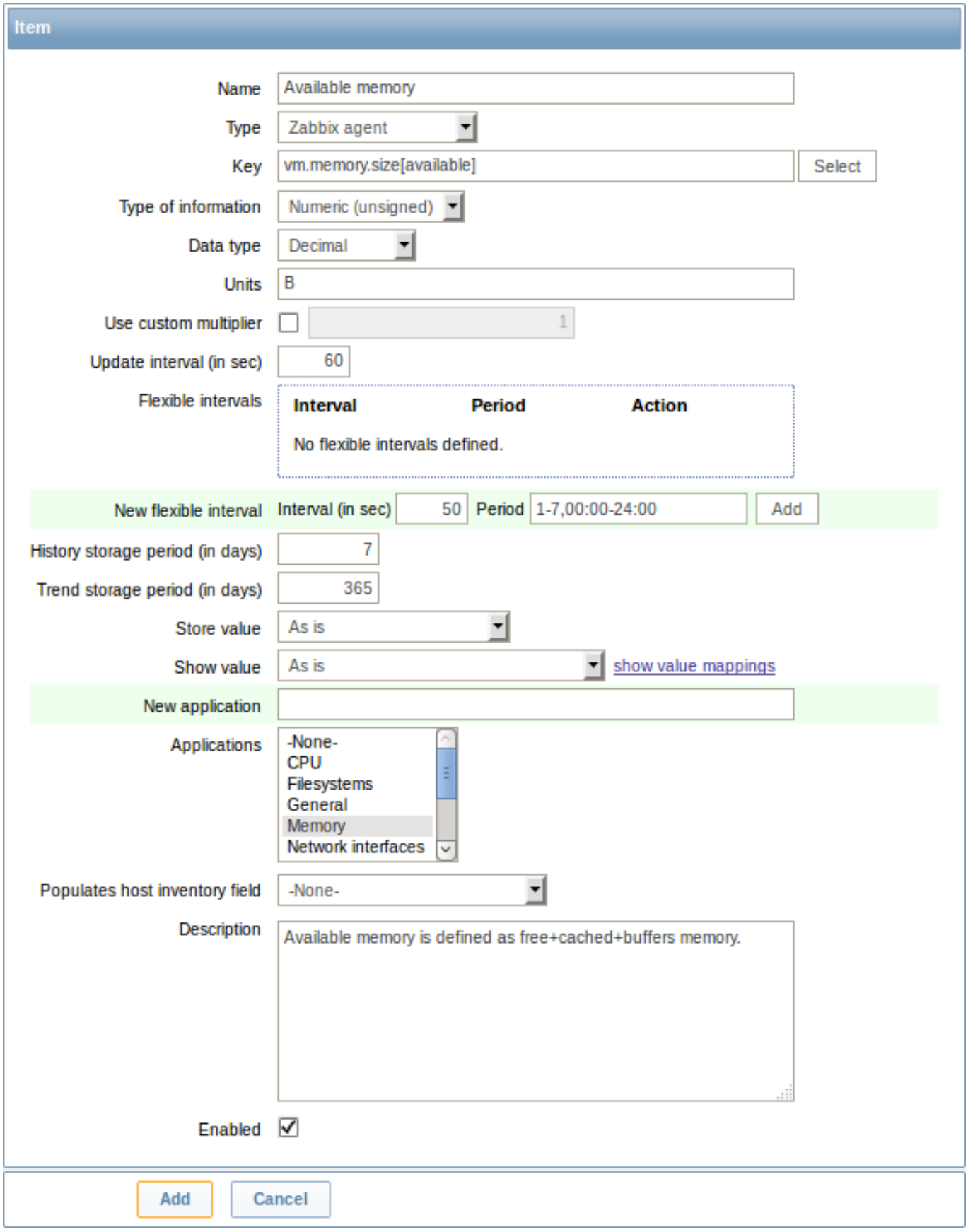
Overview

To create an item in Zabbix frontend, do the following:

- Go to: *Configuration* → *Hosts*
- Click on *Items* in the row of the host

- Click on *Create item* in the upper right corner of the screen
- Enter parameters of the item in the form

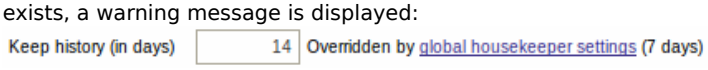
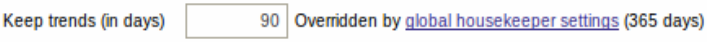
Configuration



Item attributes:

Parameter	Description
Name	This is how the item will be named. The following macros can be used: **$1, $2...$9** - referring to the first, second... ninth parameter of the item key 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 /"
Type	Item type. See individual item type sections.
Key	Item key. 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)** - 64bit unsigned integer **Numeric (float)** - floating point number Negative values can be stored. Allowed range (for MySQL): -999999999999.9999 to 999999999999.9999 (double(16,4)). Starting with Zabbix 2.2, receiving values in scientific notation is also supported. E.g. 1e+70, 1e-70. **Character** - character (string) data limited to 255 bytes **Log** - log file. Must be set for log*, eventlog item keys. **Text** - text of unlimited size
Data type	Data type is used for integer items in order to specify the expected data type: **Boolean** - textual representation 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 *FALSE* - false, f, no, n, off, down, unused, disabled, unavailable Additionally, any non-zero numeric value is considered to be TRUE and zero is considered to be FALSE. **Octal** - data in octal format **Decimal** - data in decimal format **Hexadecimal** - data in hexadecimal format Zabbix will automatically perform the conversion to numeric. The conversion is done by Zabbix server (even when a host is monitored by Zabbix proxy).

Parameter	Description
Units	If a unit symbol is set, Zabbix will add post processing to the received value and display it with the set unit postfix. By default, if the raw value exceeds 1000, it is divided by 1000 and displayed accordingly. For example, if you set *bps* and receive a value of 881764, it will be displayed as 881.76 Kbps. Special processing is used for **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. See also the unit blacklist.
Use custom multiplier	If you enable this option, all received values will be multiplied by the integer or floating-point value set in the value field. Use this option to convert values received in KB, MBps, etc into B, Bps. Otherwise Zabbix cannot correctly set prefixes (K, M, G etc). Starting with Zabbix 2.2, using scientific notation is also supported. E.g. 1e+70.
Update interval (in sec)	Retrieve a new value for this item every N seconds. *Note*: If set to '0', the item will not be polled. However, if a flexible interval also exists with a non-zero value, the item will be polled during the flexible interval duration.

Parameter	Description
Flexible intervals	You can create exceptions to *Update interval*. For example: Interval: **10**, Period: **1-5,09:00-18:00** - will check the item every 10 seconds during working hours. Interval: **0**, Period: **1-7,00:00-7:00** - will disable checking the item at night. Interval: **0**, Period: **7-7,00:00-24:00** - will disable checking the item on Sundays. To check an item once per day at a specific time (say, 12:00), set the default *Update interval* to '0', but specify **60** in the flexible interval and a period like **1-7,12:00-12:01** Up to seven flexible intervals can be defined. If multiple flexible intervals overlap, the smallest *Interval* value is used for the overlapping period. Note that if the smallest value of overlapping flexible intervals is '0', no polling will take place. Outside the flexible intervals the default update interval is used. See the page about setting time periods for description of the *Period* format. 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. *Note*: Not available for Zabbix agent active items.
History storage period (in days)	Number of days to keep detailed history in the database. Older data will be removed by the housekeeper. Starting with Zabbix 2.2, this value can be overridden globally in *Administration* → *General* → *Housekeeper*. If the global setting exists, a warning message is displayed:  It is recommended to keep the recorded values for the smallest possible number of days to reduce the size of value history in the database. Instead of keeping long history of values, you can keep longer data of trends. See also History and trends.
Trend storage period (in days)	Keep aggregated (hourly min, max, avg, count) detailed history for N days in the database. Older data will be removed by the housekeeper. Starting with Zabbix 2.2, this value can be overridden globally in *Administration* → *General* → *Housekeeper*. If the global setting exists, a warning message is displayed:  *Note:* Keeping trends is not available for non-numeric data - character, log and text. See also History and trends.

Parameter	Description
Store value	**As is** - no pre-processing **Delta (speed per second)** - evaluate value as (**value**-**prev_value**)/(**time**-**prev_time**), where *value* - current value *value_prev* - previously received value *time* - current timestamp *prev_time* - timestamp of previous value This setting is extremely useful to get speed per second for a constantly growing value. If 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. **Delta (simple change)** - evaluate as (**value**-**prev_value**), where *value* - current value *value_prev* - 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.
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 not be parsed. 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 line. Log time format for this line would be "ppppp:yyyyMMdd:hhmmss". Note that "p" and ":" chars are just placeholders and can be anything but "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.
Description	Enter an item description.
Enabled	Mark the checkbox to enable the item so it will be processed.

You can also create an item by opening an existing one, pressing the *Clone* button and then saving under a different name.

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.

Note:
If you use a custom multiplier or store value as *Delta (speed 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 integer.

Unit blacklist

By default, specifying a unit for an item will result in a multiplier prefix being added - for example, value 2048 with unit B would be displayed as 2KB. For a pre-defined, hardcoded list of units this is prevented:

- ms
- RPM
- rpm
- %

Note that both lowercase and uppercase **rpm** (*rpm* and *RPM*) strings are blacklisted.

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 section.

1 Item key

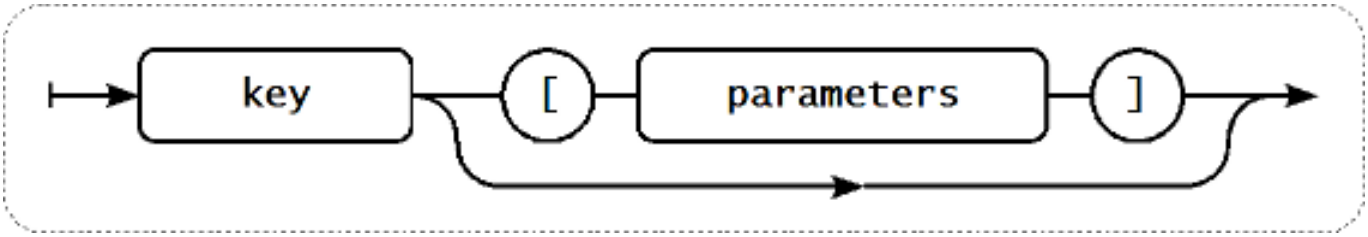
1 Flexible and non-flexible parameters

A flexible parameter is a parameter which accepts an argument. For example, in vfs.fs.size[*] the asterisk symbol '*' indicates a flexible parameter. '*' is any string that will be passed as an argument to the parameter. Correct definition examples:

- vfs.fs.size[/]
- vfs.fs.size[/opt]

2 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.

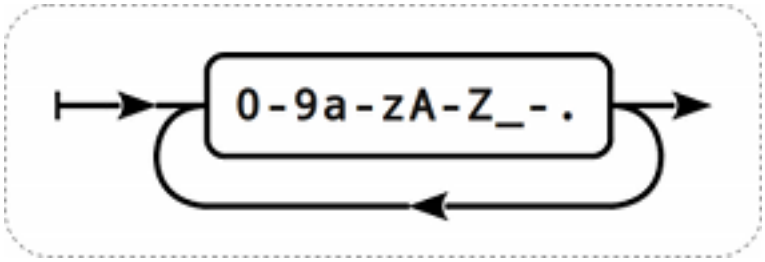
Key name

The key name itself has a limited range of allowed characters, which just follow each other. Allowed characters are:

`0-9a-zA-Z_-.`

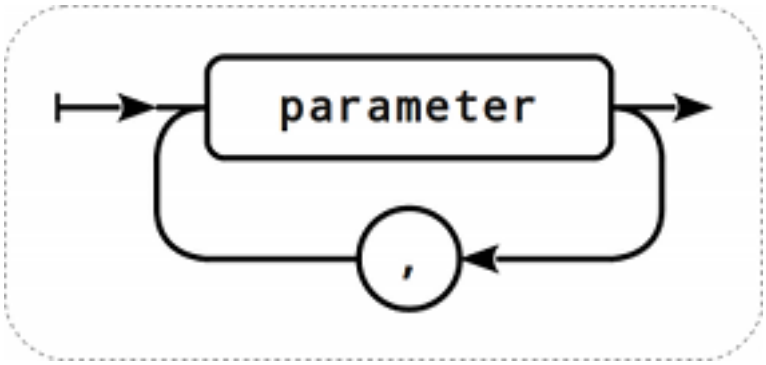
Which means:

- all numbers;
- all lowercase letters;
- all uppercase letters;
- underscore;
- dash;
- dot.

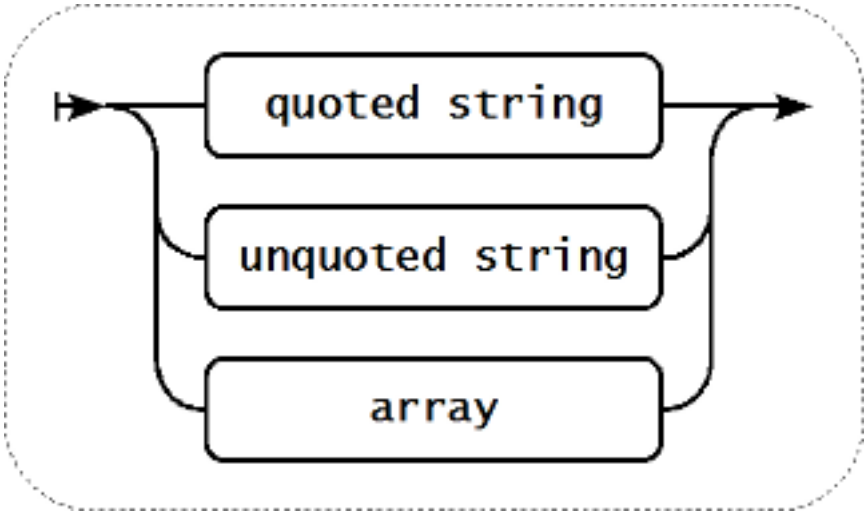


Key parameters

An item key can have multiple parameters that are comma separated.



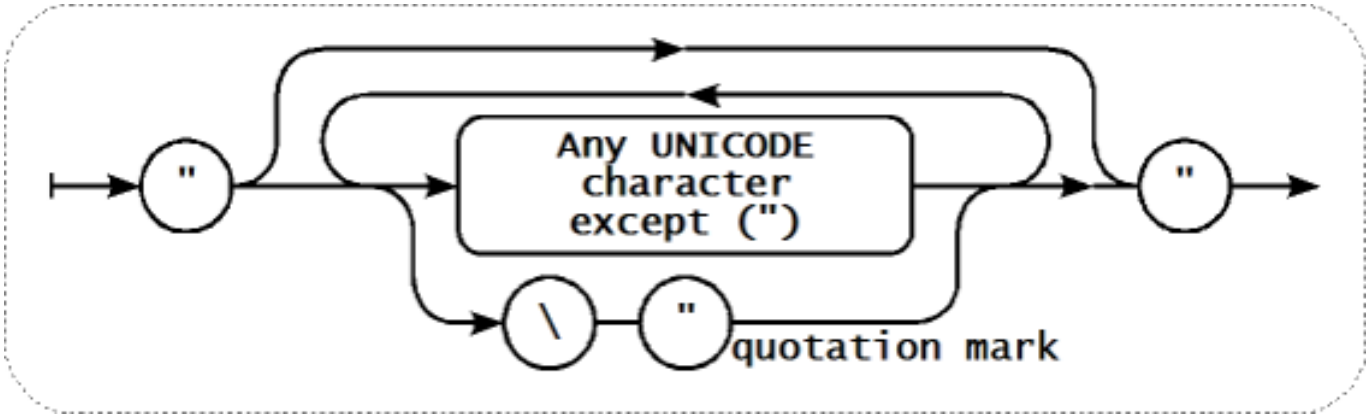
Each key parameter can be either a quoted string, an unquoted string or an array.



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.

Parameter - quoted string

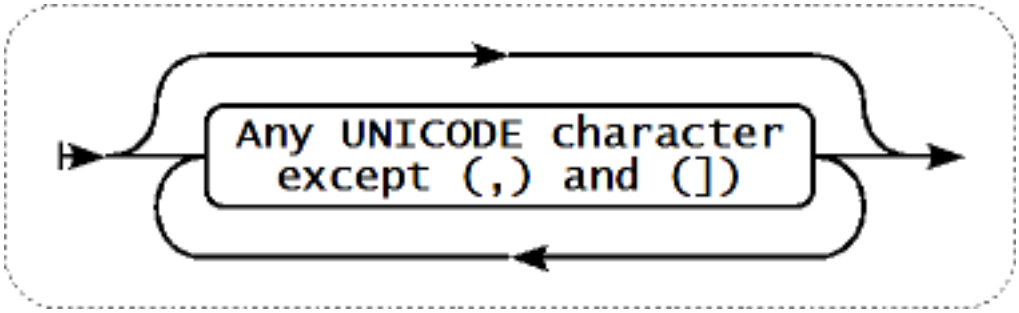
If the key parameter is a quoted string, any Unicode character is allowed, and included double quotes must be backslash escaped.



Warning:
To quote item key parameters, use double quotes only. Single quotes are not supported.

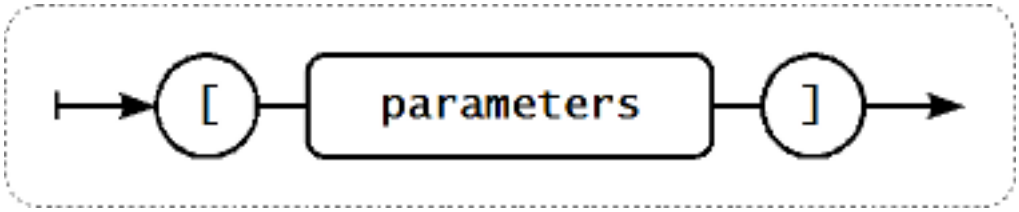
Parameter - unquoted string

If the key parameter is an unquoted string, any Unicode character is allowed except comma and right square bracket (]).



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.



2 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

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 specific for Windows agent

** Mandatory and optional parameters **

Parameters without angle brackets are mandatory. Parameters marked with angle brackets **< >** are optional.

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.version	Version of Zabbix agent.	String		Example of returned value: 1.8.2
kernel.maxfiles	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>]				

Key					
	Log file monitoring.	Log	**file** - full path and name of log file **regexp** - regular expression 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** - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items that have not returned any data yet). **output** - an optional output formatting template. The **\0** escape sequence is replaced with the matched text 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).	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„ The `output` parameter is supported since Zabbix 2.2. The `mode` parameter is supported since Zabbix 2.0. See also additional information on log monitoring.

Key
logrt[file_regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>]

Key				
	Log file monitoring with log rotation support.	Log	**file_regexp** - absolute path to file and regexp describing the file name pattern **regexp** - regular expression describing the required content pattern **encoding** - code page identifier **maxlines** - maximum number of new lines per second the agent will send to Zabbix server or proxy. This parameter overrides the value of 'MaxLinesPerSecond' in zabbix_agentd.conf **mode** - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items that have not returned any data yet). **output** - an optional output formatting template. The **\0** escape sequence is replaced with the matched text 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	The item must be configured as an active check. Log rotation is based on the last modification time of files. 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[”/home/zabbix/logs/^logfil 9]{1,3}$”„,100] → will match a file like ”logfile1” (will not match ”.logfile1”) => log[”/home/user/^logfile_.*_[0 9]{1,3}$”,”pattern_to_match” 8”,100] → will collect data from files such ”logfile_abc_1” or ”logfile__001”. The `output` parameter is supported since Zabbix 2.2. The `mode` parameter is supported since Zabbix

Key				
net.dns[<ip>,name,<type>,<timeout>,<count>]	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) **name** - DNS name to query **type** - record type to be queried (default is *SOA*) **timeout** (ignored on Windows) - timeout for the request in seconds (default is 1 second) **count** (ignored on Windows) - number of tries for the request (default is 2)	Example: => net.dns[8.8.8.8,zabbix.com,MX The possible values for `type` are: *ANY*, *A*, *NS*, *CNAME*, *MB*, *MG*, *MR*, *PTR*, *MD*, *MF*, *MX*, *SOA*, *NULL*, *WKS* (except for Windows), *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>]				

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) **name** - DNS name to query **type** - record type to be queried (default is *SOA*) **timeout** (ignored on Windows) - timeout for the request in seconds (default is 1 second) **count** (ignored on Windows) - number of tries for the request (default is 2)	Example: => net.dns.record[8.8.8.8,zabbix. The possible values for `type` are: *ANY*, *A*, *NS*, *CNAME*, *MB*, *MG*, *MR*, *PTR*, *MD*, *MF*, *MX*, *SOA*, *NULL*, *WKS* (except for Windows), *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>]	Incoming traffic statistics on network interface.	Integer	**if** - network interface name (Unix); network interface full description or IPv4 address (Windows) **mode** - possible values: *bytes* - number of bytes (default) *packets* - number of packets *errors* - number of errors *dropped* - number of dropped packets	Multi-byte interface names on Windows are supported since Zabbix agent version 1.8.6. 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 a *Delta (speed per second)* store value 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 (Windows) **mode** - possible values: *bytes* - number of bytes (default) *packets* - number of packets *errors* - number of errors *dropped* - number of dropped packets	Multi-byte interface names on Windows are supported since Zabbix agent 1.8.6 version. 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 a *Delta (speed per second)* store value 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 (Windows) **mode** - possible values: *bytes* - number of bytes (default) *packets* - number of packets *errors* - number of errors *dropped* - number of dropped packets	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 a *Delta (speed per second)* store value 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.
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
net.tcp.port[<ip>,port]				

Key				
	Checks if it is possible to make TCP connection to specified port.	0 - cannot connect 1 - can connect	**ip** - IP address (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[*]*
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*, *ntp*, *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 by Windows agent is currently not supported. 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. *ntp* service only works since Zabbix 2.0.15, 2.2.10 and 2.4.6, despite being

Key				
net.tcp.service.perf[service,<ip>,<port>]	Checks performance of service.	0 - service is down seconds - the number of seconds spent while connecting to the service	**service** - either of: *ssh*, *ntp*, *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 by Windows agent is currently not supported. 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. *ntp* service only works since Zabbix 2.0.15, 2.2.10 and 2.4.6, despite being available in earlier versions. Old naming: *check_service_perf[*]*
net.udp.listen[port]				

Key				
	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
proc.mem[<name>,<user>,<mode>,<cmdline>]				
	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)	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.
proc.num[<name>,<user>,<state>,<cmdline>]				

Key				
	The number of processes.	Integer	**name** - process name (default is *all processes*) **user** - user name (default is *all users*) **state** - possible values: *all* (default), *run*, *sleep*, *zomb* **cmdline** - filter by command line (it is a regular expression)	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 On Windows, only the `name` and `user` parameters are supported.
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.

Key				
				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>]	CPU load.	Float	**cpu** - possible values: *all* (default), *percpu* (total load divided by online CPU count) **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*	Example: => system.cpu.load[,avg5] *percpu* is supported since Zabbix 2.0.0. 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>]				

Key				
	CPU utilisation in percent.	Float	**cpu** - *<CPU number>* or *all* (default) **type** - possible values: *idle*, *nice*, *user* (default), *system* (default for Windows), *iowait*, *interrupt*, *softirq*, *steal* **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*	Example: => system.cpu.util[0,user,avg5] Old naming: *system.cpu.idleX*, *system.cpu.niceX*, *system.cpu.systemX*, *system.cpu.userX*
system.hostname[<type>]				

Key				
	System host name.	String	**type** (Windows only, must not be used on other systems) - possible values: *netbios* (default) or *host*	The value is acquired by either GetComputerName() (for **netbios**) or gethostname() (for **host**) functions on Windows and by "hostname" command on other systems. Examples of returned values: *on Linux*: => system.hostname → linux-w7x1 => system.hostname → www.zabbix.com *on Windows*: => system.hostname → WIN-SERV2008-I6 => system.hostname[host] → Win-Serv2008-I6LonG The `type` parameter for this item is supported since Zabbix **1.8.6**. See also a more detailed description.
system.hw.chassis[<info>]				

Key				
	Chassis information.	String	**info** - possible values: *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 in memory. **Root permissions** are required because the value is acquired by reading from memory. Supported since Zabbix agent version 2.0.
system.hw.cpu[<cpu>,<info>]	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>]				

Key				
	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>]	Listing of MAC addresses.	String	**interface** - *all* (default) or a regular expression **format** - *full* (default) or *short*	Lists MAC adresses of the interfaces whose name matches the given `interface` regexp (*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>]				

Key				
	System time.	Integer - with `type` as *utc* String - with `type` as *local*	**type** - possible values: *utc* - (default) the time since the Epoch (00:00:00 UTC, January 1, 1970), measured in seconds. *local* - the time in the 'yyyy-mm-dd,hh:mm:ss.nnn,+hh:mm' format	Parameters for this item are supported since Zabbix agent version 2.0. 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*: To enable this functionality, agent configuration file must contain EnableRemoteCommands=1 option. *Note*: The return value of the item is standard output together with standard error produced by command. *Note:* 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				
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>]	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 are present in all distributions): /proc/version (*full*) /proc/version_signature (*short*) /etc/issue.net (*name*) Supported since Zabbix agent version 2.0.
system.sw.packages[<package>,<manager>,<format>]				

Key					
	Listing of installed packages.	Text	**package** - *all* (default) or a regular expression **manager** - *all* (default) or a package manager **format** - *full* (default) or *short*	Lists (alphabetically) installed packages whose name matches the given `package` regexp (*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 seperate 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 supported by platform for details on defaults.	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 supported by platform for details on defaults.	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[1]	**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)	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*). Old naming: *system.swap.free*, *system.swap.total*
system.uname				

Key			
	Detailed host information.	String	Example of returned value: FreeBSD localhost 4.2-RELEASE FreeBSD 4.2-RELEASE #0: Mon Nov i386 Since Zabbix 2.2.0, the value for this item is obtained by using the uname() system call, whereas previously it was obtained by invoking ”uname -a” on Unix systems. Hence, 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.
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.read[<device>,<type>,<mode>]			

Key				
	Disk read statistics.	Integer - with `type` in *sectors*, *operations*, *bytes* Float - with `type` in *sps*, *ops*, *bps*	**device** - disk device (default is *all*[2]) **type** - possible values: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* This parameter must be specified, since defaults differ under various OSes. *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.	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*). Supports LVM since Zabbix 1.8.6. Only relative device names could be used (for example, **sda**) until Zabbix 1.8.6. Since then, an optional **/dev/** prefix may be used (for example, **/dev/sda**). 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*	**device** - disk device (default is *all*[2]) **type** - possible values: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* This parameter must be specified, since defaults differ under various OSes. *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.	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*). Supports LVM since Zabbix 1.8.6. Only relative device names could be used (for example, **sda**) until Zabbix 1.8.6. Since then, an optional **/dev/** prefix may be used (for example, **/dev/sda**). Old naming: *io[*]*
vfs.file.cksum[file]				

Key				
	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. 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]	Checks if file exists	0 - not found 1 - regular file or a link (symbolic or hard) to regular file exists	**file** - full path to file	Example: => vfs.file.exists[/tmp/application The return value depends on what S_ISREG POSIX macro returns. The file size limit depends on large file support.
vfs.file.md5sum[file]				

Key				
	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>]	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** - GNU regular expression **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 text 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. 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** - GNU regular expression **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).	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) - modification time, *access* - last access time, *change* - last change time	Example: => vfs.file.time[/etc/passwd,modi The file size limit depends on large file support.
vfs.fs.discovery				
	List of mounted filesystems. Used for low-level discovery.	JSON object		Supported since Zabbix agent version 2.0.
vfs.fs.inode[fs,<mode>]				

Key				
	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)	Example: => vfs.fs.inode[/,pfree] Old naming: *vfs.fs.inode.free[*]*, *vfs.fs.inode.pfree[*]*, *vfs.fs.inode.total[*]*
vfs.fs.size[fs,<mode>]				
	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>]				

Key				
	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*, *wired*, *used*, *pused* (used, percentage), *available*, *pavailable* (available, percentage)	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*, *wired*; 3) user-level estimates on how much memory is used and available: *used*, *pused*, *available*, *pavailable*. See a more detailed description of vm.memory.size parameters. Old naming: *vm.memory.buffers*, *vm.memory.cached*, *vm.memory.free*, *vm.memory.shared*, *vm.memory.total*
web.page.get[host,<path>,<port>]	Get content of web page.	Web page source as text	**host** - hostname **path** - path to HTML document (default is /) **port** - port number (default is 80)	Returns an empty string on fail. Example: => web.page.get[www.zabbix.com
web.page.perf[host,<path>,<port>]	Loading time of full web page (in seconds).	Float	**host** - hostname **path** - path to HTML document (default is /) **port** - port number (default is 80)	Returns 0 on fail. Example: => web.page.perf[www.zabbix.co
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 **path** - path to HTML document (default is /) **port** - port number (default is 80) **regexp** - GNU regular expression **length** - maximum number of characters to return **output** - an optional output formatting template. The **\0** escape sequence is replaced with the matched text 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).	Returns an empty string if no match was found or on fail. 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.zabbix

Note:
[1] The system.swap.size key might report incorrect data on virtualized (VMware ESXi, VirtualBox) Windows platforms. In this case use perf_counter[\700(_Total)\702] key to obtain correct swap usage percentage.

Note:
[2] If default *all* is used for the first parameter of **vfs.dev.*** keys then the keys 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.

Note:
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.

Available encodings

The `encoding` parameter is used to specify encoding for processing corresponding item checks, so that data acquired will not be corrupted. For a list of supported encodings (code page identifiers), please consult respective documentation, such as documentation for libiconv (GNU Project) or Microsoft Windows SDK documentation for "Code Page Identifiers".

If empty `encoding` is passed, then UTF-8 (default locale for newer Unix/Linux distributions, see your system's settings) or ANSI with system-specific extension (Windows) is used by default.

Troubleshooting agent items

1. If used with 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.

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 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 (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.win.conf **mode** - possible values: *all* (default), *skip* - skip processing of	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$\|^70$**). The `mode` parameter is supported since Zabbix 2.0.0. "Windows Eventing 6.0" is supported since Zabbix 2.2.0. See also additional information on log monitoring.

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 utilisation. Since 1.6, **interval** is used, so the check returns an average value for last "interval" seconds every time. See also: Windows performance counters.
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_state[service]				
	State of a service.	0 - running 1 - paused 2 - start pending 3 - pause pending 4 - continue pending 5 - stop pending 6 - stopped 7 - unknown 255 - no such service	**service** - a real service name or its display name as seen in MMC Services snap-in	
services[<type>,<state>,<exclude>]				
	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>]				

Key				
	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	This key is supported starting with Zabbix 2.2.0. Example: => wmi.get[root\cimv2,select status from Win32_DiskDrive where Name like '%PHYSICALDRIVE0%'] - returns the status of the first physical disk

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.

To monitor the up/down status of a service you need to perform the following steps:

Step 1

Get the service name.

You can get that name by going to the services mmc 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, with:

- *Key*: service_state[lanmanworkstation]
- *Type of information*: Numeric (unsigned)
- *Show value*: select the *Windows service state* value mapping

2 SNMP agent

Overview

You may want to use SNMP monitoring on devices such as printers, network switches, routers or UPS that usually are SNMP-enabled and on which it would be impractical to attempt setting up complete operating systems and Zabbix agents.

To be able to retrieve data provided by SNMP agents on these devices, Zabbix server must be initially configured with SNMP support.

SNMP checks are performed over the UDP protocol only.

Since Zabbix 2.2.3 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. Please see the technical detail section below on how it works internally. Since Zabbix 2.4 there is also a "Use bulk requests" setting for each interface that allows to disable bulk requests for devices that cannot handle them properly.

Since Zabbix 2.2.7 and Zabbix 2.4.2 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.

Since Zabbix 2.2 Zabbix server and proxy daemons correctly use the Timeout configuration parameter when performing SNMP checks. Additionally the daemons do not perform retries after a single unsuccessful SNMP request (timeout/wrong credentials). Previously the SNMP library default timeout and retry values (1 second and 5 retries respectively) were actually used.

Since Zabbix 2.2.8 and Zabbix 2.4.2 Zabbix server and proxy daemons will always retry at least one time: 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.

Configuring SNMP monitoring

To start monitoring a device through SNMP, the following steps have to be performed:

Step 1

Create a host for the device with an SNMP interface.

Enter the IP address. 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.

Note:
SNMP checks do not use *Agent port*, it is ignored.

Step 2

Find out the SNMP string (or OID) of the item you want to monitor.

To get a list of SNMP strings, use the **snmpwalk** command (part of net-snmp software which you should have installed as part of the Zabbix installation) or equivalent tool:

```
shell> snmpwalk -v 2c -c public <host IP> .
```

As '2c' here stands for SNMP version, you may also substitute it with '1', to indicate SNMP Version 1 on the device.

This should give you a list of SNMP strings and their last value. If it doesn't then it is possible that the SNMP 'community' is different from the standard 'public' in which case you will need to find out what it is.

You can then go through the list until you find the string you want to monitor, e.g. if you wanted to monitor the bytes coming in to your switch on port 3 you would use the `IF-MIB::ifInOctets.3` string from this line:

```
IF-MIB::ifInOctets.3 = Counter32: 3409739121
```

You may now use the **snmpget** command to find out the numeric OID for 'IF-MIB::ifInOctets.3':

```
shell> snmpget -v 2c -c public -On 10.62.1.22 IF-MIB::ifInOctets.3
```

Note that the last number in the string is the port number you are looking to monitor. See also: Dynamic indexes.

This should give you something like the following:

```
.1.3.6.1.2.1.2.2.1.10.3 = Counter32: 3472126941
```

Again, the last number in the OID is the port number.

Note:
3COM seem to use port numbers in the hundreds, e.g. port 1 = port 101, port 3 = port 103, but Cisco use regular numbers, e.g. port 3 = 3.

Note:
Some of the most used SNMP OIDs are translated automatically to a numeric representation by Zabbix.

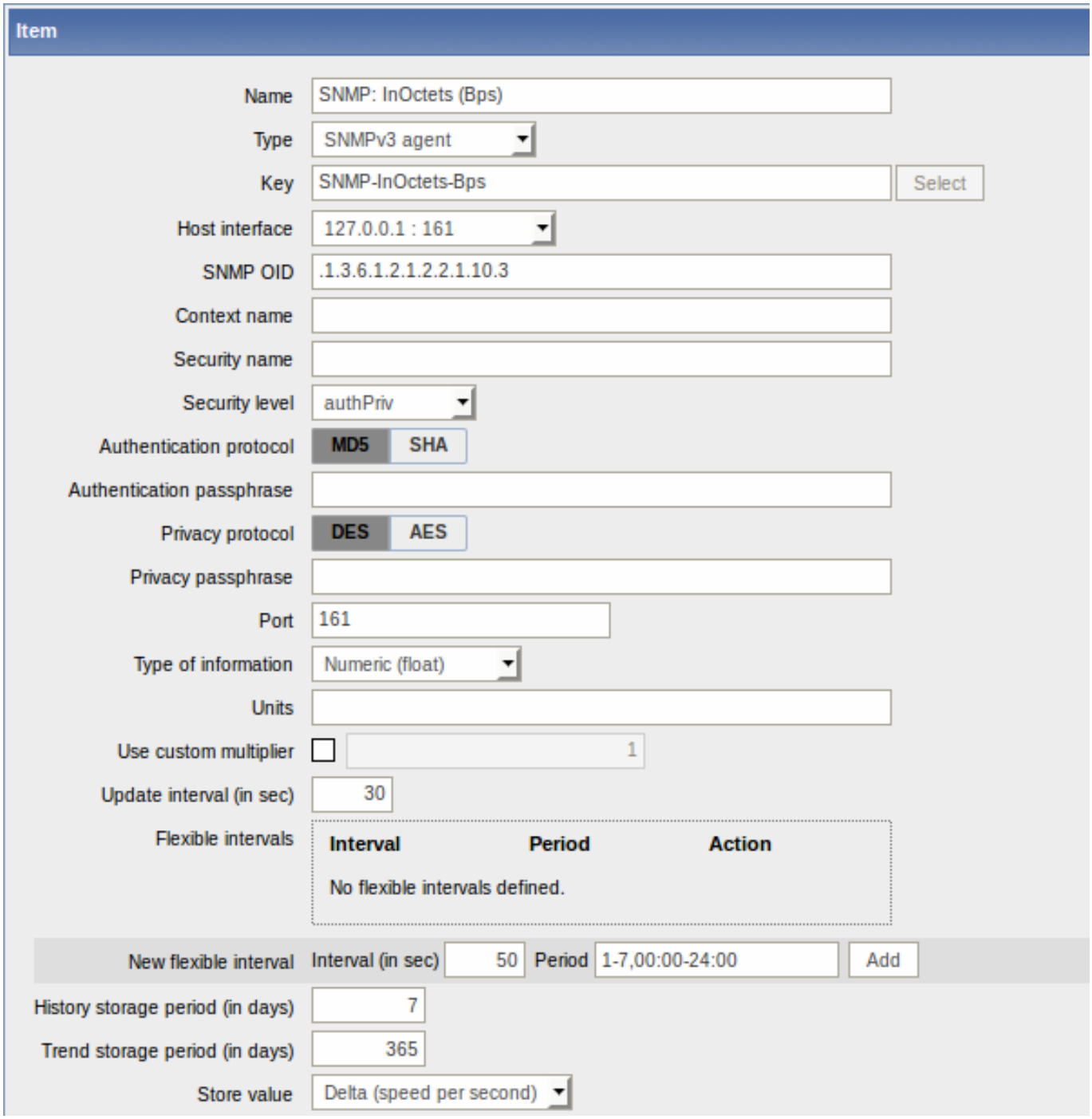
In the last example above value type is "Counter32", which internally corresponds to ASN_COUNTER type. The full list of supported types is ASN_COUNTER, ASN_COUNTER64, ASN_UINTEGER, ASN_UNSIGNED64, ASN_INTEGER, ASN_INTEGER64, ASN_FLOAT, ASN_DOUBLE, ASN_TIMETICKS, ASN_GAUGE, ASN_IPADDRESS, ASN_OCTET_STR 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 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'. Make sure the 'Host interface' field has your switch/router in it and change the 'Type' field to "SNMPv* agent". Enter the community (usually public) and 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

Enter the SNMP 'Port' as 161 and the 'Key' as something meaningful, e.g. SNMP-InOctets-Bps. Choose a custom multiplier if you want one and enter an 'Update interval' and 'History storage period' if you want them to be different from the default. Set the 'Type of information' to *Numeric (float)* and the 'Store value' to *Delta (speed per second)* (important, otherwise you will get cumulative values from the SNMP device instead of the latest change).



Now save the item and go to *Monitoring* → *Latest data* for your SNMP data!

Take note of specific options available for SNMPv3 items:

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.

Note:
Since Zabbix 2.2, SHA and AES protocols are supported for SNMPv3 authentication and privacy, in addition to MD5 and DES supported before that.

Example 1

General example:

Parameter	Description
Community	public
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
Community	public
Oid	MIB::sysUpTime.0
Key	router.uptime
Value type	Float
Units	uptime
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.

Special OID	Identifier	Description
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.
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 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 from a corresponding address that match the regular expression specified in **regexp**	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 from a corresponding address 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:
Multi-line regexp matching is not supported at this time.

Set the **Type of information** to be ’Log’ for the timestamps to be parsed. Note that other formats such as ’Numeric’ are also acceptable but might require a custom trap handler.

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):

1. StartSNMPTrapper=1
2. 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 to log the traps:

1. log traps to the trap file which will be read by Zabbix:
   log_enable = 1
   log_file = [TRAP FILE]
2. set the date-time format:
   date_time_format = %H:%M:%S %Y/%m/%d = [DATE TIME FORMAT]

Now format the traps for Zabbix to recognise 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 recognise 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
```

SNMP trap format

All customised perl trap receivers and SNMPTT trap configuration must format the trap in the following way: **[timestamp] [the trap, part 1] ZBXTRAP [address] [the trap, part 2]**, where

- [timestamp] - timestamp used for log items
- ZBXTRAP - header that indicates that a new trap starts in this line
- [address] - IP address used to find the host for this trap

Note that "ZBXTRAP" and "[address]" will be cut out from the message during processing. If the trap is formatted otherwise, Zabbix might parse the traps unexpectedly.

Example trap:
11:30:15 2011/07/27 .1.3.6.1.6.3.1.1.5.3 Normal "Status Events" localhost - ZBXTRAP 192.168.1.1 Link down on interface 2. Admin state: 1. Operational state: 2
This will result in the following trap for SNMP interface with IP=192.168.1.1:
11:30:15 2011/07/27 .1.3.6.1.6.3.1.1.5.3 Normal "Status Events" localhost - Link down on interface 2. Admin state: 1.

3 System requirements

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.

Attention:
The maximum log file size supported by Zabbix is 2 gigabytes. The log file must be rotated before reaching this limit.

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** - 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 "teststring000"
2. The received trap:
   15:48:18 2011/07/26 .1.3.6.1.6.3.1.1.5.3.0.33 Normal "General event" localhost - ZBXTRAP 127.0.0.1 127.0.0.1
3. Value for item TEST:
   15:48:18 2011/07/26 .1.3.6.1.6.3.1.1.5.3.0.33 Normal "General event" localhost - 127.0.0.1

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

- CentOS based SNMP trap tutorial on zabbix.org

4 IPMI checks

Overview

You can monitor the health and availability of Intelligent Platform Management Interface (IPMI) devices in Zabbix.
To perform IPMI checks Zabbix server must be initially configured with IPMI support.

IPMI is a standardized interface for remote "lights-out" or "out-of-band" management of computer systems. It allows to monitor hardware status directly from the so-called "out-of-band" management cards, independently from the operating system or whether the machine is powered on at all.

Zabbix IPMI monitoring works only for devices having IPMI support (HP iLO, DELL DRAC, IBM RSA, Sun SSP, etc).

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:

- For *Host interface* select the IPMI IP and port
- Select 'IPMI agent' as the *Type*
- Specify the *IPMI sensor* (for example 'FAN MOD 1A RPM' on Dell Poweredge)
- Enter an item key that is unique within the host (say, ipmi.fan.rpm)
- 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

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, get a copy of IPMI 2.0 specifications at http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-specifications.html (At the time of writing the newest document was http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/second-gen-interface-spec-v2.pdf)

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.zabbix.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.) Zabbix server/proxy will continue polling IPMI host until server/proxy restart.

5 Simple checks

1 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]
```

Note:
User name and *Password* fields in simple check item configuration are used for VMware monitoring items; ignored otherwise.

2 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** - one of min, max, avg (default)	If host is not available (timeout reached), the item will return 0. See also: table of default values.
net.tcp.service[service,<ip>,<port>]				

Key				
	Check if service is running and accepting TCP connections.	0 - service is down 1 - service is running	**service** - one of ssh, ntp, 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. 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.service[tcp,<ip>,port] for checks like these. Services **https** and **telnet** supported since Zabbix 2.0. Service **ntp** only works since Zabbix 2.0.15, 2.2.10 and 2.4.6, despite being available in earlier versions.
net.tcp.service.perf[service,<ip>,<port>]				

Key					
	Service performance check.	0 - service is down sec - number of seconds spent while connecting to the service	**service** - one of ssh, ntp, 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. Services **https** and **telnet** supported since Zabbix 2.0. Service **ntp** only works since Zabbix 2.0.15, 2.2.10 and 2.4.6, despite being available in earlier versions. Called tcp_perf before Zabbix 2.0.

Timeout processing

Zabbix will not process a simple check longer than the Timeout seconds defined in the Zabbix server/proxy configuration file.

3 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

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.

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.eventlog[<url>]	VMware event log.	Log	**url** - VMware service URL	
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 host name	
vmware.hv.cpu.usage[<url>,<uuid>]	VMware hypervisor processor usage (Hz).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.datastore.discovery[<url>,<uuid>]				

Key			
	Discovery of VMware hypervisor datastores.	JSON object	**url** - VMware service URL **uuid** - VMware hypervisor host name
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 host name **datastore** - datastore name **mode** - latency (default)
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 host name **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 host name
vmware.hv.hw.cpu.freq[<url>,<uuid>]			
	VMware hypervisor processor frequency (Hz).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name
vmware.hv.hw.cpu.model[<url>,<uuid>]			
	VMware hypervisor processor model.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name
vmware.hv.hw.cpu.num[<url>,<uuid>]			
	Number of processor cores on VMware hypervisor.	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name
vmware.hv.hw.cpu.threads[<url>,<uuid>]			
	Number of processor threads on VMware hypervisor.	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name

Key				
vmware.hv.hw.memory[<url>,<uuid>]	VMware hypervisor total memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.hw.model[<url>,<uuid>]	VMware hypervisor model.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.hw.uuid[<url>,<uuid>]	VMware hypervisor BIOS UUID.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.hw.vendor[<url>,<uuid>]	VMware hypervisor vendor name.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.memory.size.ballooned[<url>,<uuid>]	VMware hypervisor ballooned memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.memory.used[<url>,<uuid>]	VMware hypervisor used memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.network.in[<url>,<uuid>,<mode>]	VMware hypervisor network input statistics (bytes per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware hypervisor host name **mode** - bps (default)	Starting with Zabbix 2.4.4 *bps* mode value is correctly reported in *bytes per second* instead of *kilobytes per second* as it was before.
vmware.hv.network.out[<url>,<uuid>,<mode>]	VMware hypervisor network output statistics (bytes per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware hypervisor host name **mode** - bps (default)	Starting with Zabbix 2.4.4 *bps* mode value is correctly reported in *bytes per second* instead of *kilobytes per second* as it was before.
vmware.hv.perfcounter[<url>,<uuid>,<path>,<instance>]				

Key				
	VMware hypervisor performance counter value.	Integer [2]	**url** - VMware service URL **uuid** - VMware hypervisor host name **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.status[<url>,<uuid>]				
	VMware hypervisor status.	Integer: 0 - gray; 1 - green; 2 - yellow; 3 - red	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.uptime[<url>,<uuid>]				
	VMware hypervisor uptime (seconds).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.version[<url>,<uuid>]				
	VMware hypervisor version.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.vm.num[<url>,<uuid>]				
	Number of virtual machines on VMware hypervisor.	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
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 host name	
vmware.vm.cpu.num[<url>,<uuid>]				
	Number of processors on VMware virtual machine.	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name	
vmware.vm.cpu.usage[<url>,<uuid>]				
	VMware virtual machine processor usage (Hz).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name	

Key			
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 host name
vmware.vm.memory.size[<url>,<uuid>]			
	VMware virtual machine total memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.ballooned[<url>,<uuid>]			
	VMware virtual machine ballooned memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.compressed[<url>,<uuid>]			
	VMware virtual machine compressed memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.private[<url>,<uuid>]			
	VMware virtual machine private memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.shared[<url>,<uuid>]			
	VMware virtual machine shared memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.swapped[<url>,<uuid>]			
	VMware virtual machine swapped memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.usage.guest[<url>,<uuid>]			
	VMware virtual machine guest memory usage (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.usage.host[<url>,<uuid>]			
	VMware virtual machine host memory usage (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.net.if.discovery[<url>,<uuid>]			

Key				
	Discovery of VMware virtual machine network interfaces.	JSON object	**url** - VMware service URL **uuid** - VMware virtual machine host name	
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 host name **instance** - network interface instance **mode** - bps (default)/pps - bytes/packets per second	Starting with Zabbix 2.4.4 *bps* mode value is correctly reported in *bytes per second* instead of *kilobytes per second* as it was before.
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 host name **instance** - network interface instance **mode** - bps (default)/pps - bytes/packets per second	Starting with Zabbix 2.4.4 *bps* mode value is correctly reported in *bytes per second* instead of *kilobytes per second* as it was before.
vmware.vm.perfcounter[<url>,<uuid>,<path>,<instance>]				
	VMware virtual machine performance counter value.	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine host name **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 host name	
vmware.vm.storage.committed[<url>,<uuid>]				
	VMware virtual machine committed storage space (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name	

Key				
vmware.vm.storage.uncommitted[<url>,<uuid>]				
	VMware virtual machine uncommitted storage space (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name	
vmware.vm.storage.unshared[<url>,<uuid>]				
	VMware virtual machine unshared storage space (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name	
vmware.vm.uptime[<url>,<uuid>]				
	VMware virtual machine uptime (seconds).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name	
vmware.vm.vfs.dev.discovery[<url>,<uuid>]				
	Discovery of VMware virtual machine disk devices.	JSON object	**url** - VMware service URL **uuid** - VMware virtual machine host name	
vmware.vm.vfs.dev.read[<url>,<uuid>,<instance>,<mode>]				
	VMware virtual machine disk device read statistics (bytes/operations per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine host name **instance** - disk device instance **mode** - bps (default)/ops - bytes/operations per second	Starting with Zabbix 2.4.4 *bps* mode value is correctly reported in *bytes per second* instead of *kilobytes per second* as it was before.
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 host name **instance** - disk device instance **mode** - bps (default)/ops - bytes/operations per second	Starting with Zabbix 2.4.4 *bps* mode value is correctly reported in *bytes per second* instead of *kilobytes per second* as it was before.
vmware.vm.vfs.fs.discovery[<url>,<uuid>]				
	Discovery of VMware virtual machine file systems.	JSON object	**url** - VMware service URL **uuid** - VMware virtual machine host name	VMware Tools must be installed on the guest virtual machine.
vmware.vm.vfs.fs.size[<url>,<uuid>,<fsname>,<mode>]				

Key				
	VMware virtual machine file system statistics (bytes/percentages).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name **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 peformance 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] Since Zabbix 2.4.4, 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.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

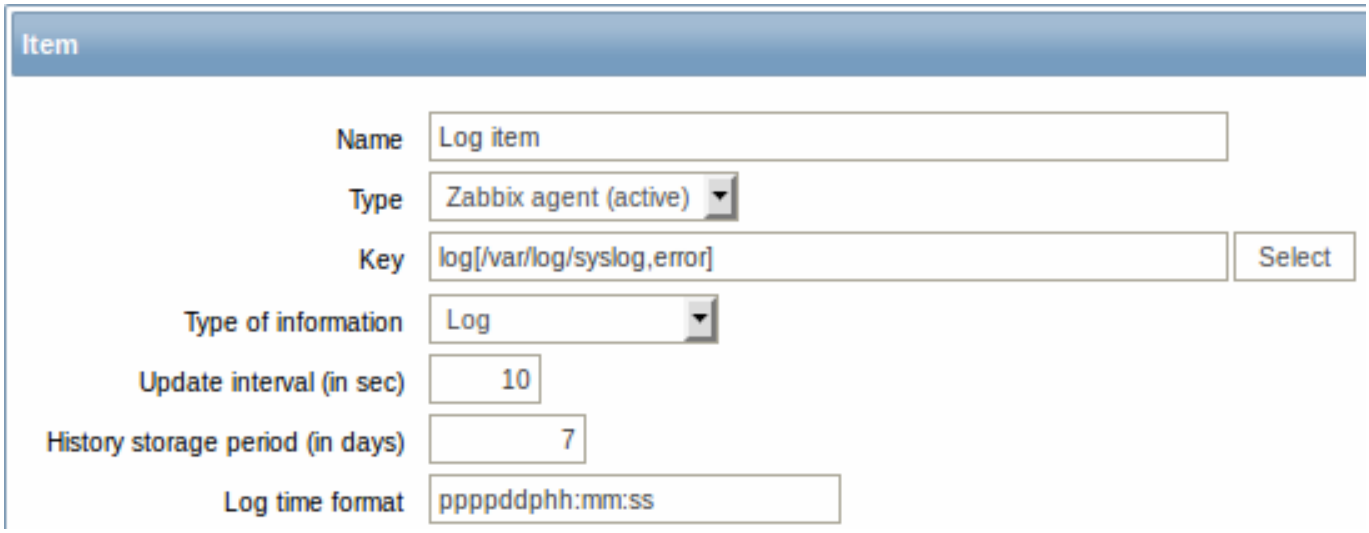
Verify agent parameters

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:



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'. See supported Zabbix agent item key section for details on using these item keys and their parameters.
Type of information	Select Log here.
Update interval (in sec)	The parameter defines how often Zabbix agent will check for any changes in the log file. Setting it to 1 second will make sure that you get new records as soon as possible.
Log time format	In this field you may optionally specify the pattern for parsing the log line timestamp. If left blank the timestamp will not be parsed. Supported placeholders: * **y**: *Year (0001-9999)* * **M**: *Month (01-12)* * **d**: *Day (01-31)* * **h**: *Hour (00-23)* * **m**: *Minute (00-59)* * **s**: *Second (00-59)* For example, consider the following line from the Zabbix agent log file: " 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 line. Log time format for this line would be "ppppp:yyyyMMdd:hhmmss". Note that "p" and ":" chars are just placeholders and can be anything but "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)
* On UNIX/GNU/Linux systems it is assumed that the file systems where log files are stored report inode
* 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 Microsft 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 ta
   * The inode numbers, file indexes and MD5 sums are internally collected by Zabbix agent. They are not t
   * Do not modify the last modification time of log files with 'touch' utility, do not copy a log file wi
   * If there are several matching log files for ''logrt[]'' item and Zabbix agent is following the most r
   * Zabbix **2.4.6** fixes an issue [[https://support.zabbix.com/browse/ZBX-9290|ZBX-9290]] (unexpected r
* 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) ar
* Whenever the log file becomes smaller than the log size counter known by the agent, the counter is reset
* If there are several matching files with the same last modification time in the directory, then the agen
* 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 overloadi
* To find the required string Zabbix will process 4 times more new lines than set in MaxLinesPerSecond. Th
* Additionally, log values are always limited to 50% of the agent send buffer size, even if there are no m
* In the absence of log items all agent buffer size is used for non-log values. When log values come in th
* For log file records longer than 256kB, only the first 256kB are matched against the regular expression
* Special note for "\" path separators: if file_format is "file\.log", then there should not be a "file" d
* Regular expressions for ''logrt'' are supported in filename only, directory regular expression matching
* On UNIX platforms a ''logrt[]'' item becomes NOTSUPPORTED if a directory where the log files are expecte
* On Microsoft Windows, if a directory does not exist the item will not become NOTSUPPORTED (for example,
* An absence of log files for ''logrt[]'' item does not make it NOTSUPPORTED. Errors of reading log files
* Zabbix agent log file can be helpful to find out why a ''log[]'' or ''logrt[]'' item became NOTSUPPORTED
```

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.

`output` 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.

7 Calculated items

1 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.

To use calculated items, choose the item type **Calculated**.

2 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. 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) not found
2. no data to calculate a function
3. division by zero
4. incorrect syntax used

Note:
Support for calculated items was introduced in Zabbix 1.8.1

3 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\",\"0\"]") / last("grpsum[\"video\",\"nginx_stat.
```

8 Internal checks

1 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

To use this item, choose the **Zabbix internal** item type.

Starting with Zabbix 2.4, internal checks are always processed by server or proxy regardless of host maintenance status.

Note:
Internal checks are processed by Zabbix pollers.

2 Supported checks

- Parameters without angle brackets are constants - for example, 'host' and 'available' in `zabbix[host,<type>,available]`. Use them in the item key *as is*.
- Values for items and item parameters that are "not supported on proxy" can only be gathered if the host is monitored by server. And vice versa, values "not supported on server" can only be gathered if the host is monitored by proxy.

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 table HISTORY	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! *(not supported on proxy)*
zabbix[history_log]			
	Number of values stored in table HISTORY_LOG	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! This item is supported starting with Zabbix **1.8.3.** *(not supported on proxy)*
zabbix[history_str]			
	Number of values stored in table HISTORY_STR	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! *(not supported on proxy)*
zabbix[history_text]			
	Number of values stored in table HISTORY_TEXT	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! This item is supported starting with Zabbix **1.8.3.** *(not supported on proxy)*
zabbix[history_uint]			

Key			
	Number of values stored in table HISTORY_UINT	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! This item is supported starting with Zabbix **1.8.3.** *(not supported on proxy)*
zabbix[host„maintenance]			
	Returns current maintenance status of a host.	0 - host in normal state, 1 - host in maintenance with data collection, 2 - host in maintenance 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 configuration data. The second parameter must be empty and is reserved for future use. This item is supported starting with Zabbix **2.4.0.**
zabbix[host,<type>,available]			

Key			
	Returns 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 starting with Zabbix **2.0.0.**
zabbix[hosts]	Number of monitored hosts	Integer.	This item is supported starting with Zabbix **2.2.0**.
zabbix[items]	Number of enabled items (supported and not supported)	Integer.	
zabbix[items_unsupported]	Number of not supported items	Integer.	
zabbix[java,,<param>]			

Key				
		Returns information associated with Zabbix Java gateway.	If <param> is **ping**, ”1” is returned. Can be used to check Java gateway availability using nodata() trigger function.	Valid values for <param> are: *ping*, *version* Second parameter must be empty and is reserved for future use.
			If <param> is **version**, version of Java gateway is returned. Example: ”2.0.0”.	This item is supported starting with Zabbix **2.0.0.**
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.	The following process types are currently supported: **alerter** - process for sending notifications *(not supported on proxy)* **configuration syncer** - process for managing in-memory cache of configura-tion data **data sender** - proxy data sender *(not supported on server)* **db watchdog** - sender of a warning message in case DB is not available *(not supported on proxy)* **discoverer** - process for discovery of devices **escalator** - process for escalation of actions *(not supported on proxy)* **heartbeat sender** - proxy heartbeat sender *(not supported on server)* **history syncer** - history DB writer **housekeeper** - process for removal of old historical data **http poller** - web

Key			
zabbix[proxy,<name>,<param>]			
	Access to Zabbix proxy related information.	Integer.	<name> - proxy name List of supported parameters (<param>): lastaccess - timestamp of last heart beat message received from proxy Example: => zab-bix[proxy,”Germany”,lastacc **fuzzytime()** trigger function can be used to check availability of proxies. Starting with Zabbix 2.4.0 this item is always processed by Zabbix server regardless of host location (on server or proxy).
zabbix[proxy_history]			
	Number of values in proxy history table waiting to be sent to the server	Integer.	This item is supported starting with Zabbix **2.2.0**. *(not supported on server)*
zabbix[queue,<from>,<to>]			

Key			
	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. Parameters `from` and `to` are supported starting with Zabbix **1.8.3.**
zabbix[rcache,<cache>,<mode>]			
	Availability statistics of Zabbix configuration cache.	Integer (for size); float (for percentage).	Cache: **buffer** Mode: **total** - total size of buffer **free** - size of free buffer **pfree** - percentage of free buffer **used** - size of used buffer
zabbix[requiredperformance]			
	Required performance of the Zabbix server or Zabbix proxy, in new values per second expected.	Float.	Approximately correlates with "Required server performance, new values per second" in *Reports* → *Status of Zabbix*. This item is supported starting with Zabbix **1.6.2.**
zabbix[trends]			
	Number of values stored in table TRENDS	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! *(not supported on proxy)*

Key			
zabbix[trends_uint]			
	Number of values stored in table TRENDS_UINT	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! This item is supported starting with 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).	Mode: **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 This item is supported starting with Zabbix **2.2.0**. *(not supported on proxy)*
zabbix[vcache,cache,<parameter>]			

Key			
	Effectiveness statistics of Zabbix value cache.	Integer.	Parameter: **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) This item is supported starting with Zabbix **2.2.0**. *(not supported on proxy)* You may use this key with a *Delta (speed per second)* store value in order to get values per second statistics.
zabbix[vmware,buffer,<mode>]	Availability statistics of Zabbix vmware cache.	Integer (for size); float (for percentage).	Mode: **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 This item is supported starting with Zabbix **2.2.0**.

Key					
zabbix[wcache,<cache>,<mode>]					
	Statistics and availability of Zabbix write cache.				
	Cache	**Mode**			
	values	all	Total number of values processed by Zabbix server or Zabbix proxy, except unsupported items.	Integer.	Counter. You may use this key with a *Delta (speed per second)* store value 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 items.	Integer.	Counter.
		text	Number of processed text items.	Integer.	Counter.
		not supported	Number of processed unsupported items.	Integer.	Counter. *Not supported* mode is supported starting with Zabbix **1.8.6.**
	history	pfree	Percentage of free history buffer.	Float.	History cache stores item and timestamp information for all item types as well as value for the numeric types. A low number indicates performance problems on the database side.

Key					
		free	Size of free history buffer.	Integer.	
		total	Total size of history buffer.	Integer.	
		used	Size of used history buffer.	Integer.	
	trend	pfree	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)*
	text	pfree	Percentage of free text history buffer.	Float.	Text history cache is used for storing character, text or log history data - item and timestamp information for these values is still stored in the history cache.
		free	Size of free text history buffer.	Integer.	
		total	Total size of text history buffer.	Integer.	
		used	Size of used text history buffer.	Integer.	

9 SSH checks

1 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.

Attention:
The minimum supported libssh2 library version is 1.0.0.

2 Configuration

2.1 Passphrase authentication

SSH checks provide two authentication methods, a user/password pair and key-file based.

If you do not intend to use keys, no additional configuration is required, besides linking libssh2 to Zabbix, if you're building from source.

2.2 Key file authentication

To use key based authentication for SSH items, certain changes to the server configuration are required.

Open the Zabbix server configuration file (zabbix_server.conf) as `root` and look for the following line:

```
# SSHKeyLocation=
```

Uncomment it and set full path to a folder where public and private keys will be located:

```
SSHKeyLocation=/home/zabbix/.ssh
```

Save the file and restart zabbix_server afterwards.

/home/zabbix here is the home directory for the *zabbix* user account and *.ssh* is a directory where by default public and private keys will be generated by a ssh-keygen command inside the home directory.

Usually installation packages of zabbix-server from different OS distributions create the *zabbix* user account with a home directory in not very well-known places (as for system accounts). For example, for CentOS it's */var/lib/zabbix*, for Debian it's */var/run/zabbix*.

Before starting to generate the keys, an approach to reallocate the home directory to a better known place (intuitively expected) could be considered. This will correspond with the *SSHKeyLocation* Zabbix server configuration parameter mentioned above.

These steps can be skipped if *zabbix* account has been added manually according to the installation section because in this case most likely the home directory is already located at */home/zabbix*.

To change the setting for the *zabbix* user account all working processes which are using it have to be stopped:

```
# service zabbix-agent stop
# service zabbix-server stop
```

To change the home directory location with an attempt to move it (if it exists) a command should be executed:

```
# usermod -m -d /home/zabbix zabbix
```

It's absolutely possible that a home directory did not exist in the old place (in the CentOS for example), so it should be created at the new place. A safe attempt to do that is:

```
# 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 now can be started again:

```
# service zabbix-agent start
# service zabbix-server start
```

Now steps to generate public and private keys can be performed by a command:

```
# sudo -u zabbix ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/zabbix/.ssh/id_rsa):
Created directory '/home/zabbix/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
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: public and private keys (*id_rsa.pub* and *id_rsa* respectively) have been generated by default in the */home/zabbix/.ssh* directory which corresponds to the Zabbix server *SSHKeyLocation* configuration parameter.

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.

2.3 Shell configuration form

This step should be performed only once for every host that will be monitored by SSH checks.

By using the following command the **public** key file can be installed on a remote host *10.10.10.10* so that then SSH checks can be performed with a *root* account:

```
# 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:
Now try logging into the machine, with "ssh 'root@10.10.10.10'", and check in:
  .ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.
```

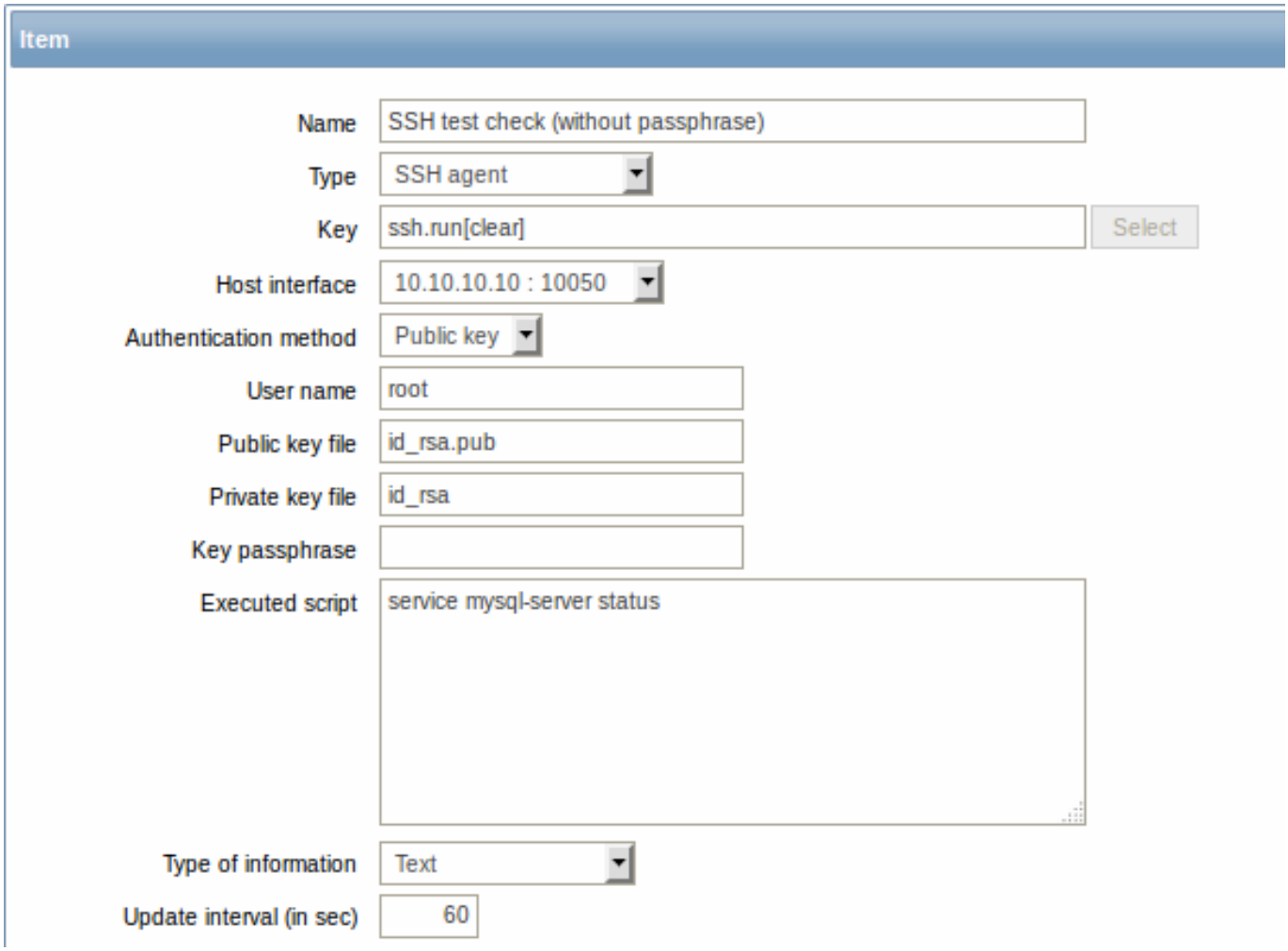
Now it's possible to check the SSH login using the default private key (*/home/zabbix/.ssh/id_rsa*) for *zabbix* user account:

```
# sudo -u zabbix ssh root@10.10.10.10
```

If the login is successful, then the configuration part in the shell is finished and remote SSH session can be closed.

2.4 Item configuration

Actual command(s) to be executed must be placed in the **Executed script** field in the item configuration.
Multiple commands can be executed one after another by placing them on a new line. In this case returned values also will be formatted as multi lined.



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

1 Overview

Telnet checks are performed as agent-less monitoring. Zabbix agent is not needed for Telnet checks.

2 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 formated as multi lined.

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

1 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 will look in the directory defined as the location for external scripts (parameter 'ExternalScripts' in Zabbix server configuration file) and execute the command. The command will be executed as the user Zabbix server runs as, so any access permissions or environment variables should be handled in a wrapper script, if necessary, and permissions on the command should allow that user to execute it. Only commands in the specified directory are available for execution.

Warning:
Do not overuse external checks! As each script requires starting a fork process by Zabbix server, running many scripts can decrease Zabbix performance a lot.

2 Usage example

Executing the script **check_oracle.sh** with the first parameters "-h". The second parameter will be replaced by IP address or DNS name, depending on the selection in the host properties.

```
check_oracle.sh["-h","{HOST.CONN}"]
```

Assuming host is configured to use IP address, Zabbix will execute:

```
check_oracle.sh "-h" "192.168.1.4"
```

3 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 has no permissions to execute it, 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.

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

Item function	Description
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*.

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)*.

An aggregate item may become unsupported in several cases:

- none of the referenced items is found (which may happen if the item key is incorrect, none of the items exists or all included groups are incorrect)
- no data to calculate a function

Usage examples

Examples of keys for aggregate checks:

Example 1

Total disk space of host group 'MySQL Servers'.

```
grpsum["MySQL Servers","vfs.fs.size[/,total]",last,0]
```

Example 2

Average processor load of host group 'MySQL Servers'.

```
grpavg["MySQL Servers","system.cpu.load[,avg1]",last,0]
```

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,0]
```

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:

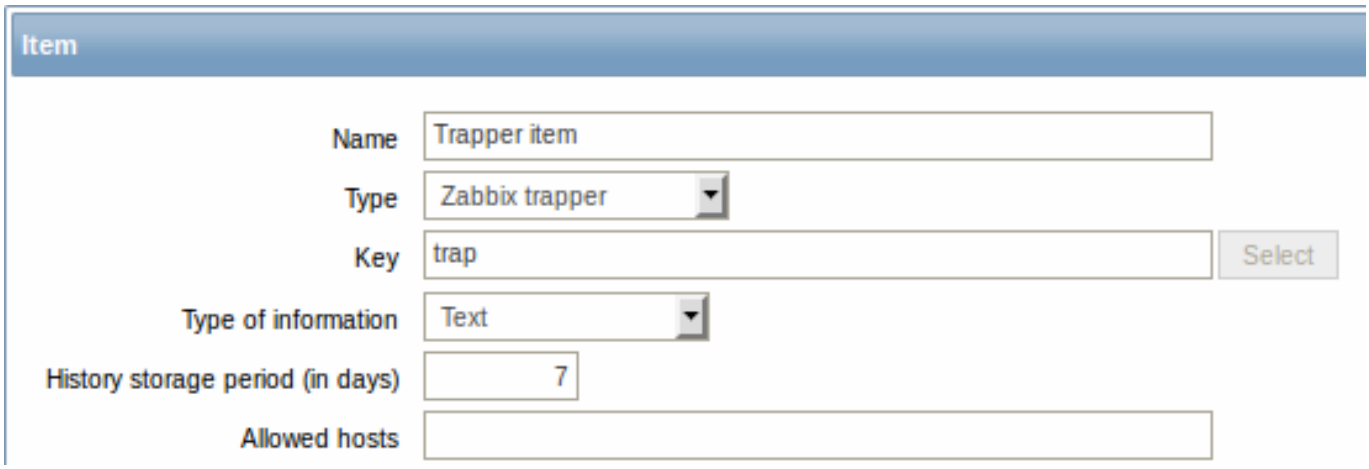
- have a trapper item set up in Zabbix
- send in the data into Zabbix

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



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	If specified, the trapper will accept incoming data only from this comma-delimited list of hosts. Hosts are identified by IP address/DNS name. For example: *Single IP*: 192.168.1.33 *List of IP addresses*: 192.168.56.5, 192.168.56.6, 192.168.56.7 *Single DNS name*: testzabbix.zabbix.com *List of DNS names*: testzabbix, testzabbix.zabbix.com, testzabbix1.zabbix.com Spaces and user macros are allowed in this field since Zabbix 2.2.0.

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*:

Trapper item	2015-01-20 10:2...	test value	-	History

Timestamps

If values are sent using `zabbix_sender` from a file with timestamps, then these timestamps will be adjusted to match server time. For instance, if an item's timestamp is "10:30:50", the current time on `zabbix_sender`'s machine is "10:40:03", and the current time on Zabbix server's machine is "10:40:05", then the item's value will be stored in the database with a timestamp of "10:30:52".

Similarly, if a value is first sent to Zabbix proxy, which later sends it to Zabbix server, the timestamp will be first adjusted to match Zabbix proxy time, and then it will be adjusted to match Zabbix server time.

14 JMX monitoring

1 Overview

JMX monitoring can be used to monitor JMX counters of a Java application.

Zabbix 2.0 added native support for JMX monitoring by introducing a new Zabbix daemon called "Zabbix Java gateway".

When Zabbix server wants to know the value of a particular JMX counter on a host, it asks the Zabbix **Java gateway**, which in turn uses the JMX management API to query the application of interest remotely.

> **Note:**
> For more details on Zabbix Java gateway, including where to get it and how to set it up see this section of the manual.

2 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 \
-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 \
-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.

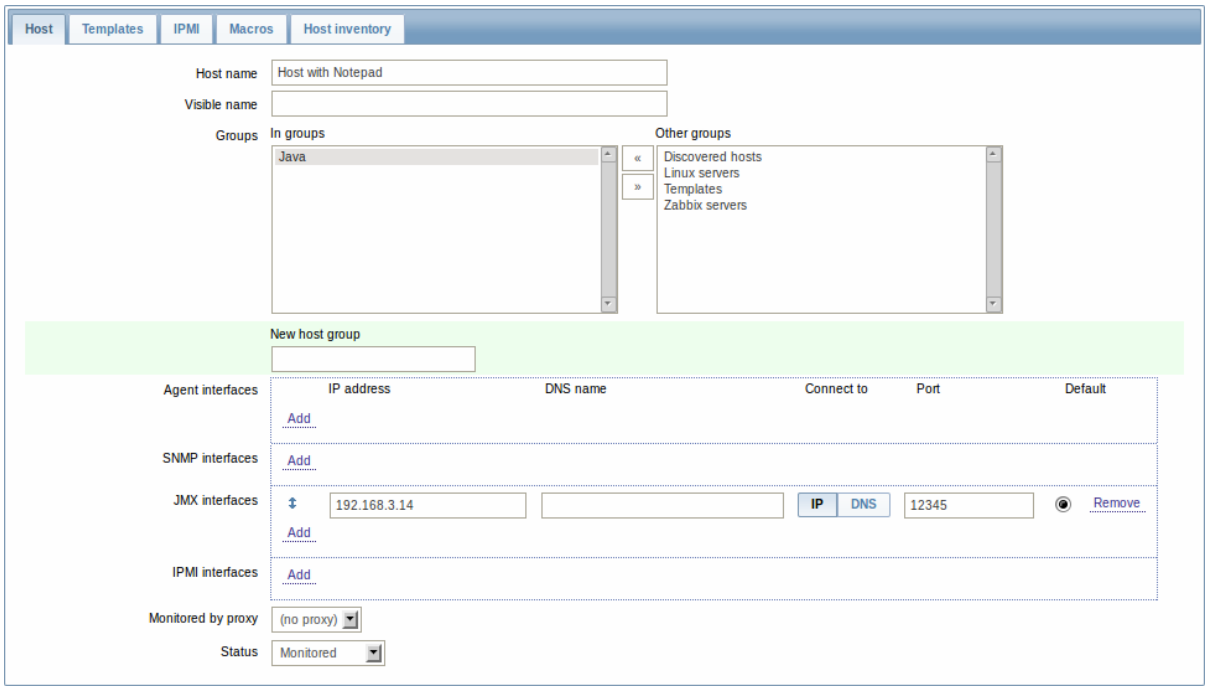
See Monitoring and Management Using JMX for a detailed description.

3 Configuring JMX interfaces and items in Zabbix GUI

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:



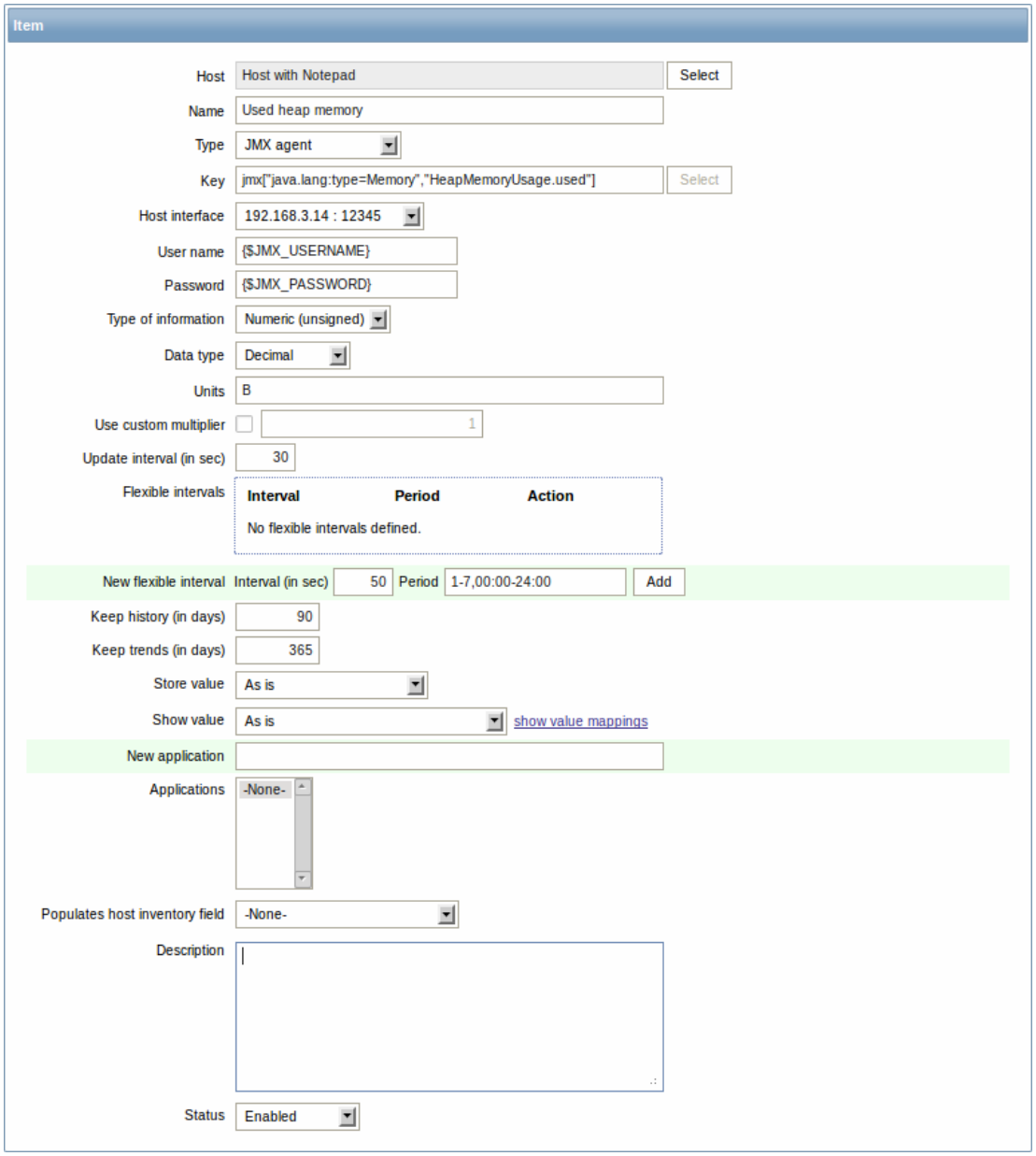
Adding JMX agent item

For each JMX counter you are interested in you add an item of type **JMX agent** attached to that interface. If you have configured authentication on your Java application, then you also specify username and password.

The key in the screenshot below says `jmx["java.lang:type=Memory","HeapMemoryUsage.used"]`. The key consists of 2 parameters:

- object name - which represents 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.



If you wish to monitor a Boolean counter that is either ”true” or ”false”, then you specify type of information as ”Numeric (unsigned)” and data type as ”Boolean”. Server will store Boolean values as 1 or 0, respectively.

JMX item keys in more detail

Simple attributes

An MBean object name is nothing but a string which you define in your Java application. An attribute name, on the other hand, can be more complex. In case an attribute returns primitive data type (an integer, a string etc.) there is nothing to worry about, the key will look like this:

```
jmx[com.example:Type=Hello,weight]
```

In this example an object name is ”com.example:Type=Hello”, attribute name is ”weight” and probably the returned value type should be ”Numeric (float)”.

Attributes returning composite data

It becomes more complicated when your attribute returns composite data. For example: your attribute name is ”apple” and it returns a hash representing its parameters, like ”weight”, ”color” etc. Your key may look like this:

```
jmx[com.example:Type=Hello,apple.weight]
```

This is how an attribute name and a hash key are separated, by using a dot symbol. Same way, if an attribute returns nested composite data the parts are separated by a dot:

```
jmx[com.example:Type=Hello,fruits.apple.weight]
```

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 should be escaped as well:

```
jmx[com.example:type=Hello,c:\\documents]
```

If the object name or attribute name contains spaces or commas double-quote it:

```
jmx["com.example:Type=Hello","fruits.apple.total weight"]
```

This is actually all there is to it. Happy JMX monitoring!

15 ODBC monitoring

1 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.

2 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.

3 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
```

4 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>
```

5 Compiling Zabbix with ODBC support

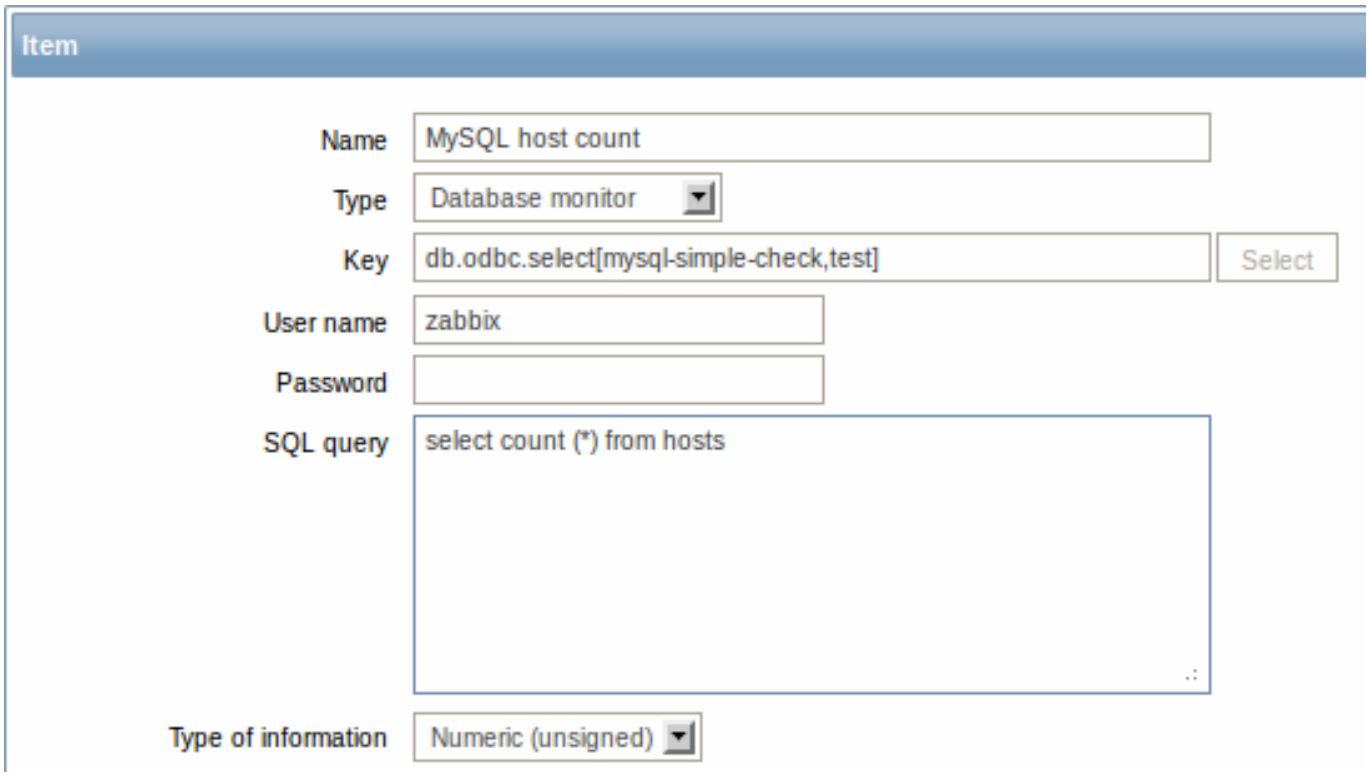
To enable ODBC support, Zabbix should be compiled with the following flag:

```
  --with-unixodbc[=ARG]   use odbc driver against unixODBC package
```

Note:
See more about Zabbix installation from the source code.

6 Item configuration in Zabbix frontend

Configure a database monitoring item:



Specifically for database monitoring items you must enter:

Type	Select *Database monitor* here.
Key	Enter **db.odbc.select**[unique_description,data_source_name] The unique description will serve to identify the item in triggers etc. The data source name (DSN) must be set as specified in odbc.ini.
User name	Enter the database user name (optional if user is specified in odbc.ini)
Password	Enter the database user password (optional if password is specified in odbc.ini)
SQL query	Enter the SQL query
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.

7 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 query must return one value only.
- 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.
- The SQL command must begin with `select`.
- The SQL command mustn't contain any line breaks.
- See also known issues for ODBC checks

8 Error messages

Starting from Zabbix 2.0.8 the ODBC error messages are structured into fields to provide more detailed information. Example:

```
Cannot execute ODBC query:[SQL_ERROR]:[42601][7][ERROR: syntax error at or near ";"; Error while executing
------------------------  ---------  -----  |  ------------------------------------------------------------
```

```
|                  |         |    `- Native error code            `- error message.
|                  |         `-SQLState
`- Zabbix message  `- ODBC return code
```

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 as far as the length limit allows.

3 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 inventory. No trigger functions will be evaluated.

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.

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 the 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.

4 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. **/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:
Unless UnsafeUserParameters agent daemon configuration option is enabled, it is not allowed to pass flexible parameters containing these symbols: \ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @. Additionally, newline is not allowed either.

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 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.

5 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.

Zabbix 2.2 comes with support of loadable modules 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 five functions in the Zabbix module API, two of which are mandatory and the other three are optional.

2.1 Mandatory interface

The two mandatory functions are **zbx_module_api_version()** and **zbx_module_init()**:

```
int zbx_module_api_version(void);
```

This function should return the API version implemented by this module. Currently, there is only one version, ZBX_MODULE_API_VERSION_ONE (defined to 1), so this function should return this constant.

```
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.

These two functions are mandatory in a sense that if any of them is absent from module API or any of them returns an unacceptable result when called for any module in the list of modules to load Zabbix will not start.

2.2 Optional interface

The three optional functions are **zbx_module_item_list()**, **zbx_module_item_timeout()**, **zbx_module_uninit()**:

```
ZBX_METRIC  *zbx_module_item_list(void);
```

This function should return a list of items supported by the module. Zabbix reads the list of supported items only once on startup. New items cannot be added during the operation. 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. If this function is absent from the module API, Zabbix will unload the module and proceed with loading other modules.

```
void    zbx_module_item_timeout(int timeout);
```

This function is used by Zabbix to specify the timeout settings in Zabbix configuration file that the module should obey. Here, the "timeout" parameter is in seconds.

```
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 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. Another useful header is **include/sysinc.h**, which performs the inclusion of the necessary system headers, which itself helps include/module.h to work properly.

In order for include/module.h and include/sysinc.h to be included, the **./configure** command (without arguments) should first be run in the root of Zabbix source tree. This will create **include/config.h** file, which include/sysinc.h relies upon. (If you obtained Zabbix source code as a Subversion repository checkout, the ./configure script does not exist yet and the **./bootstrap.sh** command should first be run to generate it.)

With this information in mind, everything is ready for the module to be built. The module should include **sysinc.h** and **module.h**, and the build script should make sure that these two 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. 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=dummy.so
```

Upon agent startup it will load the mariadb.so, apache.so, kernel.so and dummy.so modules from the /usr/local/lib/zabbix/agent directory. It will fail if a module is missing, in case of bad permissions or if a shared library is not a Zabbix module.

4 Frontend configuration

Loadable modules are supported by Zabbix agent, server and proxy. Therefore, item type in Zabbix frontend depends on where the module is loaded. If the module is loaded into the agent, then the item type should be ”Zabbix agent” or ”Zabbix agent (active)”. If the module is loaded into server or proxy, then the item type should be ”Simple check”.

5 Dummy module

Zabbix 2.4 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) 2001-2013 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 "sysinc.h"
####include "module.h"

/* the variable keeps timeout setting for item processing */
static int    item_timeout = 0;
```

```
int	zbx_module_dummy_ping(AGENT_REQUEST *request, AGENT_RESULT *result);
int	zbx_module_dummy_echo(AGENT_REQUEST *request, AGENT_RESULT *result);
int	zbx_module_dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result);

static ZBX_METRIC keys[] =
/* KEY               FLAG           FUNCTION                TEST PARAMETERS */
{
    {"dummy.ping",   0,             zbx_module_dummy_ping,  NULL},
    {"dummy.echo",   CF_HAVEPARAMS, zbx_module_dummy_echo,  "a message"},
    {"dummy.random", CF_HAVEPARAMS, zbx_module_dummy_random,"1,1000"},
    {NULL}
};

/******************************************************************************
*                                                                            *
* Function: zbx_module_api_version                                           *
*                                                                            *
* Purpose: returns version number of the module interface                    *
*                                                                            *
* Return value: ZBX_MODULE_API_VERSION_ONE - the only version supported by   *
*               Zabbix currently                                             *
*                                                                            *
******************************************************************************/
int	zbx_module_api_version()
{
    return ZBX_MODULE_API_VERSION_ONE;
}

/******************************************************************************
*                                                                            *
* 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()
{
    return keys;
}

int	zbx_module_dummy_ping(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    SET_UI64_RESULT(result, 1);

    return SYSINFO_RET_OK;
}
```

```
int    zbx_module_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: zbx_module_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->timeout - processing should not take longer than      *
*                                 this number of seconds                      *
*              request->params[N-1] - pointers to item key parameters         *
*                                                                             *
*             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.                              *
*                                                                             *
******************************************************************************/
int    zbx_module_dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    char  *param1, *param2;
    int   from, to;

    if (request->nparam != 2)
    {
        /* 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 for simplicity sake */
    from = atoi(param1);
    to = atoi(param2);

    if (from > to)
```

```
    {
        SET_MSG_RESULT(result, strdup("Incorrect range given"));
        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()
{
    /* 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()
{
    return ZBX_MODULE_OK;
}
```

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.

6 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, see Windows-specific item keys.

In order to get a full list of performance counters available for monitoring, you may run:

```
typeperf -qx
```

Numeric representation

As the naming of performance counters may differ on different Windows servers, depending on local settings, it introduces a certain problem when creating a template for monitoring several Windows machines having different locales.

At the same time every performance counter can also be referred to by its numeric form, which is unique and exactly the same regardless of language settings, so you might use the numeric representation instead of strings.

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.

Troubleshooting

Sometimes Zabbix agent cannot retrieve performance counter values in Windows 2000-based systems, because the pdh.dll file is outdated. It shows up as failure messages in Zabbix agent and server log files. In this case pdh.dll should be updated to a newer 5.0.2195.2668 version.

7 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
- Select *Mass update* from the dropdown below and click on *Go*
- Mark the checkboxes of the attributes to update
- Enter new values for the attributes and click on *Update*

Mass update

Type	☐	Original
Host interface	☐	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
Data type	☐	Original
Units	☐	Original
Authentication method	☐	Original
User name	☐	Original
Public key file	☐	Original
Private key file	☐	Original
Password	☐	Original
Custom multiplier (0 - Disabled)	☐	Original
Update interval (in sec)	☑	30
Flexible intervals	☐	Original
History storage period (in days)	☑	7
Trend storage period (in days)	☐	Original
Status	☐	Original
Log time format	☐	Original
Store value	☐	Original
Show value	☐	Original
Allowed hosts	☐	Original
Replace applications	☐	Original
Add new or existing applications	☑	type here to search Select
Description	☐	Original

Replace applications will remove the item from any existing applications and replace those with the one(s) specified in this field.

Add new or existing applications allows to specify additional applications from the existing ones or enter completely new applications for the items.

Both these fields are auto-complete - starting to type in them 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.

8 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/jabber etc.

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'

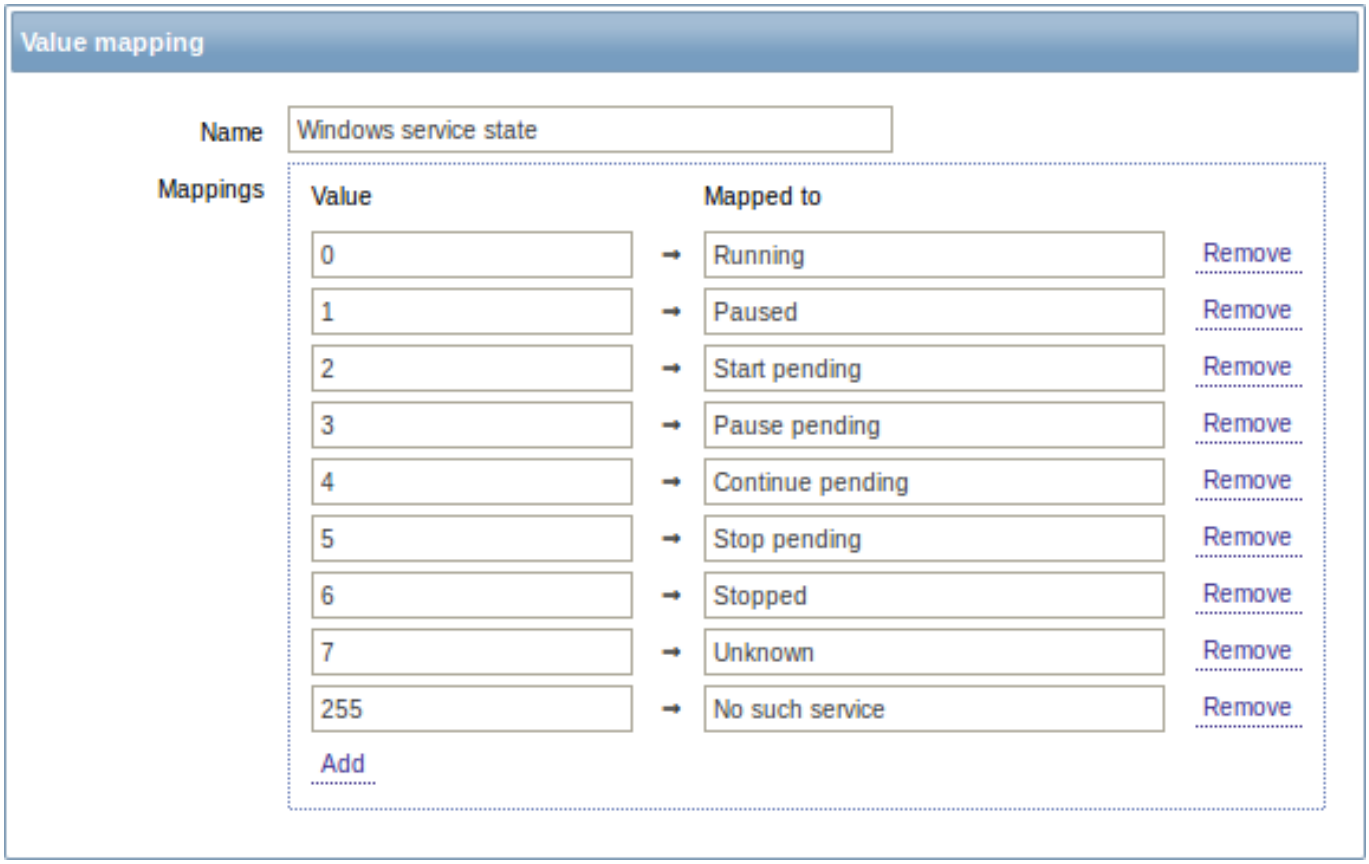
Thus, when configuring items you can use a value map to "humanize" the way an item value will be displayed. To do that, you refer to the name of a previously defined value map in the *Show value* field.

Note:
Value mapping can be used with items having *Numeric (unsigned)*, *Numeric (float)* and *Character* type of information.

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)



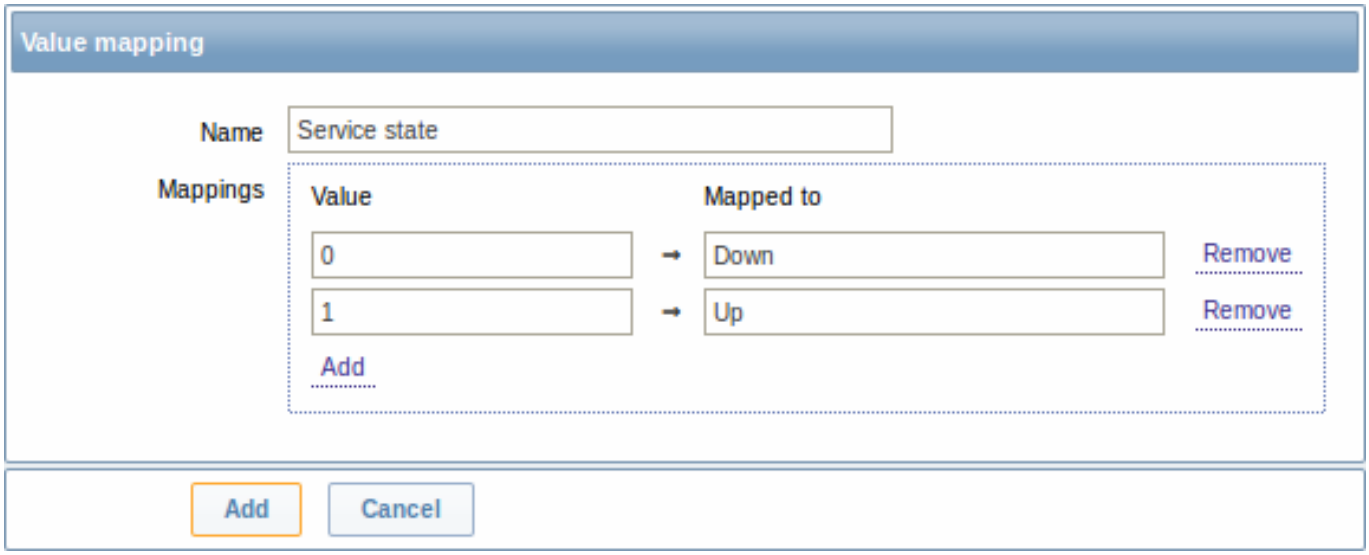
Parameters of a value map:

Parameter	Description
Name	Unique name of a set of value mappings.
Mappings	Individual mappings - pairs of numeric values and their string representations.

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).

Agent ping	2014-09-23 10:52...	Up (1)	-	Graph

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:

Agent ping	2014-09-23 10:54...	1	-	Graph

So in this case you would either have to guess what the '1' stands for or do a search of documentation to find out.

9 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.

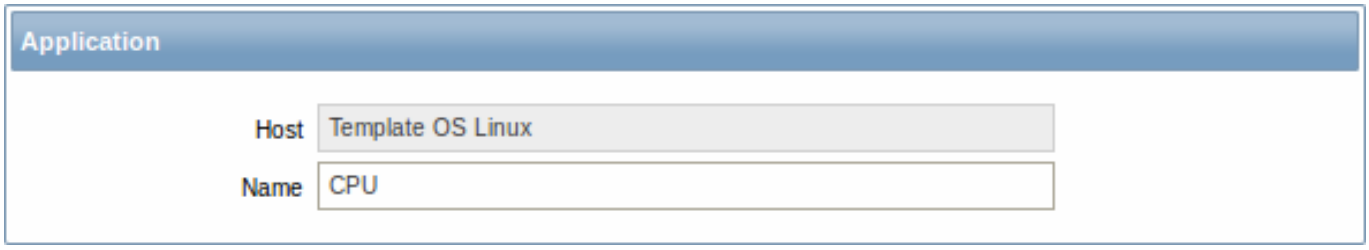
If you are using applications, then in *Monitoring* → *Latest data* you will see items and web scenarios grouped under their respective applications.

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.

10 Queue

Overview

The queue displays items that are waiting for a refresh. The queue is just a **logical** representation of data. There is no IPC queue or any other queue mechanism in Zabbix.

Items monitored by proxies are also included in the queue - they will be counted as queued for the proxy history data update period.

Only items with scheduled refresh times are displayed in the queue. This means that the following item types are excluded from the queue:

- log, logrt and event log active Zabbix agent items
- SNMP trap items
- trapper items
- web monitoring items

Statistics shown by the queue is a good indicator of the performance of Zabbix server.

The queue is retrieved directly from Zabbix server using JSON protocol. The information is available only if Zabbix server is running.

Reading the queue

To read the queue, go to *Administration* → *Queue*. *Overview* should be selected in the dropdown to the right.

Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	0	1	0	0	1	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
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

The picture here is generally ”green” so we may assume that the server is doing fine.

The queue shows one item waiting for 10 seconds and one for 5 minutes. Nice, it would be great to know what items these are.

To do just that, select *Details* in the dropdown in the upper right corner. Now you can see a list of those delayed items.

Scheduled check	Delayed by	Host	Name
2015-01-23 14:08:31	15s	Remote proxy: New host	Used disk space on /
2015-01-23 14:08:32	14s	Remote proxy: New host	Processor load

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.

Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	0	1	0	1	0	45
Zabbix agent (active)	0	0	0	0	0	0

Is the agent down?

Queue item

A special internal item **zabbix[queue,<from>,<to>]** can be used to monitor the health of the queue in Zabbix. It will return the number of items delayed by the set amount of time. For more information see Internal items.

11 Value cache

Overview

To make the calculation of trigger expressions, calculated/aggregate items and some macros much faster, starting with 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.

3 Triggers

Overview

Triggers are logical expressions that ”evaluate” data gathered by items and represent the current system state.

While items are used to gather system data, it is highly impractical to follow these data all the time waiting for a condition that is alarming or deserves attention. The job of ”evaluating” data can be left to trigger expressions.

Trigger expressions allow to define a threshold of what state of data is ”acceptable”. Therefore, should the incoming data surpass the acceptable state, a trigger is ”fired” - or changes status to PROBLEM.

A trigger may have the following status:

VALUE	DESCRIPTION
OK	This is a normal trigger state. Called FALSE in older Zabbix versions.
PROBLEM	Normally means that something happened. For example, the processor load is too high. Called TRUE in older Zabbix versions.

Trigger status (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 *timer* 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.

1 Configuring a trigger

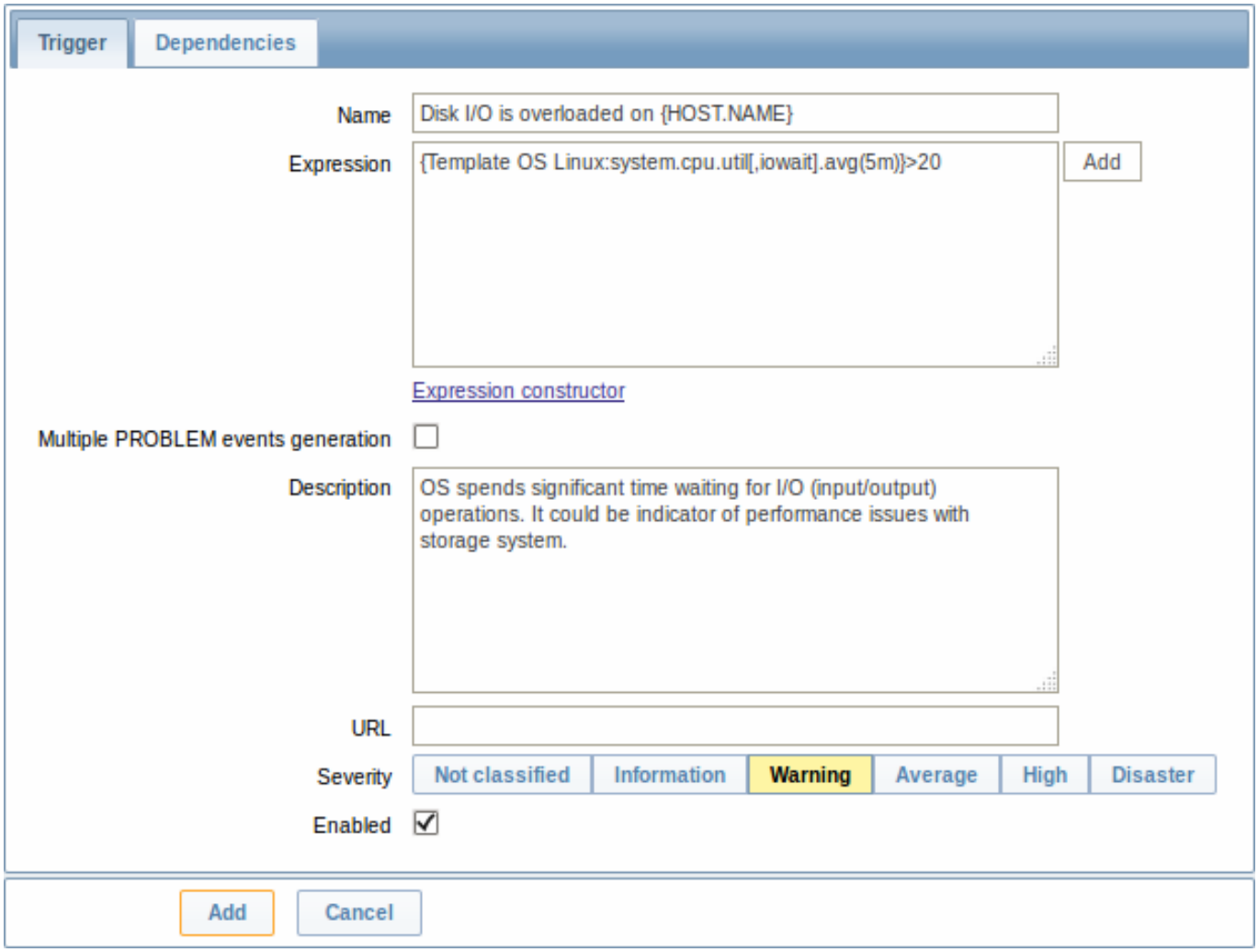
Overview

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.



Parameter	Description
Name	Trigger name. The name may contain the supported macros: {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {ITEM.VALUE}, {ITEM.LASTVALUE} and {$MACRO}. **$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
Expression	Logical expression used for calculating the trigger state.
Multiple PROBLEM events generation	By checking this option you can set that an event is generated upon *every* 'Problem' evaluation of the trigger.
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.
URL	If not empty, the URL entered here is available as a link when clicking on the trigger name in *Monitoring* → *Triggers*. One macro may be used in the trigger URL field - {TRIGGER.ID}.
Severity	Set the required trigger severity by clicking the buttons.
Enabled	Unchecking this box will disable the trigger if required.

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.

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>
```

Functions

Trigger functions allow to reference the collected values, current time and other factors.

A complete list of supported functions is available.

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 last 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.

Operators

The following operators are supported for triggers **(in descending priority of execution)**:

PRIORITY	OPERATOR	DEFINITION
1	**-**	Unary minus
2	**not**	Logical NOT
3	*** |**Multiplication** | | |/**	Division
4	**+**	Arithmetical plus
	-	Arithmetical minus
5	**<**	Less than. The operator is defined as: A<B ⇔ (A<=B-0.000001)
	<=	Less than or equal to.
	>	More than. The operator is defined as: A>B ⇔ (A>=B+0.000001)
	>=	More than or equal to.
6	**=**	Is equal. The operator is defined as: A=B ⇔ (A>B-0.000001) and (A<B+0.000001)
	<>	Not equal. The operator is defined as: A<>B ⇔ (A<=B-0.000001) or (A>=B+0.000001)
7	**and**	Logical AND
8	**or**	Logical OR

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;
- **and** shall yield '1' if both of its operands compare unequal to '0'; otherwise, it yields '0';
- **or** shall yield '1' if either of its operands compare unequal to '0'; otherwise, it yields '0';
- The result of the logical negation operator **not** is '0' if the value of its operand compares unequal to '0'; '1' if the value of its operand compares equal to '0'.

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.zabbix.com

```
{www.zabbix.com:system.cpu.load[all,avg1].last()}>5
```

'www.zabbix.com:system.cpu.load[all,avg1]' gives a short name of the monitored parameter. It specifies that the server is 'www.zabbix.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.zabbix.com is greater than 5.

Example 2

www.zabbix.com is overloaded

```
{www.zabbix.com:system.cpu.load[all,avg1].last()}>5 or {www.zabbix.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.zabbix.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.zabbix.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.zabbix.com:net.tcp.service[smtp].last()}=0 and {smtp2.zabbix.com:net.tcp.service[smtp].last()}=0
```

The expression is true when both SMTP servers are down on both smtp1.zabbix.com and smtp2.zabbix.com.

Example 6

Zabbix agent needs to be upgraded

Use of function str():

```
{zabbix.zabbix.com:agent.version.str("beta8")}=1
```

The expression is true if Zabbix agent has version beta8 (presumably 1.0beta8).

Example 7

Server is unreachable

```
{zabbix.zabbix.com:icmpping.count(30m,0)}>5
```

The expression is true if host "zabbix.zabbix.com" is unreachable more than 5 times in the last 30 minutes.

Example 8

No heartbeats within last 3 minutes

Use of function nodata():

```
{zabbix.zabbix.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 status to true, only at night (00:00-06:00) time.

Example 10

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.

Example 11

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 12

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 13

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.

Hysteresis

Sometimes a trigger must have different conditions for different states. For example, we would like to define a trigger which would become PROBLEM when server room temperature is higher than 20C while it should stay in the state until temperature will not become lower than 15C.

In order to do this, we define the following trigger:

Example 1

Temperature in server room is too high

```
({TRIGGER.VALUE}=0 and {server:temp.last()}>20) or
({TRIGGER.VALUE}=1 and {server:temp.last()}>15)
```

Note the use of a macro {TRIGGER.VALUE}. The macro returns current trigger value.

Example 2

Free disk space is too low

Problem: it is less than 10GB for last 5 minutes

Recovery: it is more than 40GB for last 10 minutes

```
({TRIGGER.VALUE}=0 and {server:vfs.fs.size[/,free].max(5m)}<10G) or
({TRIGGER.VALUE}=1 and {server:vfs.fs.size[/,free].min(10m)}<40G)
```

Note use of {TRIGGER.VALUE} macro. The macro returns current trigger value.

3 Trigger dependencies

Overview

Sometimes the availability of one host depends on another. A server that is behind some router will become unreachable if the router goes down. With triggers configured for both, you might get notifications about two hosts down - while only the router was the guilty party.

This is where some dependency between hosts might be useful. With dependency set notifications of the dependants 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 dependant 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.

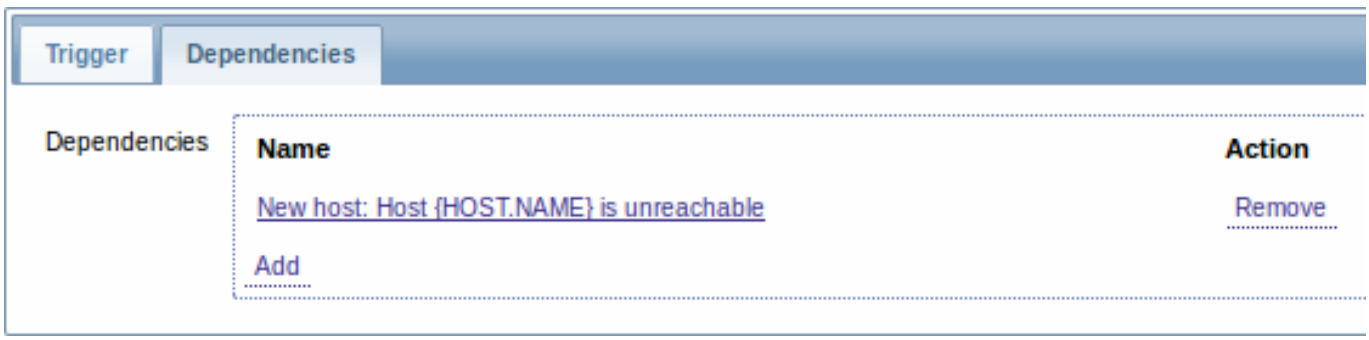
It is important to note that events/actions for dependent triggers will not be suppressed if the trigger they depend on is disabled, has disabled item or disabled item host.

Also:

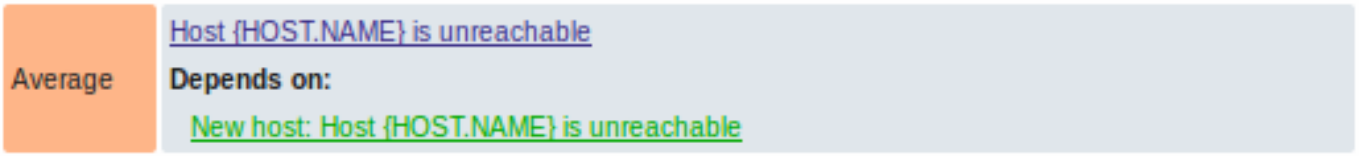
- Trigger dependency may be added from any host trigger to any other host trigger, as long as it wouldn't result in a circular dependency.
- Trigger dependency may be added from a template to a template. If a trigger from template A depends on a trigger from template B, template A may only be linked 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.

Configuration

To define a dependency, open the Dependencies tab in a trigger configuration form. Click on *Add* in the 'Dependencies' block and select one or more triggers that our trigger will depend on.



Click *Update*. Now the trigger has an indication of its dependency in the list.



Example of several dependencies

For example, a Host is behind a Router2 and the Router2 is behind a Router1.

```
Zabbix - Router1 - Router2 - Host
```

If Router1 is down, then obviously Host and Router2 are also unreachable yet we don't want to receive three notifications about Host, Router1 and Router2 all being down.

So in this case we define two dependencies:

```
'Host is down' trigger depends on 'Router2 is down' trigger
'Router2 is down' trigger depends on 'Router1 is down' trigger
```

Before changing the status of the 'Host is down' trigger, Zabbix will check for corresponding trigger dependencies. If found, and one of those triggers is in 'Problem' state, then the trigger status will not be changed and thus actions will not be executed and notifications will not be sent.

Zabbix performs this check recursively. If Router1 or Router2 is unreachable, the Host trigger won't be updated.

4 Trigger severity

Trigger severity defines how important a trigger is. Zabbix supports the following trigger severities:

SEVERITY	DEFINITION	COLOUR
Not classified	Unknown severity.	Grey
Information	For information purposes.	Light green
Warning	Be warned.	Yellow
Average	Average problem.	Orange
High	Something important has happened.	Red
Disaster	Disaster. Financial losses, etc.	Bright red

The severities are used for:

- visual representation of triggers. Different colours for different severities.
- audio in global alarms. Different audio for different severities.
- user media. Different media (notification channel) for different severities. For example, SMS - high severity, email - other.
- limiting actions by conditions against trigger severities

It is possible to customise trigger severity names and colours.

5 Customising trigger severities

Trigger severity names and colours for severity related GUI elements can be configured in *Administration → General → Trigger severities*. Colours are shared among all GUI themes.

Translating customised 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 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' you can simply enter '1d'. Suffixes function as multipliers.

Trigger expressions

Time and memory size suffixes are supported in trigger expression constants and function parameters.

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 are also supported in parameters of the **zabbix[queue,<from>,<to>]** internal item and the last parameter of aggregate checks.

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
```

7 Mass update

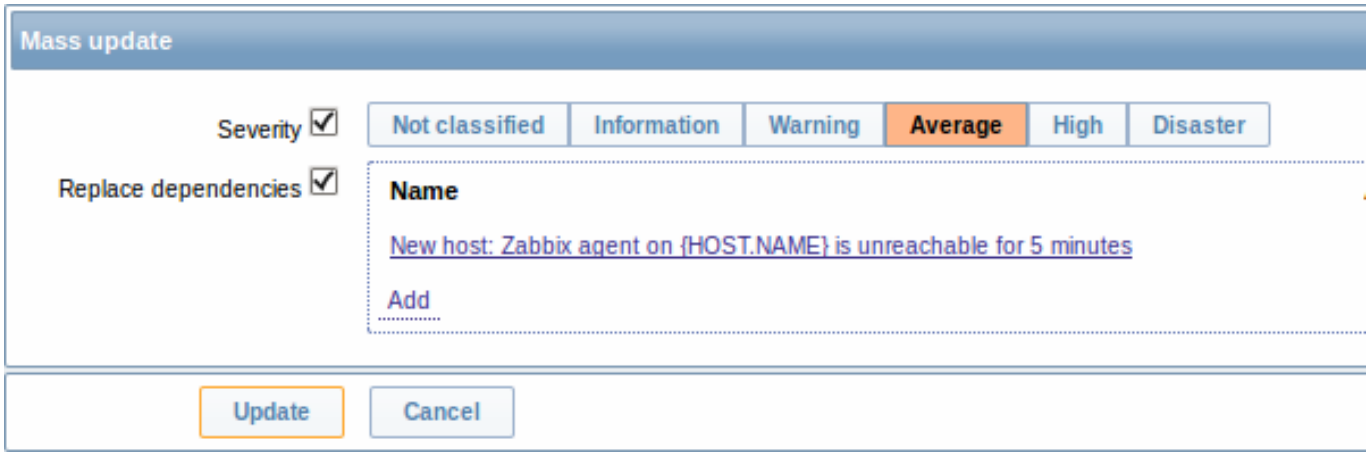
Overview

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 to update in the list
- Select *Mass update* below the list and click on *Go*
- Mark the checkboxes of the attributes to update
- Specify new values for the attributes and click on *Update*



Replace dependencies will replace existing trigger dependencies (if any) with the ones specified in mass update.

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
- auto registration 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 → Events*. There you can click on the event date and time to view details of an event.

More information is available on each event source.

1 Event sources

1 Trigger events

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 did it happen and what the new state is.

2 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.

3 Active agent auto-discovery events

Active agent auto-registration creates events in Zabbix.

If configured, active agent auto-registration can happen when a previously unknown active agent asks for checks. 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 auto-registration page.

4 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 starting with 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.

5 Visualisation

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 customised graphs
- access to a comparison of several items quickly in ad-hoc graphs

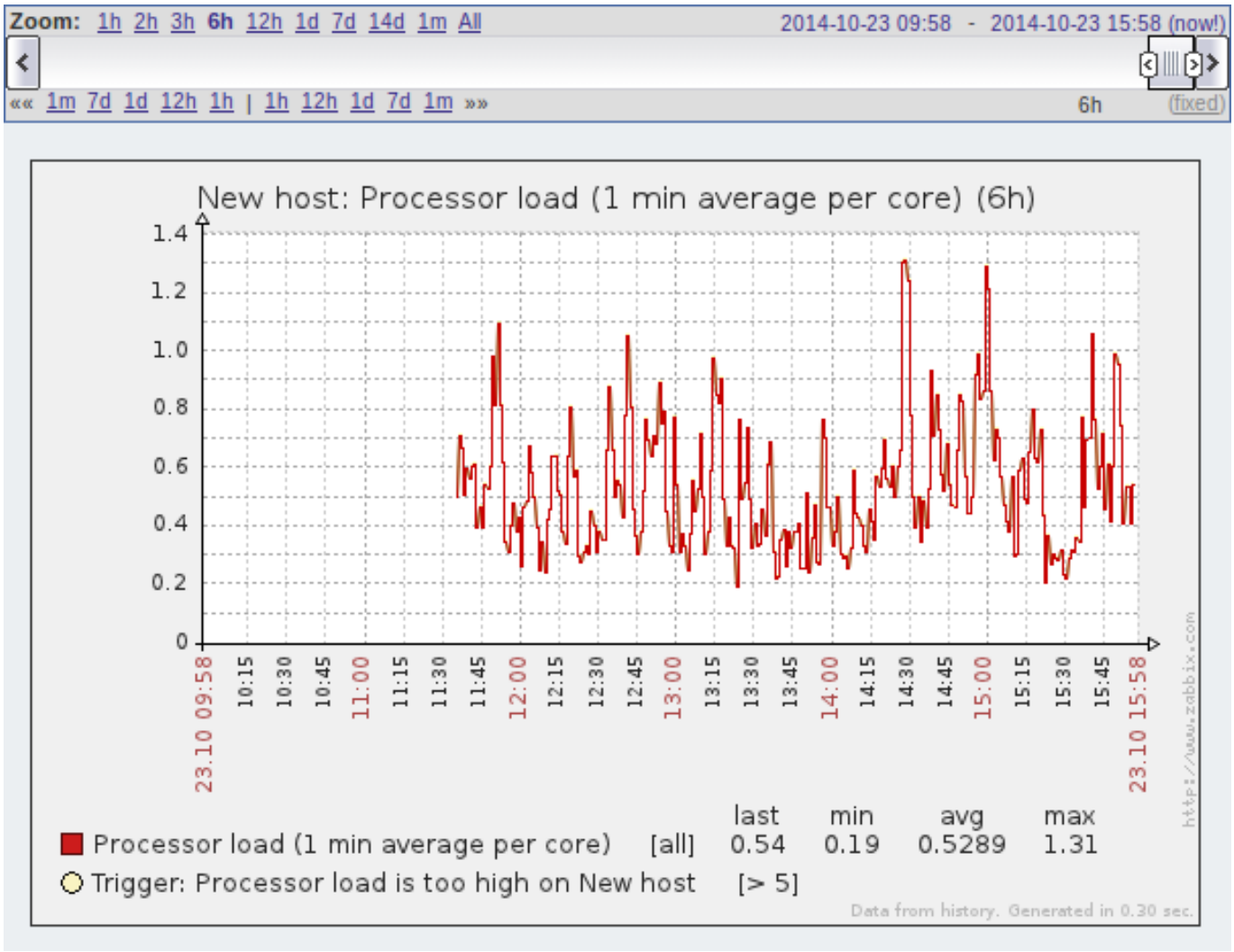
1 Simple graphs

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.



Time period selector

Take note of the time period selector above the graph. It allows you to select the desired time period easily.

The slider within the selector can be dragged back and forth, as well as resized, effectively changing the time period displayed. Links on the left hand side allow to choose some often-used predefined periods (above the slider area) and move them back and forth in time (below the slider area). The dates on the right hand side actually work as links, popping up a calendar and allowing to set a specific start/end time.

The **fixed/dynamic** link in the lower right hand corner has the following effects:

- controls whether the time period is kept constant when you change the start/end time in the calendar popup.
- when *fixed*, time moving controls (« 6m 1m 7d 1d 12h 1h | 1h 12h 1d 7d 1m 6m ») will move the slider, while not changing its size, whereas when *dynamic*, the control used will enlarge the slider in the respective direction.
- when *fixed*, pressing the larger **<** and **>** buttons will move the slider, while not changing its size, whereas when *dynamic*, **<** and **>** will enlarge the slider in the respective direction. The slider will move by the amount of its size, so, for example, if it is one month, it will move by a month; whereas the slider will enlarge by 1 day.

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.

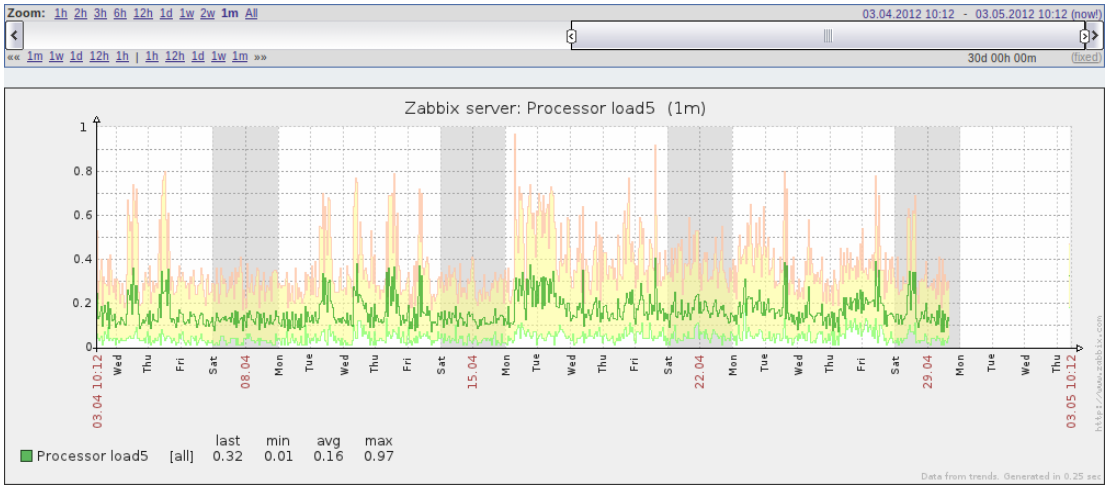
Note:
Simple graphs are provided for all numeric items. For textual items, a link to History is available in *Monitoring → Latest data*.

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 grey (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.

Generating from history/trends

Graphs can be drawn based on either item history or trends. A grey caption at the bottom right of a graph indicates 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).

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.

2 Custom graphs

Overview

Custom graphs, as the name suggests, offer customisation 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.

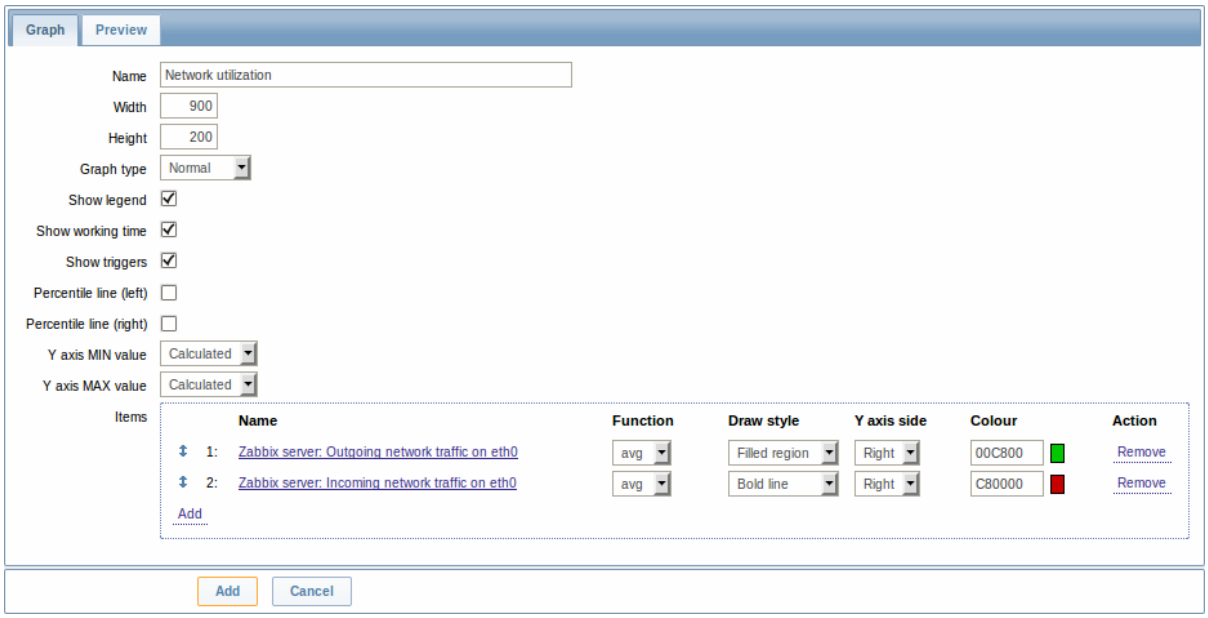
Custom graphs are configured manually.

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



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 red lines. 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.

Parameter	Description
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.

Configuring graph items

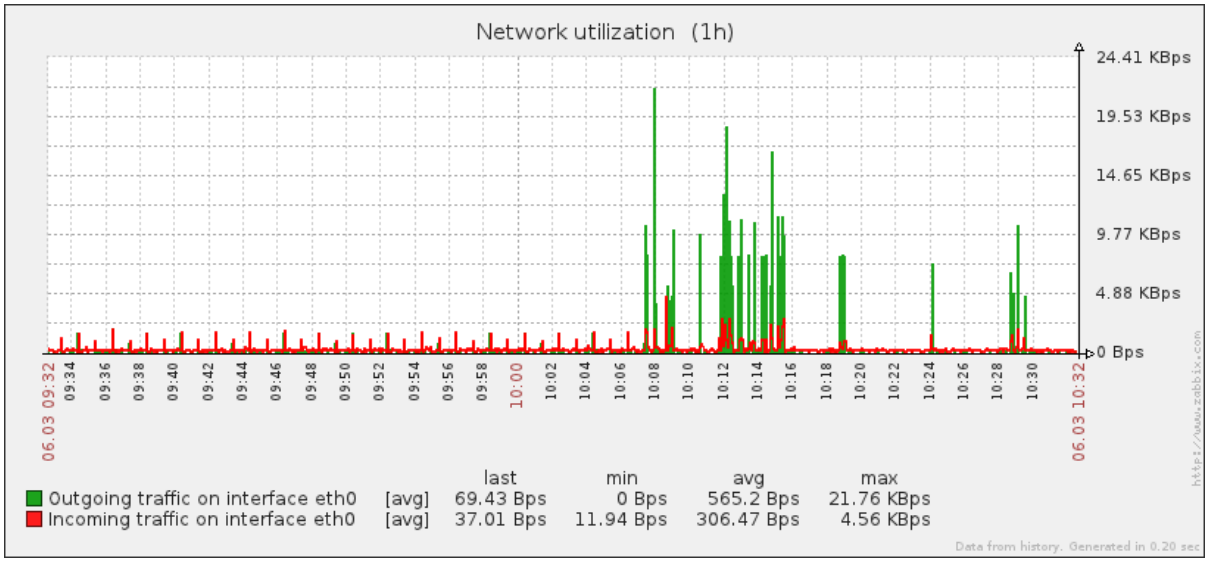
To add items, data of which are to be displayed in the graph, click on *Add* in the *Items* block, select items and then set attributes for the way item data will be displayed.

Item display attributes:

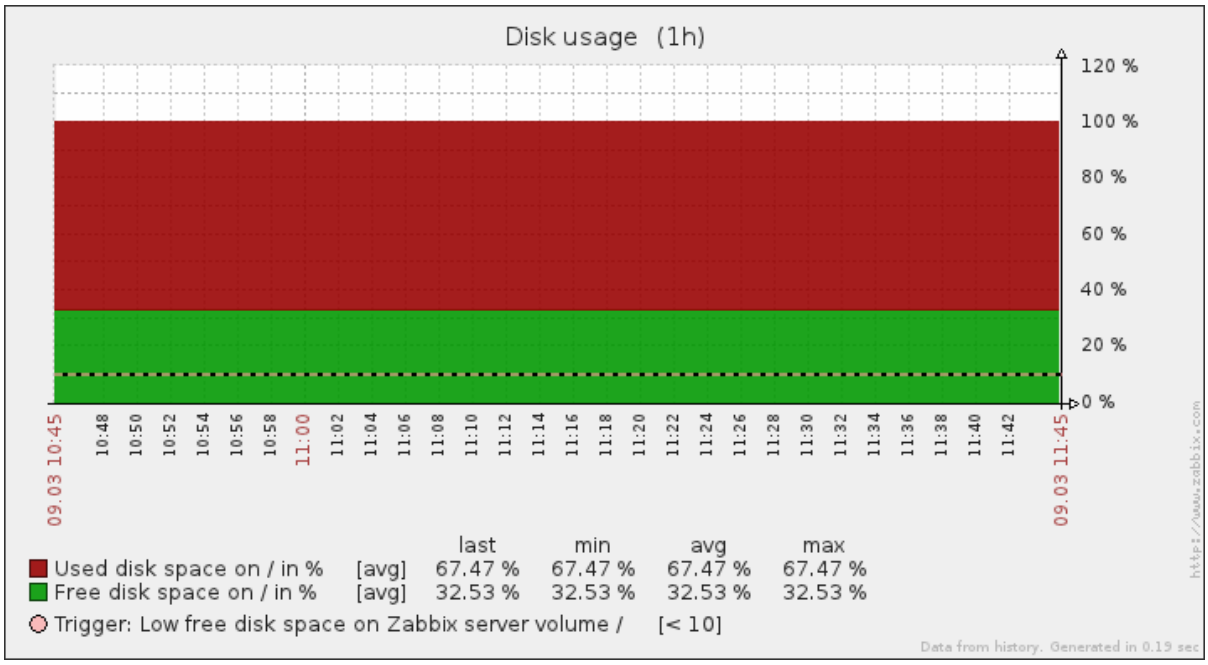
Parameter	Description
Sort order (0→100)	Draw order. 0 will be processed first. Can be used to draw lines or regions behind (or in front of) another. You can drag and drop items by the arrow in the beginning of line to set the sort order or which item is displayed in front of the other.
Name	Name of item, data of which will be displayed.
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 colouring of the ”graph sum” item will only be visible to the extent that it is not taken up by ”proportional” items.
Function	What values will be displayed when more than one value exists for an item: **all** - all (minimum, average and maximum) **min** - minimum only **avg** - average only **max** - maximum only
Draw style	Draw style (only available for normal graphs; for stacked graphs filled region is always used): **Line** - draw lines **Filled region** - draw filled region **Bold line** - draw bold lines **Dot** - draw dots **Dashed line** - draw dashed line
Y axis side	Which Y axis side the element is assigned to.
Colour	RGB colour in HEX notation.

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:
3 triggers is the hard-coded limit for the number of triggers displayed in the legend.
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 customisation options, none of the two allow to quickly create a comparison graph for multiple items with little effort and no maintenance.

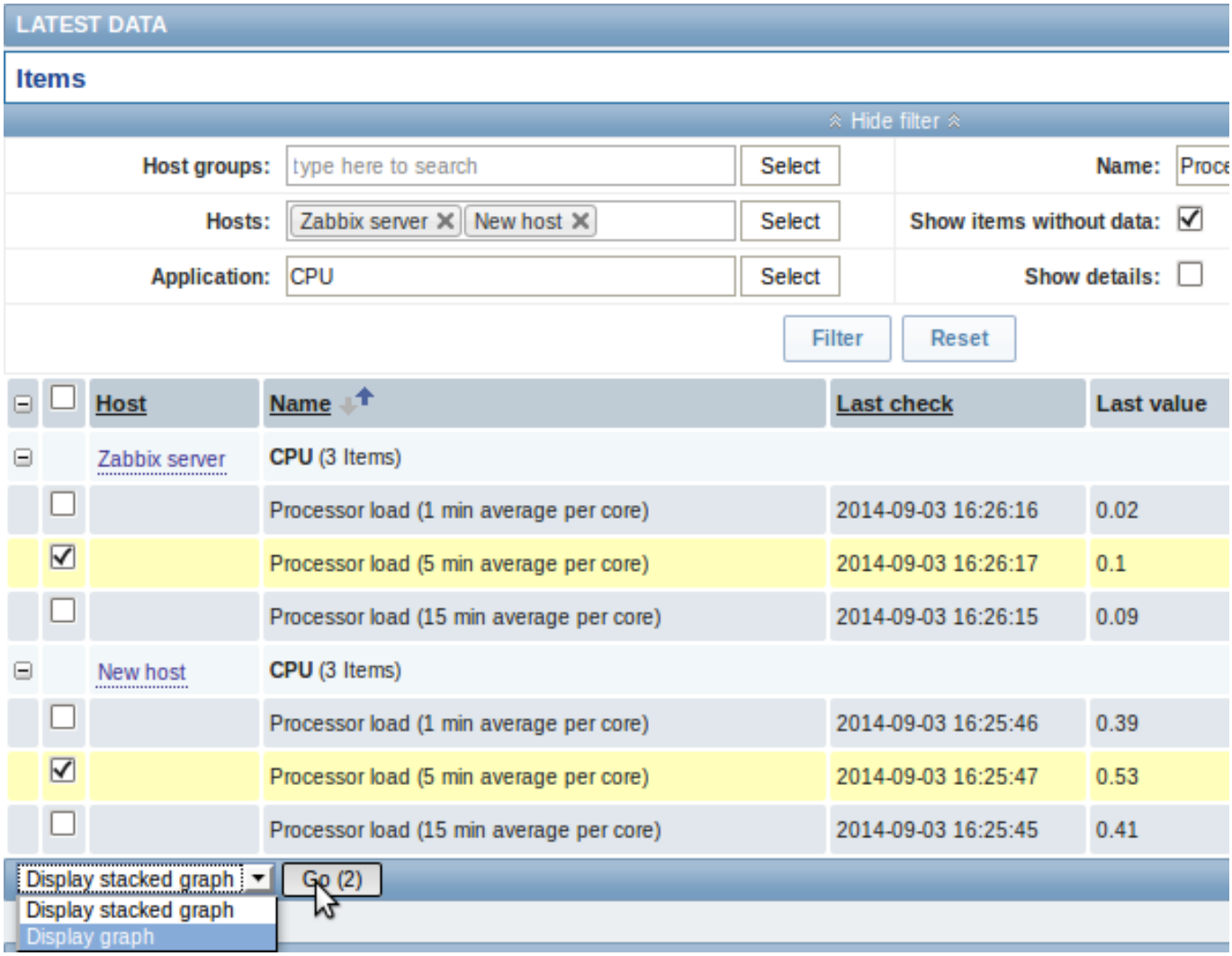
To address this issue, since Zabbix 2.4 it is possible to create ad-hoc graphs for several items in a very quick way.

Configuration

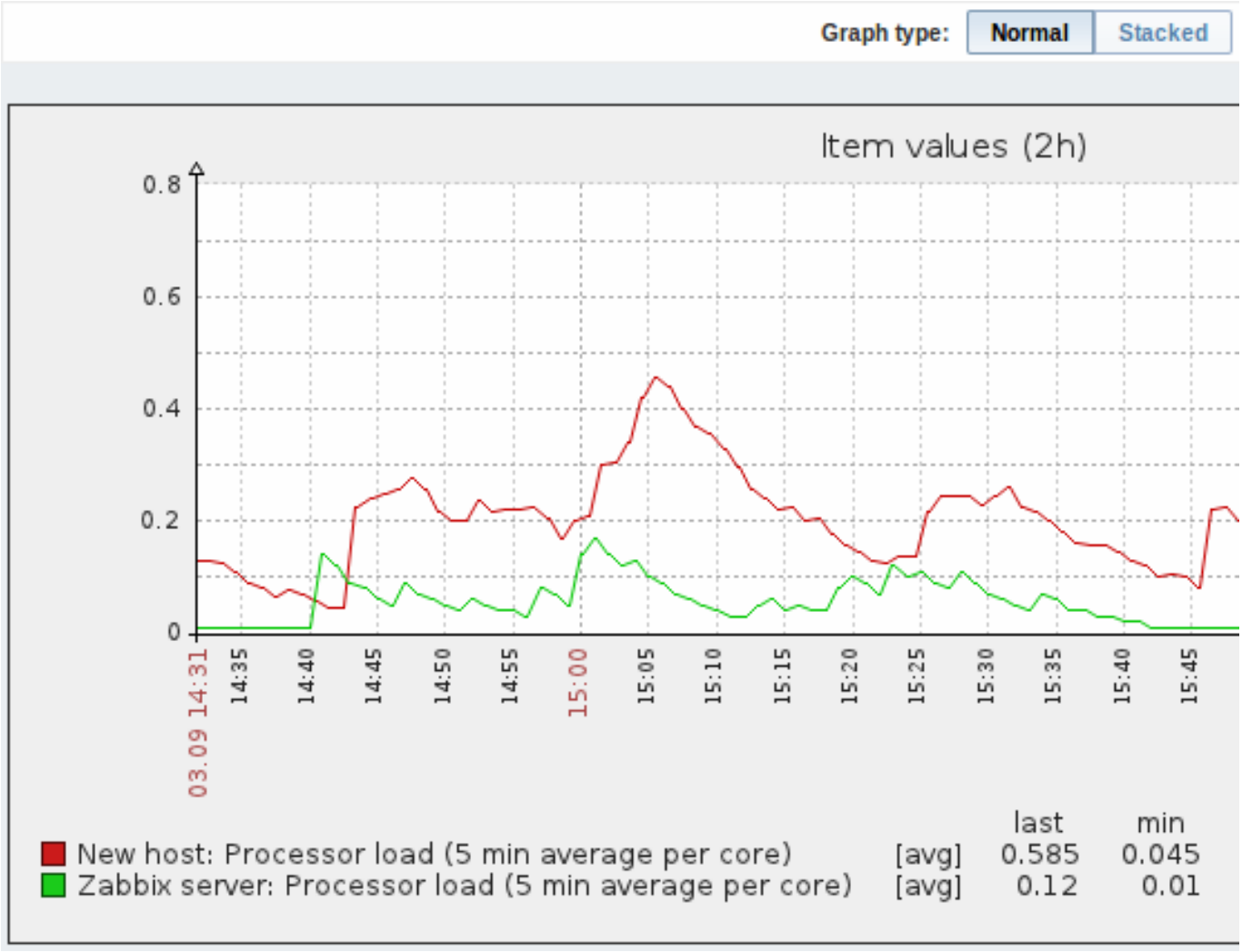
To create an ad-hoc graph, do the following:

- Go to *Monitoring* → *Latest data*
- Use filter to display items that you want
- Mark checkboxes of the items you want to graph

- Select *Display stacked graph* or *Display graph* in the dropdown below and click on *Go*

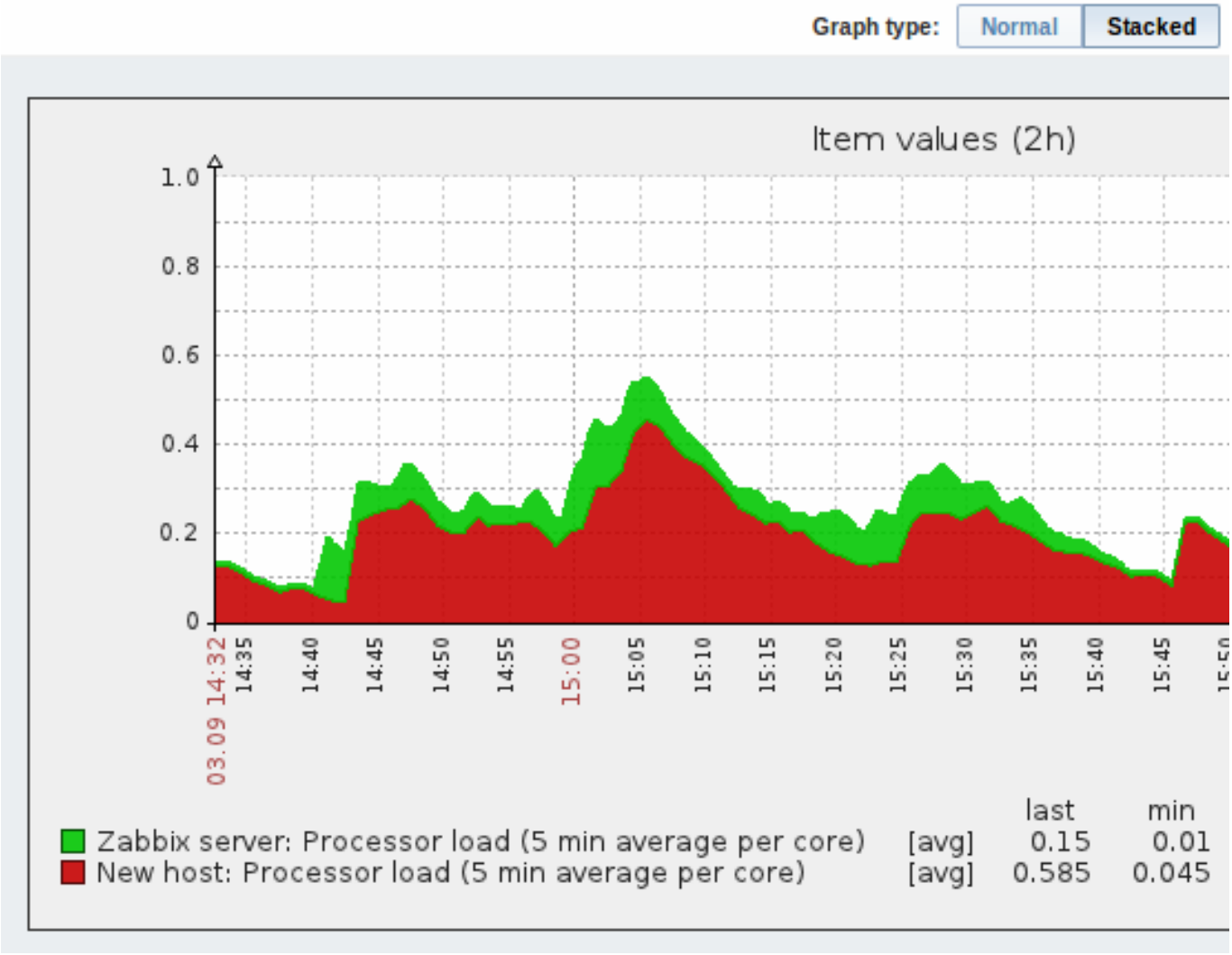


Your graph is created instantly:



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).



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.

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.

Maps that are ready can be viewed in *Monitoring* → *Maps*. In the monitoring view you can click on the icons and take advantage of the links to some scripts and URLs.

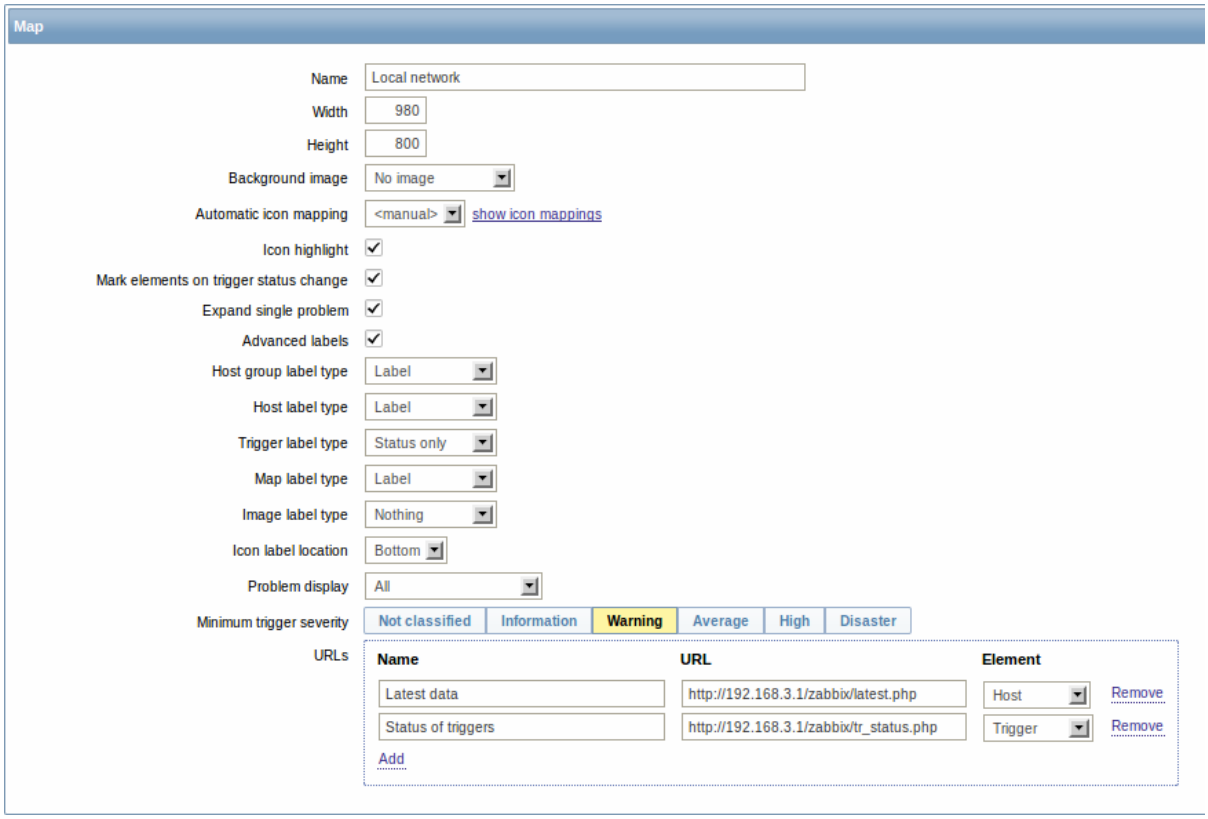
You can add custom URLs to be accessible by clicking on the icons. Thus you may link a host icon to host properties or a map icon to another map.

Creating a map

To create a map, do the following:

- Go to *Configuration* → *Maps*

- Click on *Create map*
- Edit general map attributes



General map attributes:

Parameter	Description
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, icons will receive highlighting. Elements with an active trigger will receive a round background, in the same colour 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.
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.
Expand single problem	If a map element (host, host group or another map) has one single problem, this option controls whether the problem (trigger) name is displayed, or problem count. If marked, problem name is used.
Advanced labels	If you check this box you will be able to define separate label types for separate element types.

Parameter	Description
Icon label type	Label type used for icons: **Label** - icon 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
Icon label location	Label location in relation to the icon: **Bottom** - beneath the icon **Left** - to the left **Right** - to the right **Top** - above the icon
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.
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 monitoring section. Macros that can be used in map URLs: {MAP.ID}, {HOSTGROUP.ID}, {HOST.ID}, {TRIGGER.ID}

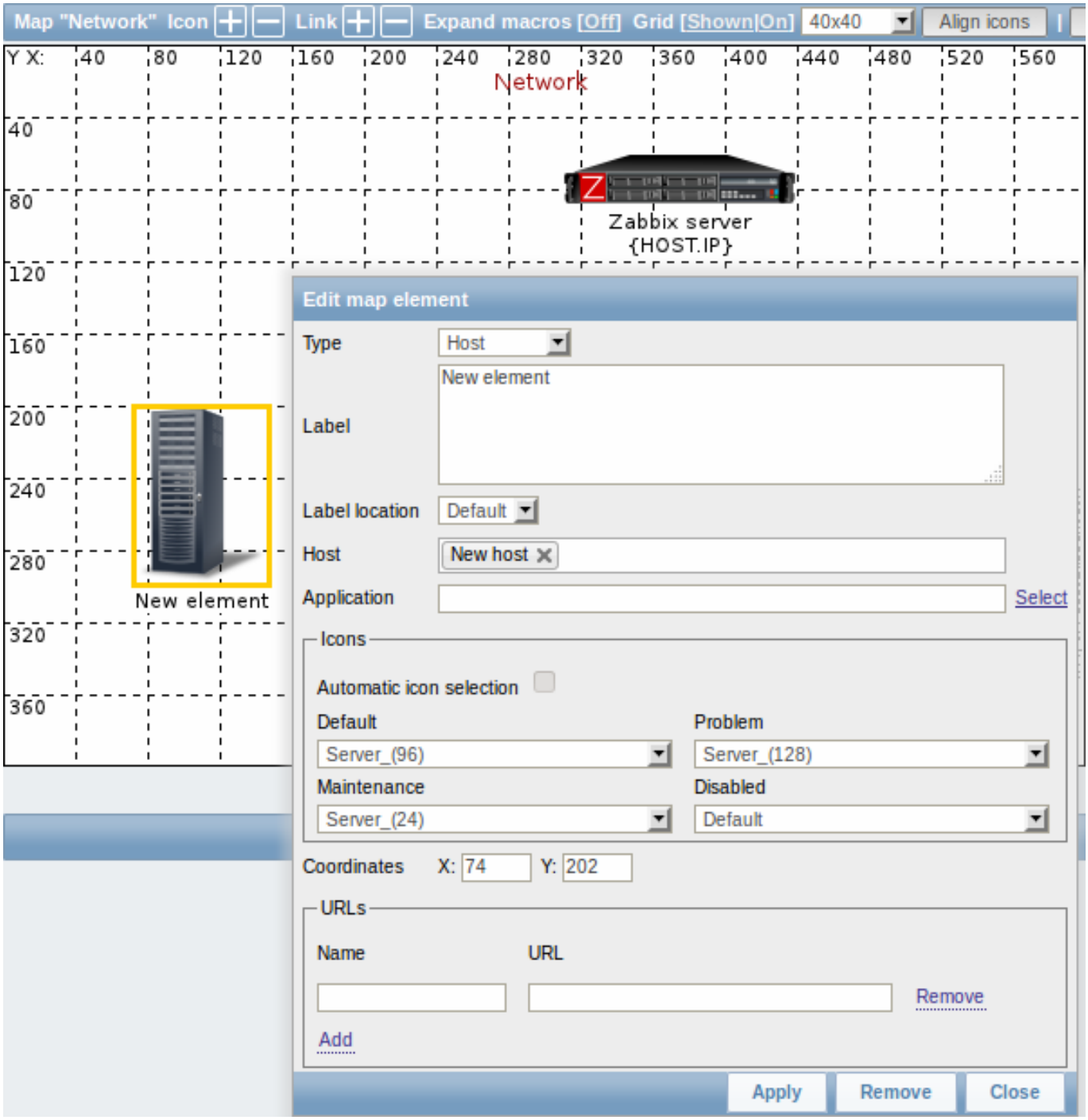
When you click on *Add* to save this, you have created an empty map with a name, dimensions and certain preferences. Now you need to add some elements. For that, click on the map name in the list to open the editable area.

Adding elements

To add an element, click on the ”+” next to Icon. 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 icons* 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 a single trigger **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 multi-line strings can be used in 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'.
Trigger	Select the trigger, if the element type is 'Trigger'.
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**.
Icon (default)	Icon to be used.
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 monitoring section. 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 that can be used in map URLs: {MAP.ID}, {HOSTGROUP.ID}, {HOST.ID}, {TRIGGER.ID}

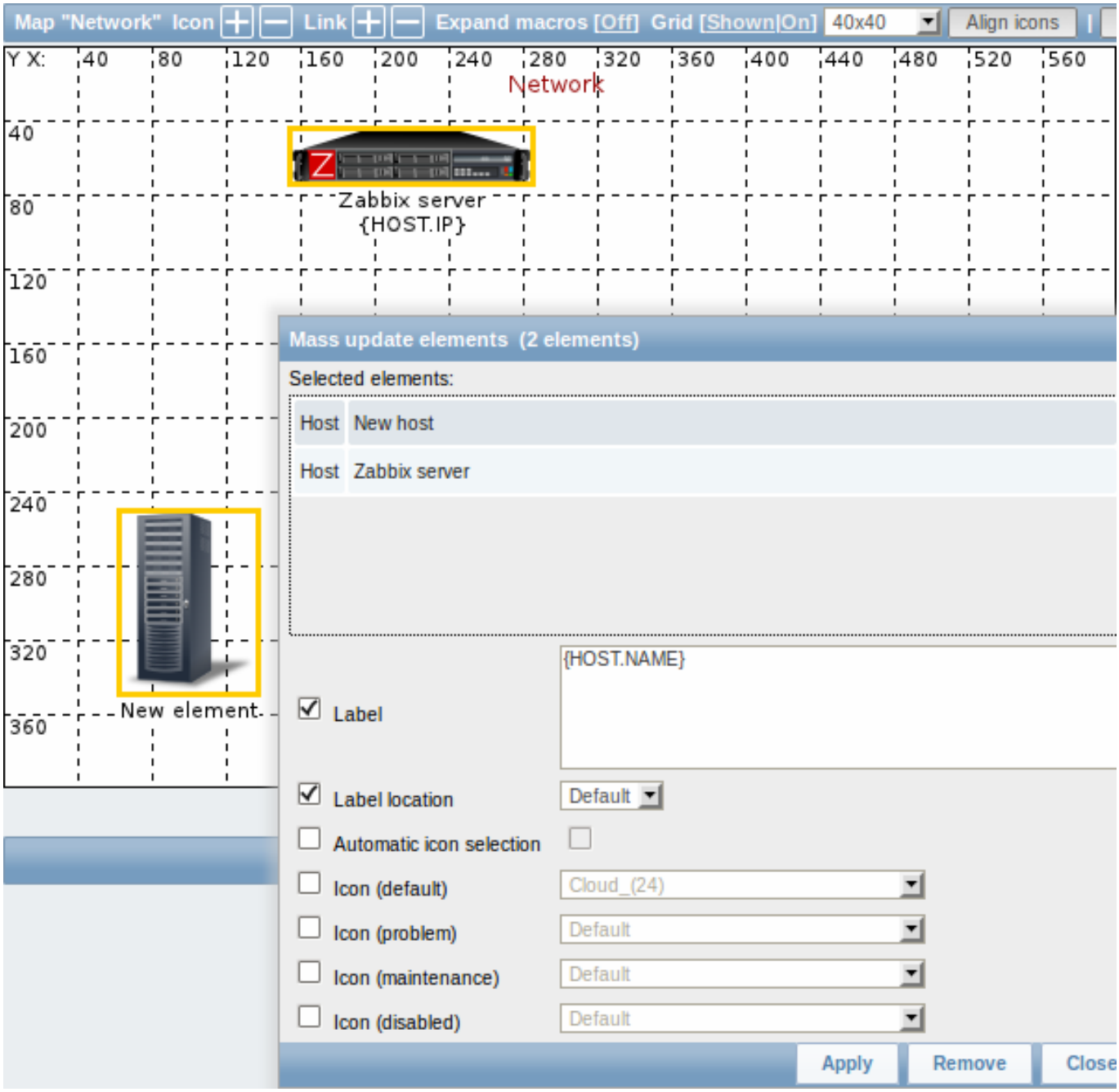
Attention:
Added elements are not automatically saved. If you navigate away from the page, all changes may be lost.
Therefore it is a good idea to click on the **Update** button in the top right corner. Once clicked, the changes are saved regardless of what you choose in the following popup.
Selected 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 (option available since Zabbix 2.0).

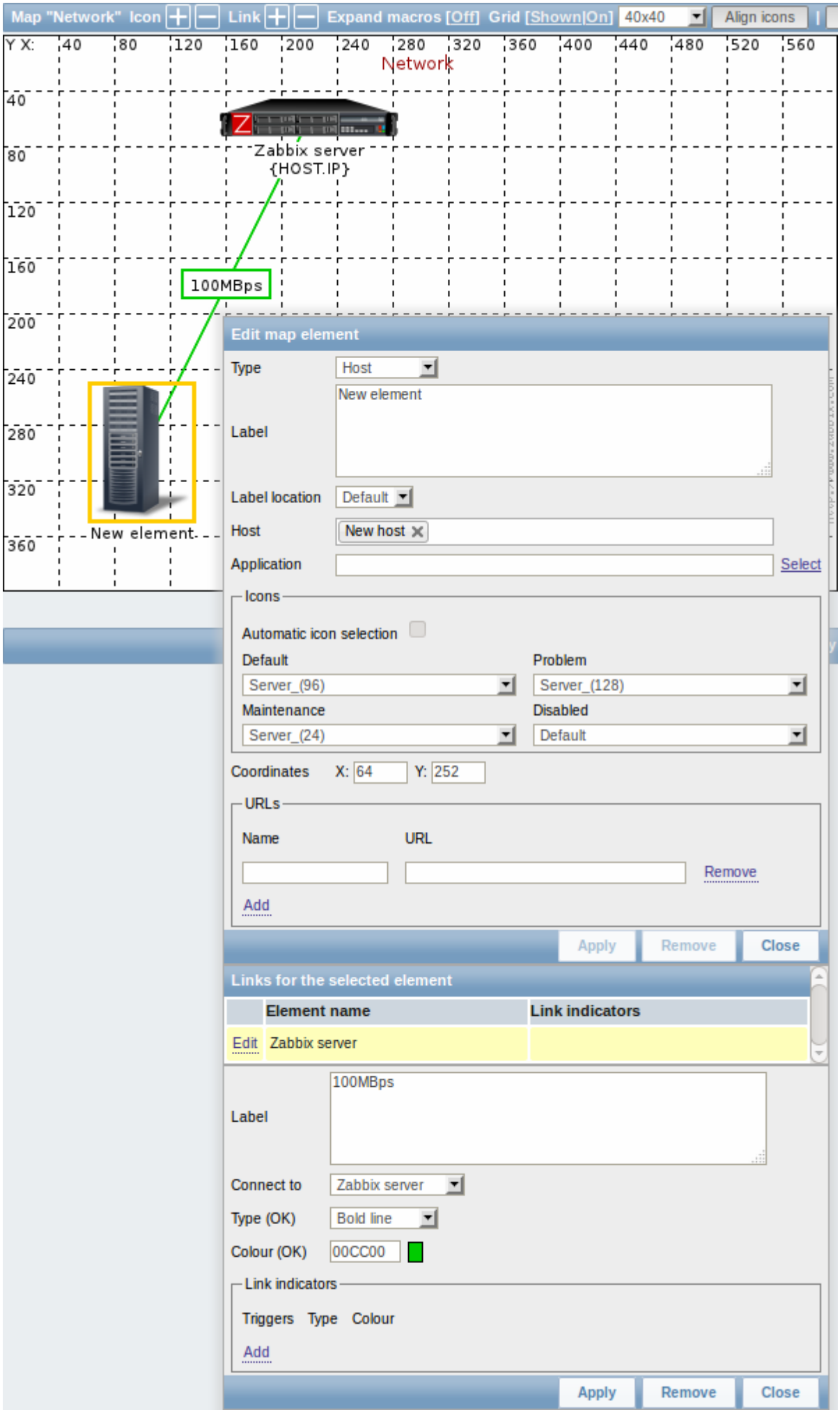
Once you select more than one element, the element property form shifts to the mass-update mode so you can change attributes of selected elements in one go. To do so, mark the attribute using the checkbox and enter a new value for it. You may use macros here (such as, say, {HOST.NAME} for the element 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 the "+" next to Link.

With a link created, the single element form now contains an additional *Links for the selected element* section. Click on *Edit* before the link to edit its attributes.


Map "Network" Icon Link Expand macros [Off] Grid [Shown|On] 40x40 Align icons
Network
Zabbix server
{HOST.IP}
100MBps
New element
Edit map element
Type Host
Label New element
Label location Default
Host New host
Application Select
Icons
Automatic icon selection
Default Server_(96)
Problem Server_(128)
Maintenance Server_(24)
Disabled Default
Coordinates X: 64 Y: 252
URLs
Name URL
Remove
Add
Apply Remove Close
Links for the selected element
Element name Link indicators
Edit Zabbix server
Label 100MBps
Connect to Zabbix server
Type (OK) Bold line
Colour (OK) 00CC00
Link indicators
Triggers Type Colour
Add
Apply Remove Close

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
Colour (OK)	Default link colour.
Link indicators	List of triggers linked to the link. In case a trigger has status PROBLEM, its style is applied to the link.

2 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.

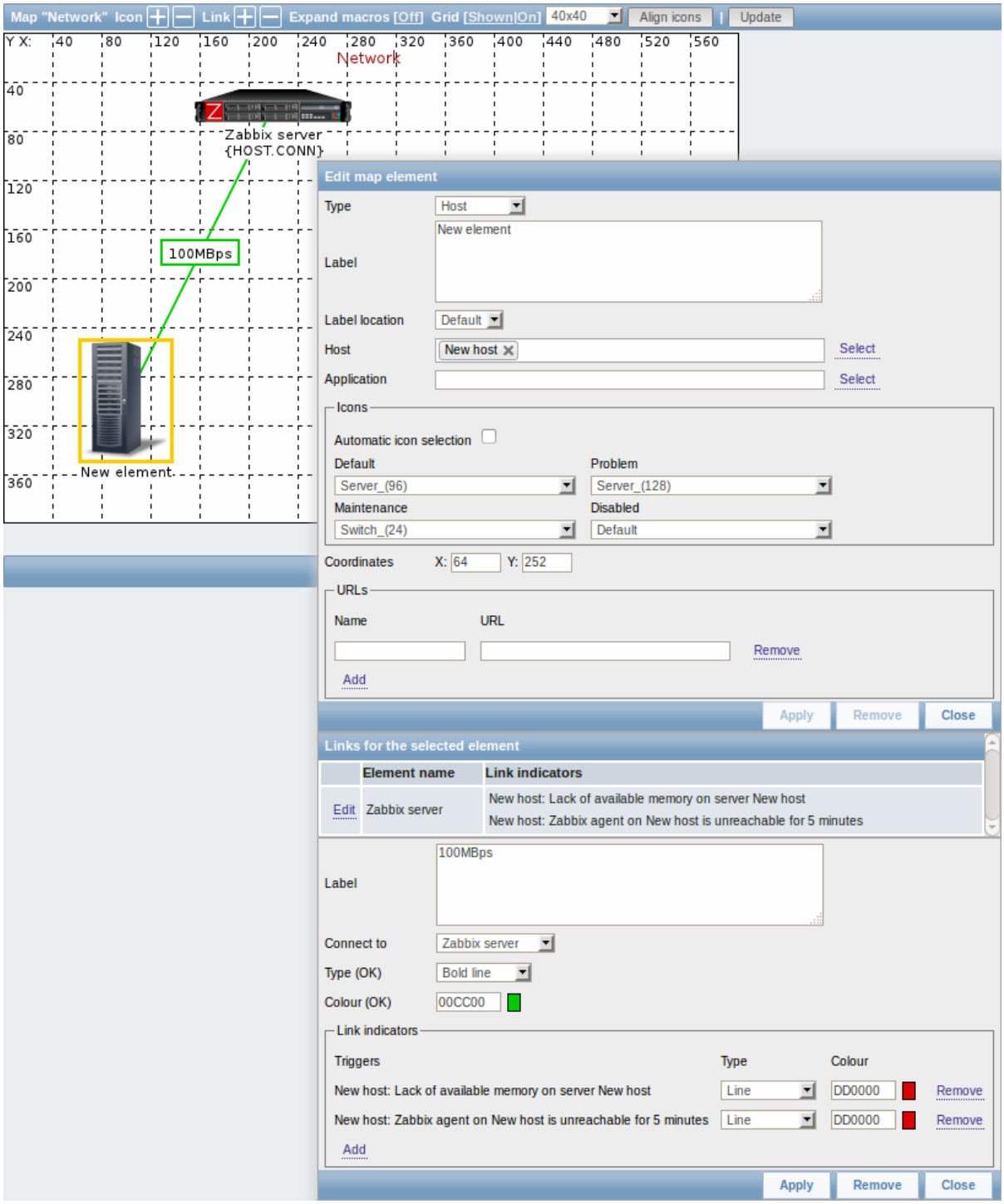
When you configure a link, you set the default link type and color. When you assign triggers to a link, you can assign different link types and colors with these triggers.

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 for the selected element* section before the appropriate link
- click on *Add* in the *Link indicators* block and select one or more triggers

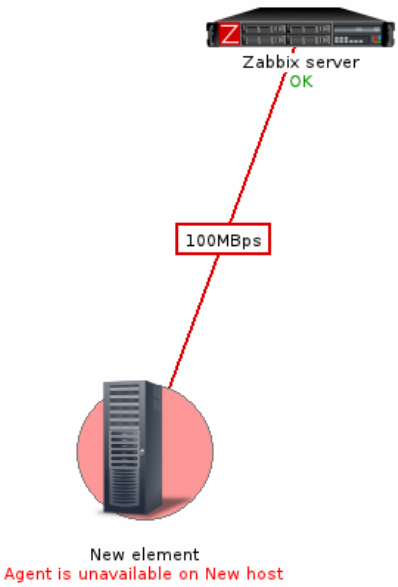


Added triggers can be seen in the *Link indicators* list.

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 one 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.

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
- other screens
- plain text information
- server information (overview)
- host information (overview)
- trigger information (overview)
- host/hostgroup issues (status of triggers)
- system status
- data overview
- clock
- history of events
- history of recent actions
- URL (data taken from another location)

Screens that are ready can be viewed in *Monitoring → Screens*. They can also be added to the favourites section of the Dashboard.

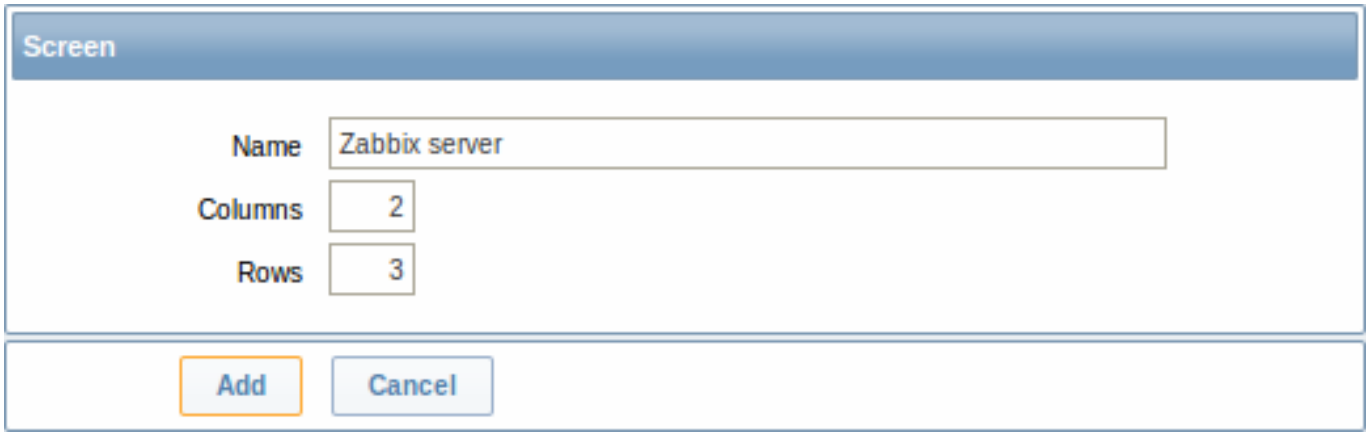
To configure a screen you must first create it by defining its general properties and then add individual elements in the cells.

Creating a screen

To create a screen, do the following:

- Go to *Configuration → Screens*
- Click on *Create Screen*

- Edit general screen attributes



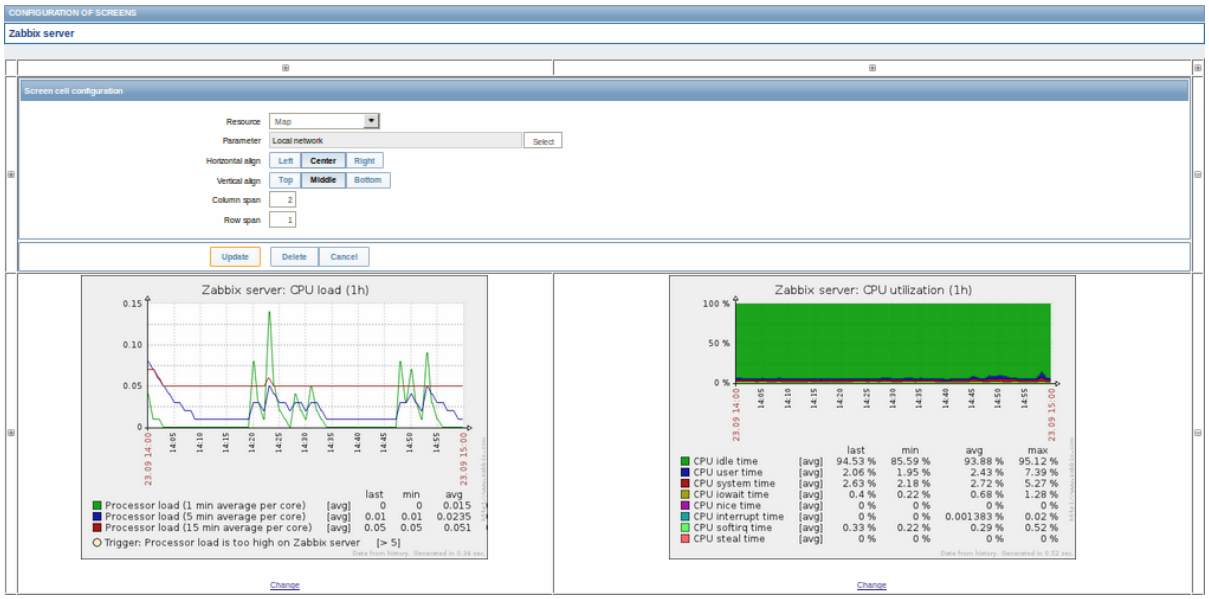
Give your screen a unique name and set the number of columns (vertical cells) and rows (horizontal cells). Click *Add*.

Now you can click on the screen name in the list to be able to add elements.

Adding elements

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.



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 (available since Zabbix 2.4) **History of events** - latest events **Host group issues** - status of triggers filtered by the hostgroup (includes triggers without events, since Zabbix 2.2) **Host issues** - status of triggers filtered by the host (includes triggers without events, since Zabbix 2.2) **Hosts info** - high level host related information **Map** - single map **Plain text** - plain text data **Screen** - screen (one screen may contain other screens inside) **Server info** - server high-level information **Simple graph** - single simple graph **Simple graph prototype** - simple graph based on item generated by low-level discovery (available since Zabbix 2.4) **System status** - displays system status (similar to the Dashboard) **Triggers info** - high level trigger related information **Triggers overview** - status of triggers for a host group **URL** - include content from an external 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 extra dropdowns 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

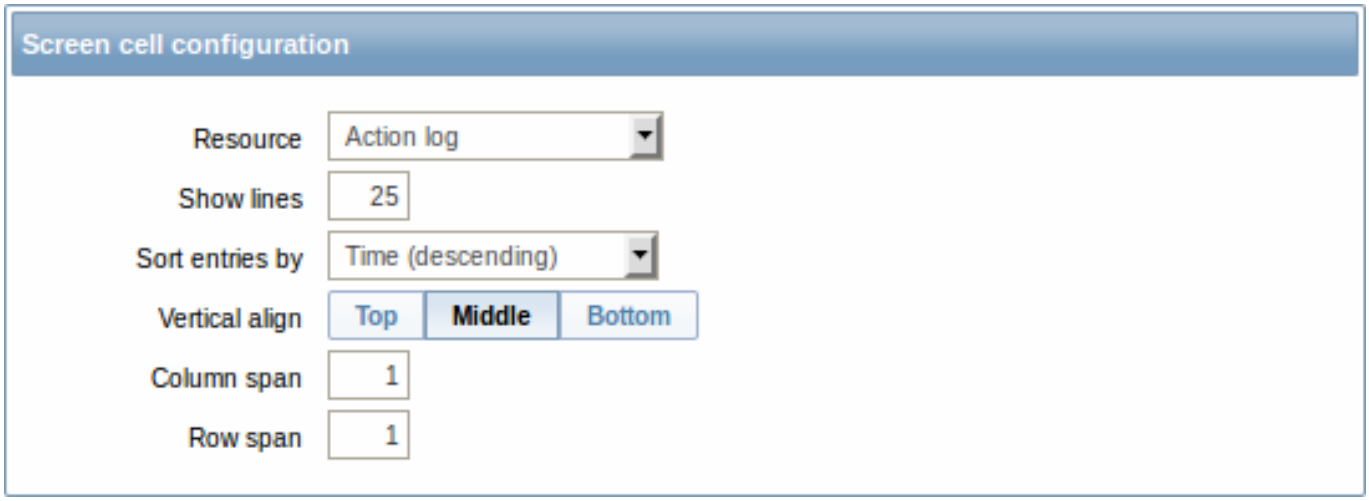
Overview

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 *Administration → Audit*.

To configure, select *Action log* as resource:



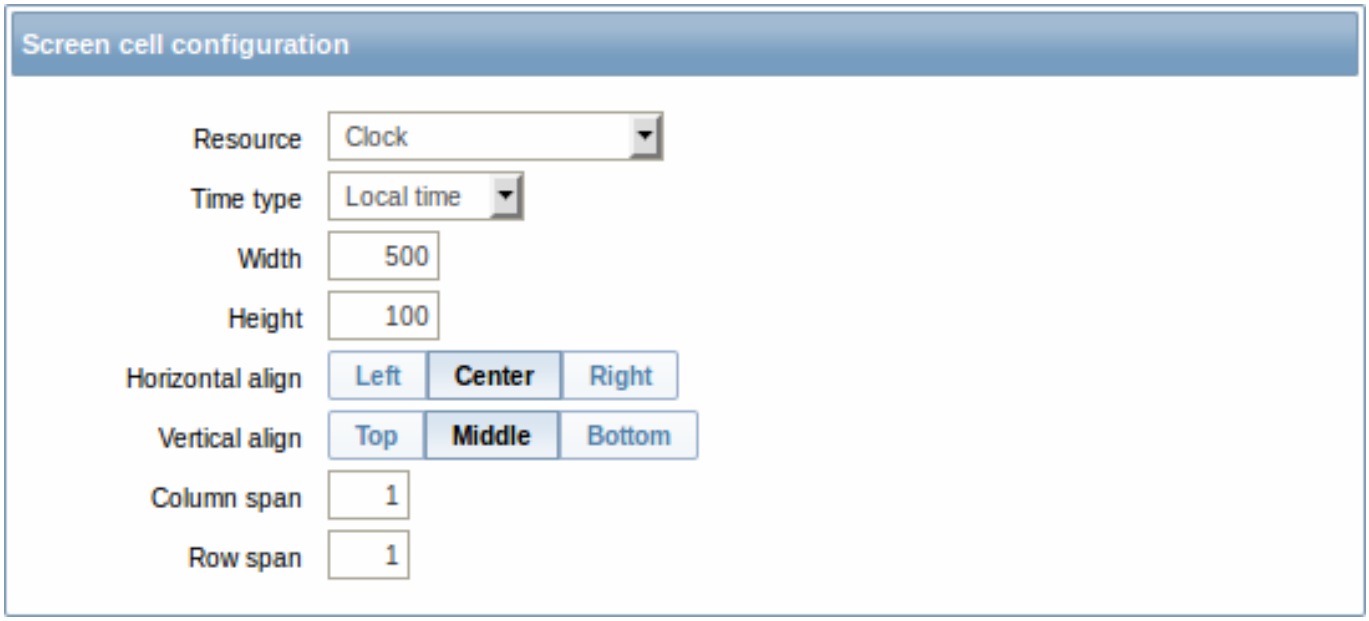
You may set the following specific options:

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:



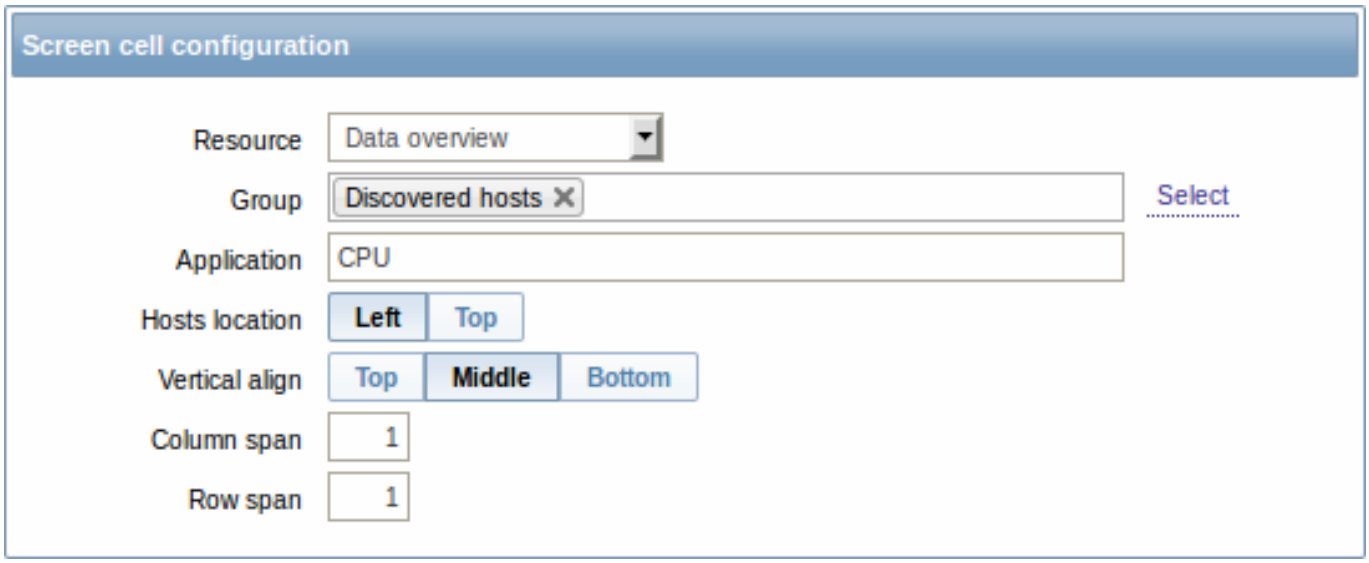
You may set the following specific options:

Time type	Select local, server or specified host time.
Parameter	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:



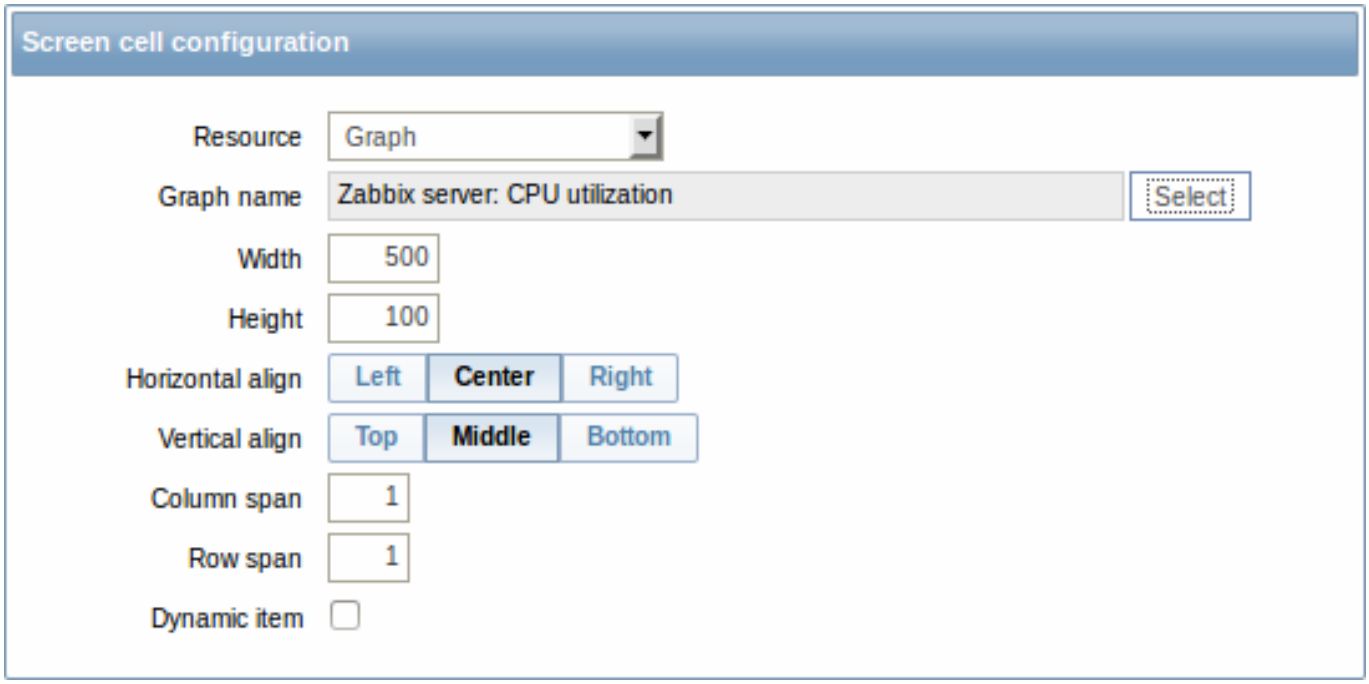
You may set the following specific options:

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:



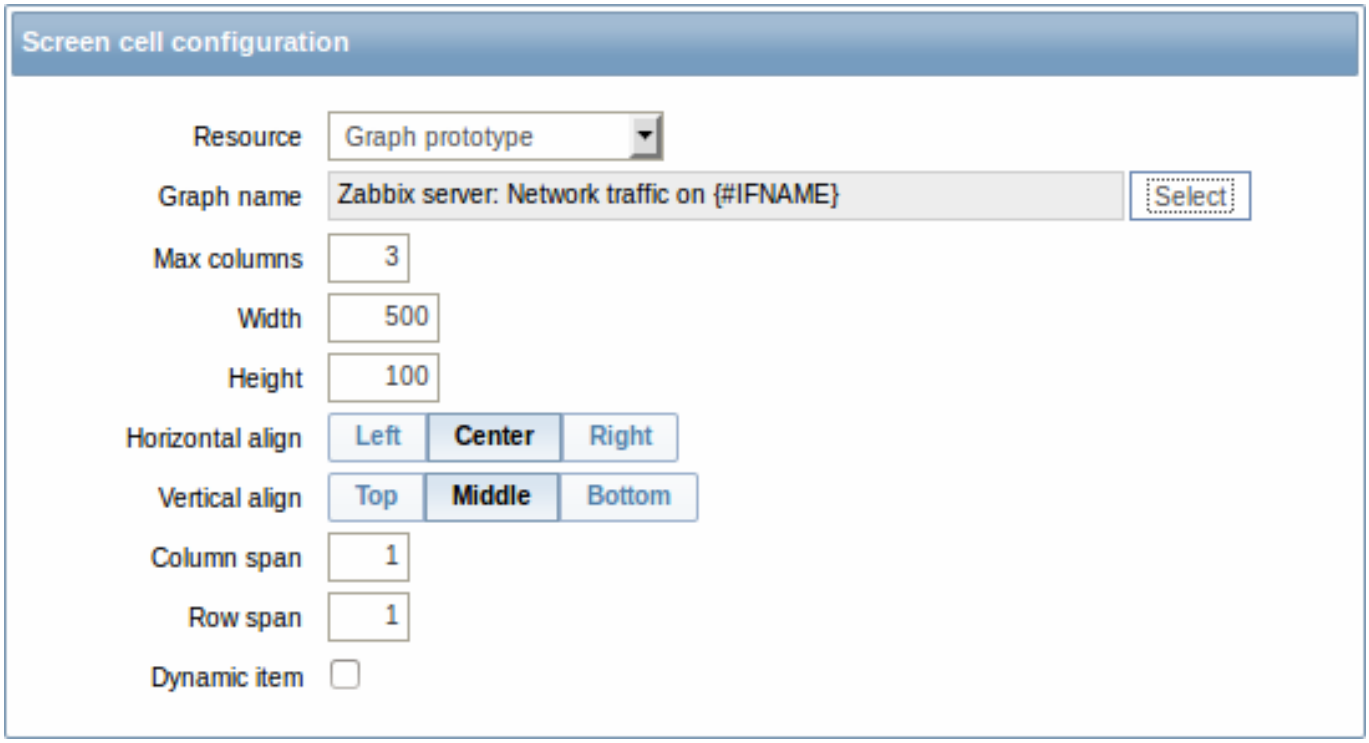
You may set the following specific options:

Graph name	Select the graph to display.
Width	Select graph width.
Height	Select graph height.
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:



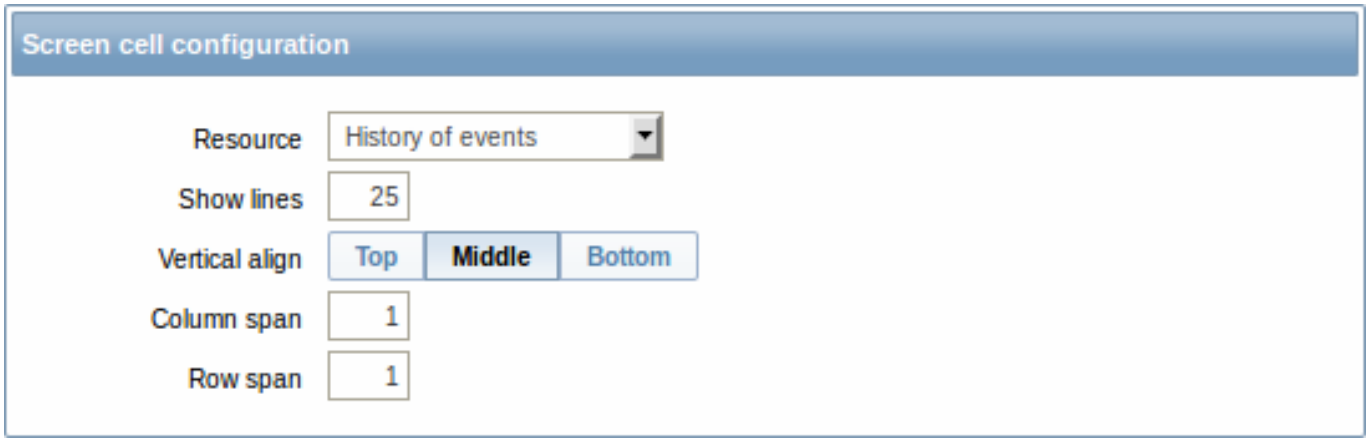
You may set the following specific options:

Graph name	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.
Height	Select graph height.
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:



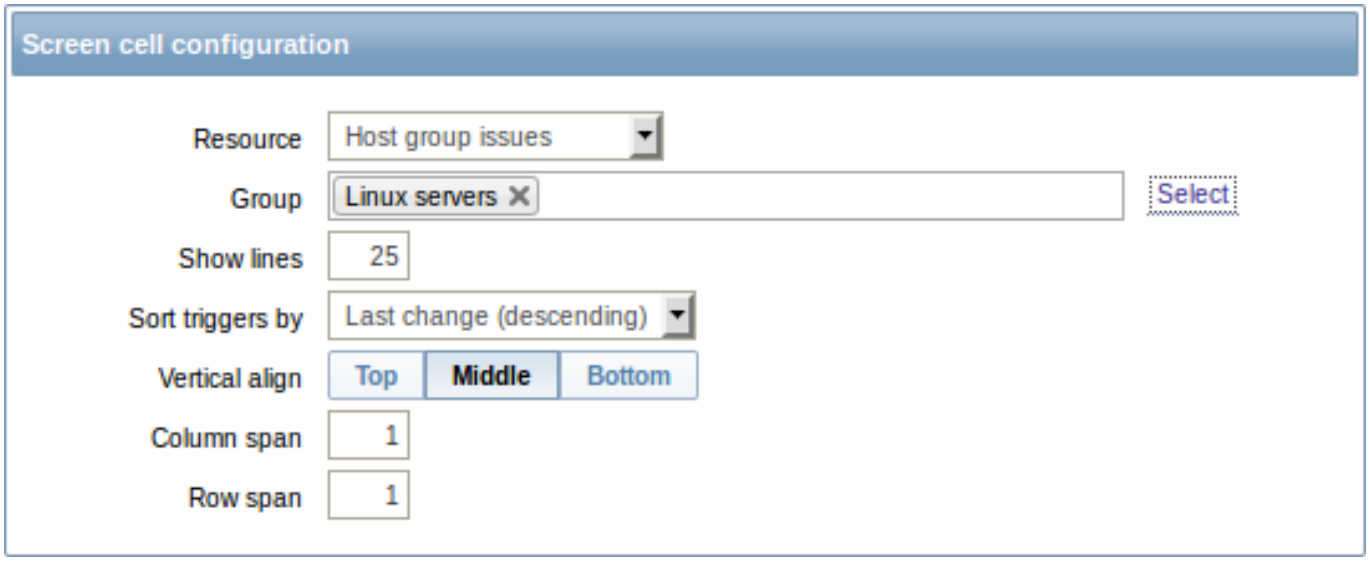
You may set the following specific option:

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 status of triggers filtered by the host group. It will be displayed similarly as in *Last 20 issues* from the Dashboard.

To configure, select *Host group issues* as resource:



You may set the following specific options:

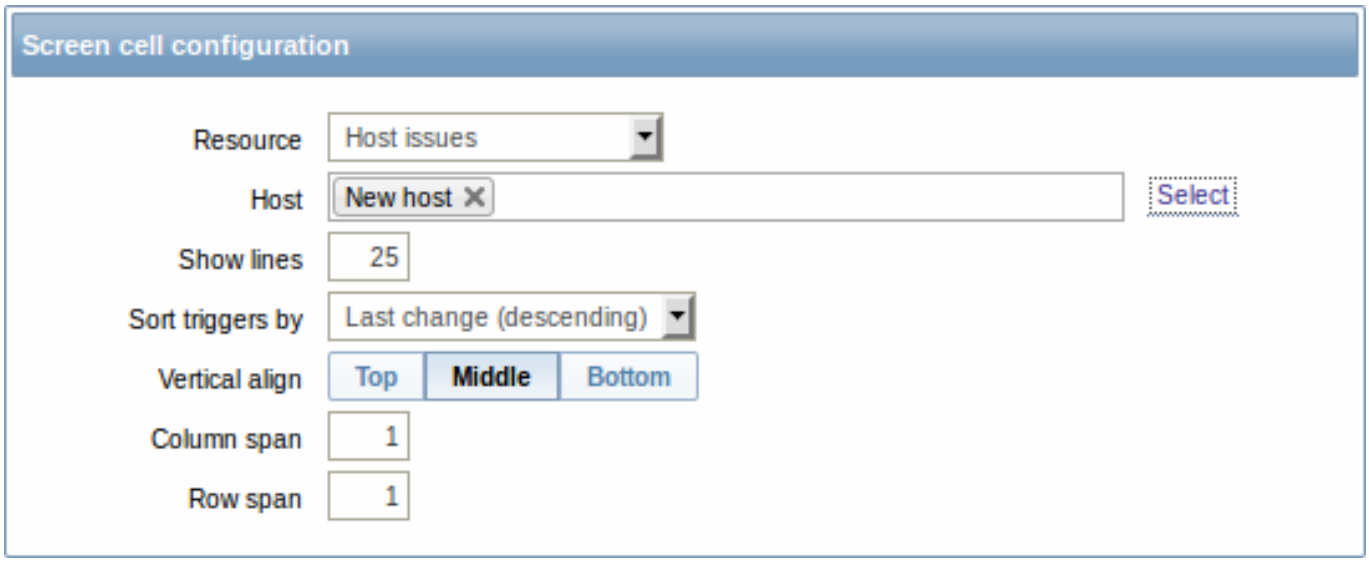
Group	Select host group.
Show lines	Set how many trigger status lines will be displayed in the screen cell.

Sort triggers by	Select from the dropdown to sort triggers by last change, severity (both descending) or host (ascending).

8 Host issues

In the host issue element you can display status of triggers filtered by the host. It will be displayed similarly as in *Last 20 issues* from the Dashboard.

To configure, select *Host issues* as resource:



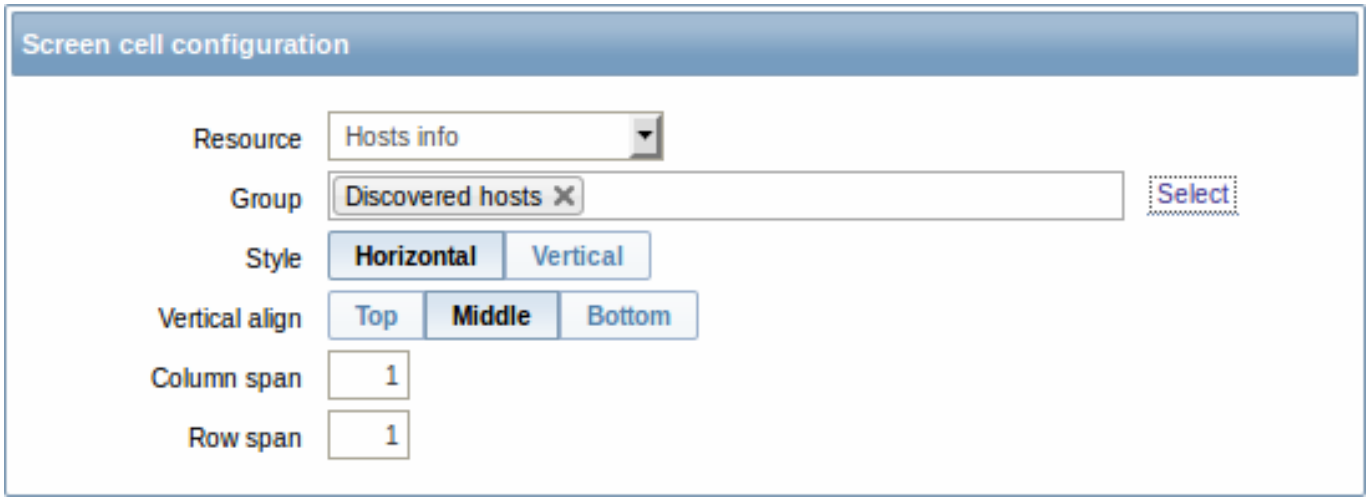
You may set the following specific options:

Host	Select the host.
Show lines	Set how many trigger status lines will be displayed in the screen cell.
Sort triggers by	Select from the dropdown to sort triggers by last change, severity (both descending) or host (ascending).

9 Hosts info

In the host information element you can display high-level information about host availability.

To configure, select *Hosts info* as resource:



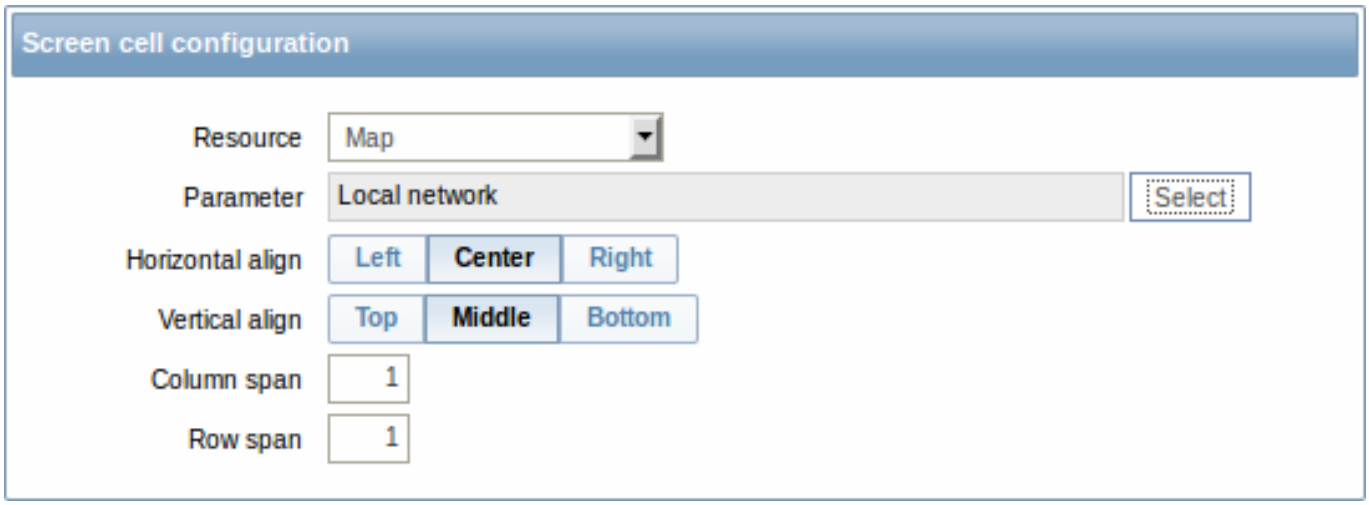
You may set the following specific options:

Group	Select host group(s).
Style	Select vertical or horizontal display.

10 Map

In the map element you can display a configured network map.

To configure, select *Map* as resource:



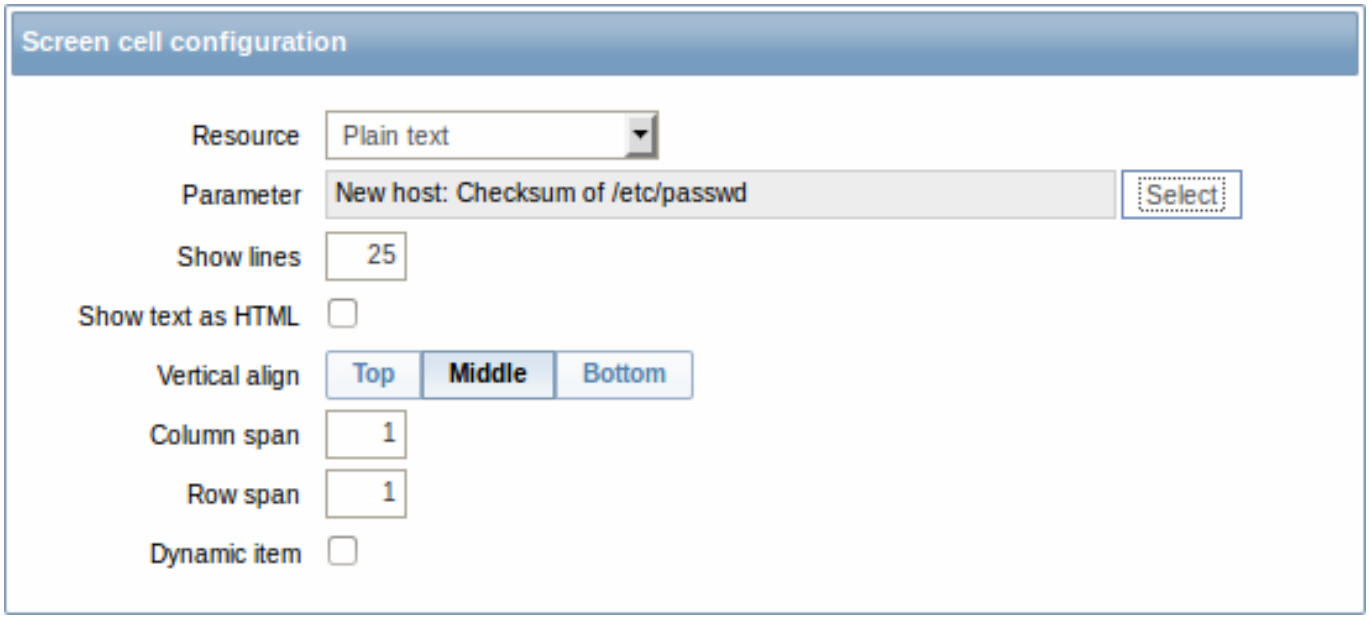
You may set the following specific options:

Parameter	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:



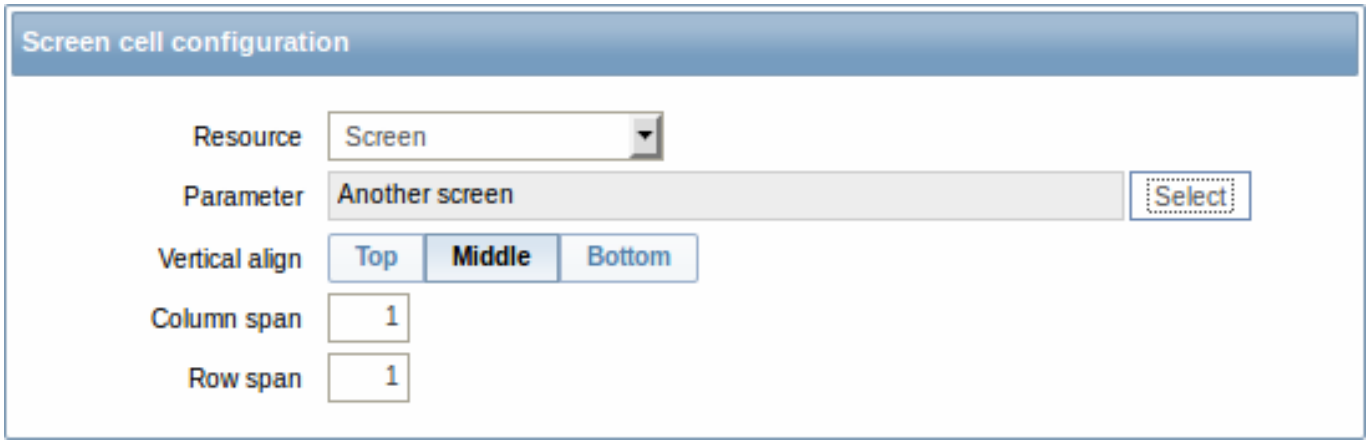
You may set the following specific options:

Parameter	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 Screen

In the screen element you can display another Zabbix screen. One screen may contain other screens inside.

To configure, select *Screen* as resource:



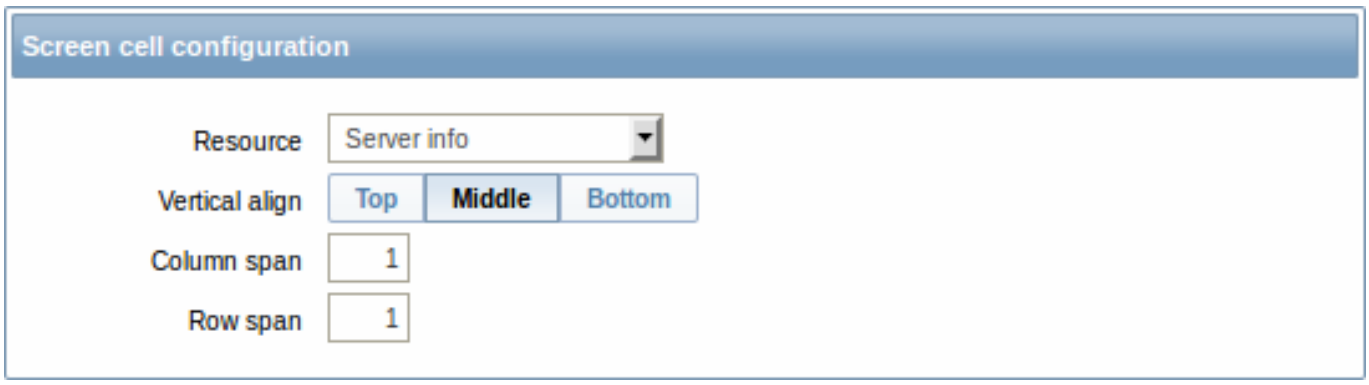
You may set the following specific options:

Parameter	Select the screen to display.

13 Server info

In the server element you can display high-level Zabbix server information.

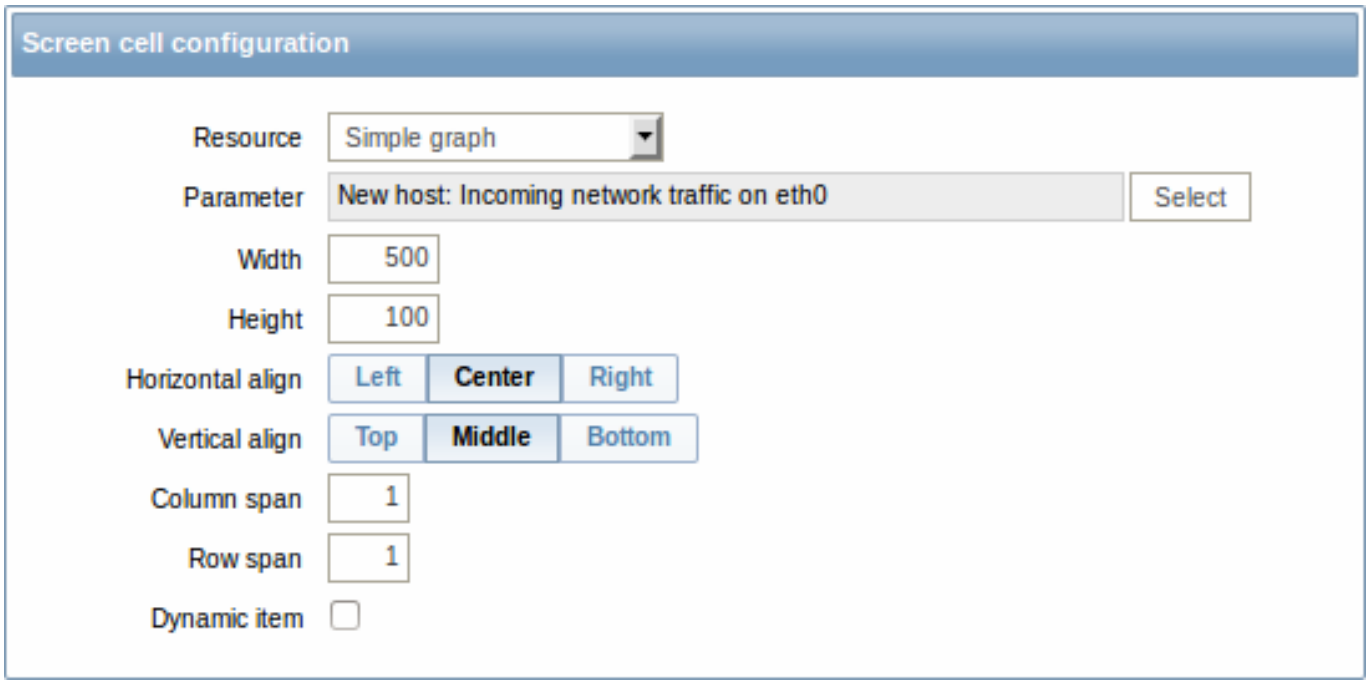
To configure, select *Server info* as resource:



14 Simple graph

In the simple graph element you can display a single simple graph.

To configure, select *Simple graph* as resource:



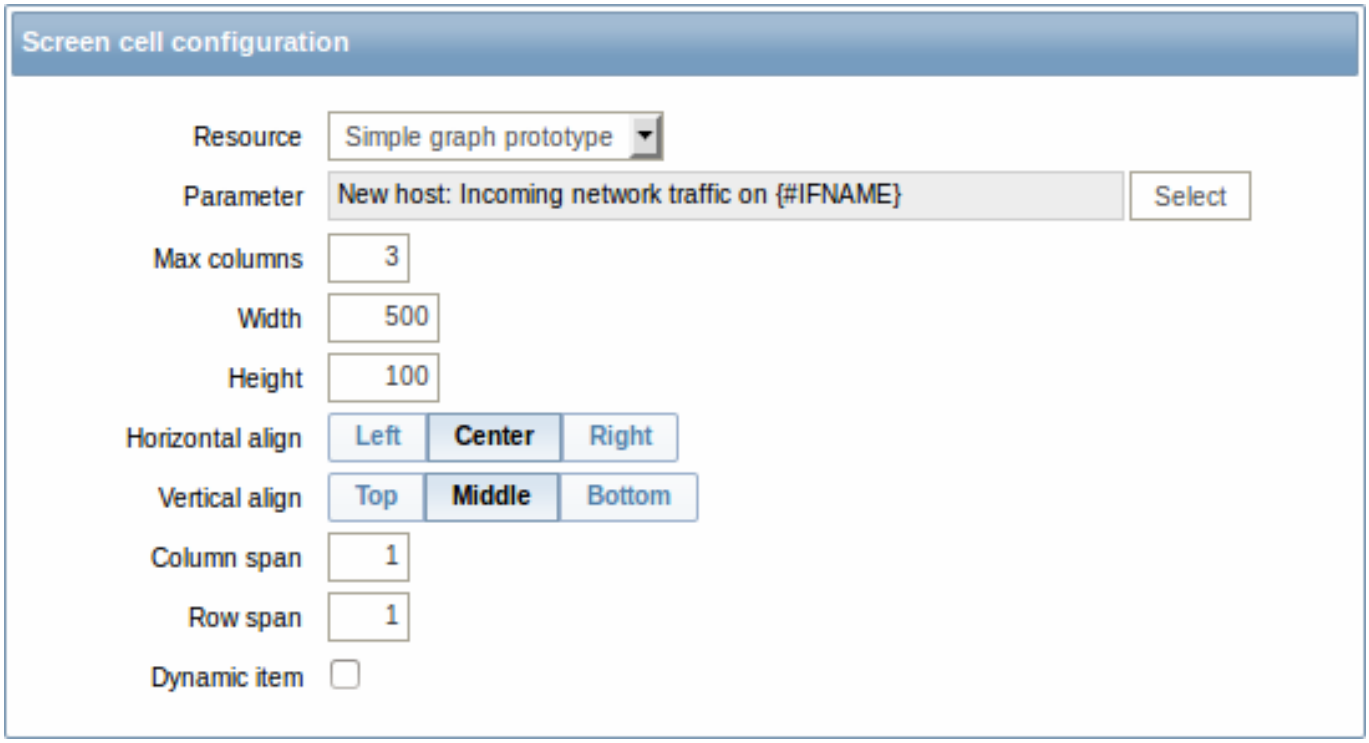
You may set the following specific options:

Parameter	Select the item for the simple graph.
Width	Select graph width.
Height	Select graph height.
Dynamic item	Set graph to display different data depending on the selected host.

15 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:



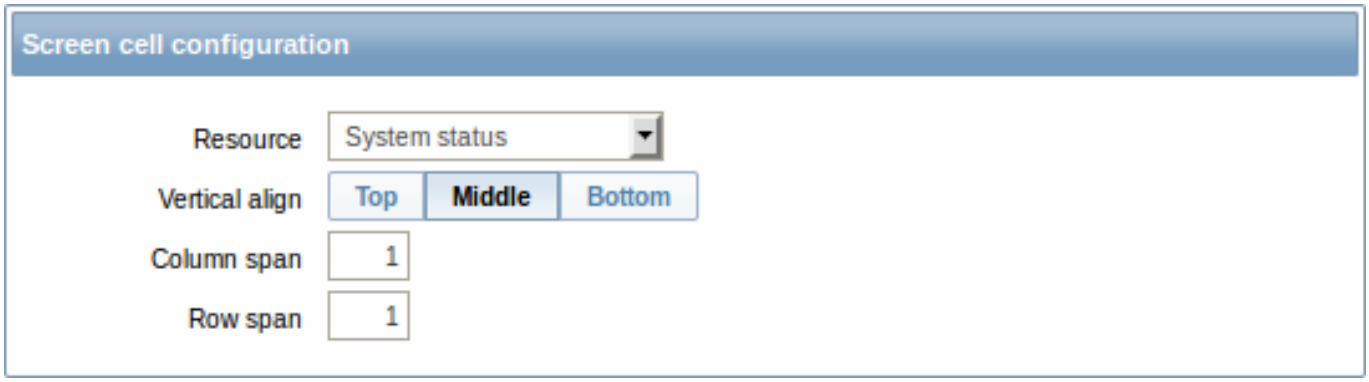
You may set the following specific options:

Parameter	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.
Height	Select graph height.
Dynamic item	Set graph to display different data depending on the selected host.

16 System status

In this element you can display system status similarly as in the Dashboard widget.

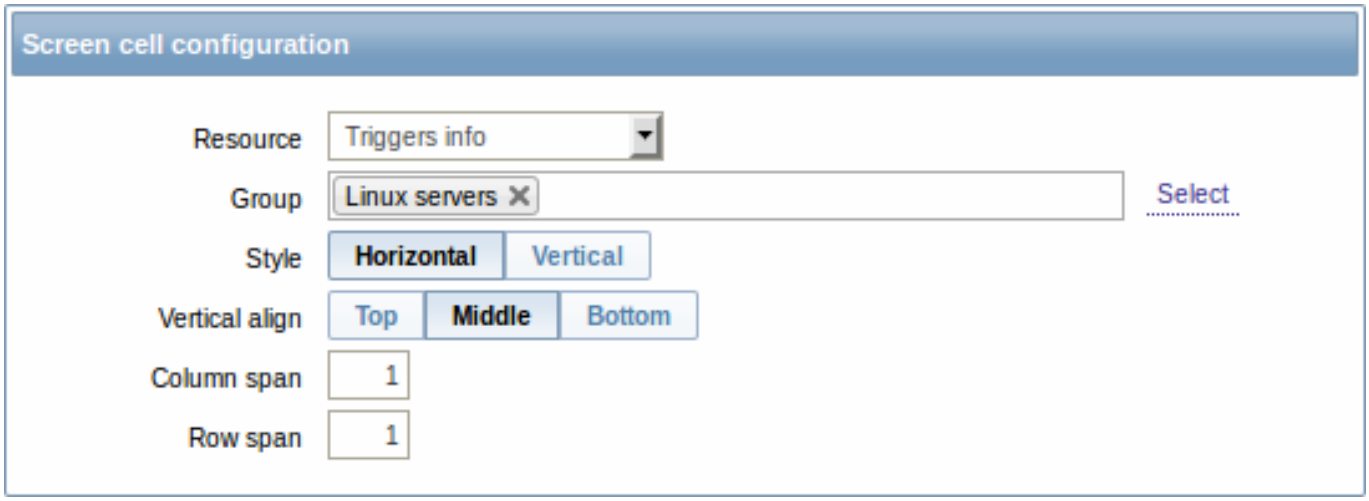
To configure, select *System status* as resource:



17 Triggers info

In the trigger info element you can display high-level information about trigger states.

To configure, select *Triggers info* as resource:



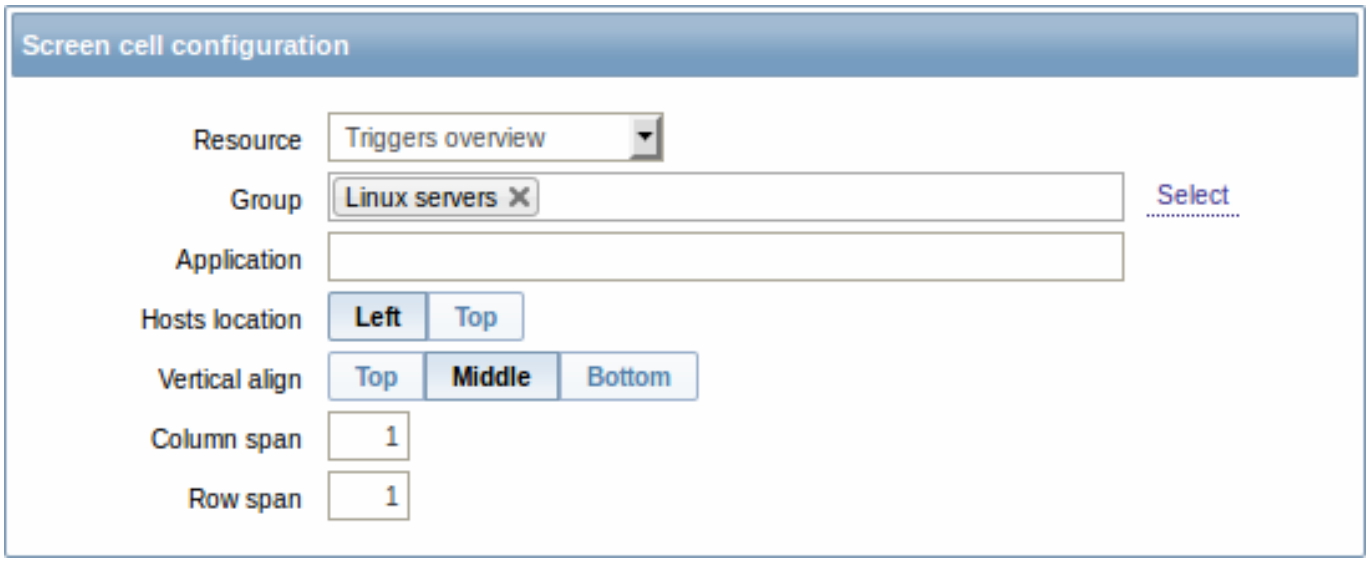
You may set the following specific options:

Group	Select the host group(s).
Style	Select vertical or horizontal display.

18 Triggers 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 *Triggers overview* as resource:



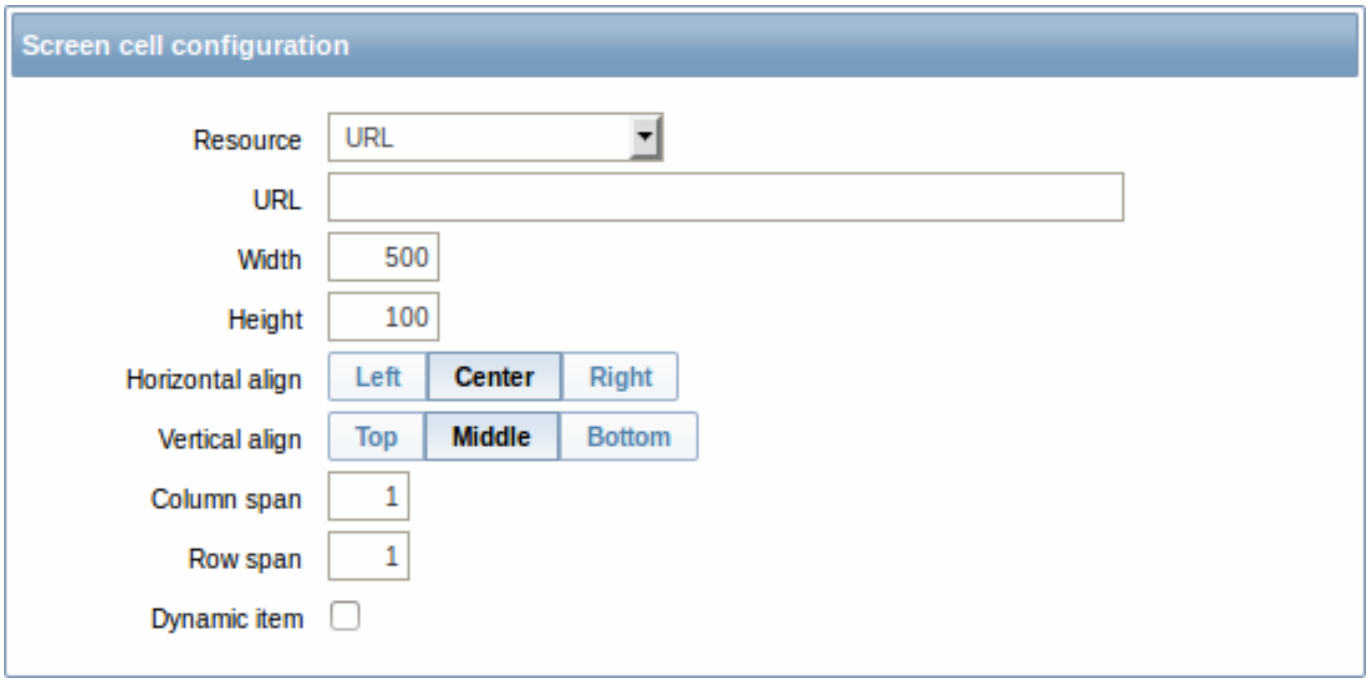
You may set the following specific options:

Group	Select the host group(s).
Application	Enter the application name.
Hosts location	Select host location - left or top.

19 URL

In the URL element you can display a URL content from an external resource.

To configure, select *URL* as resource:



You may set the following specific options:

URL	Enter the URL to display.
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

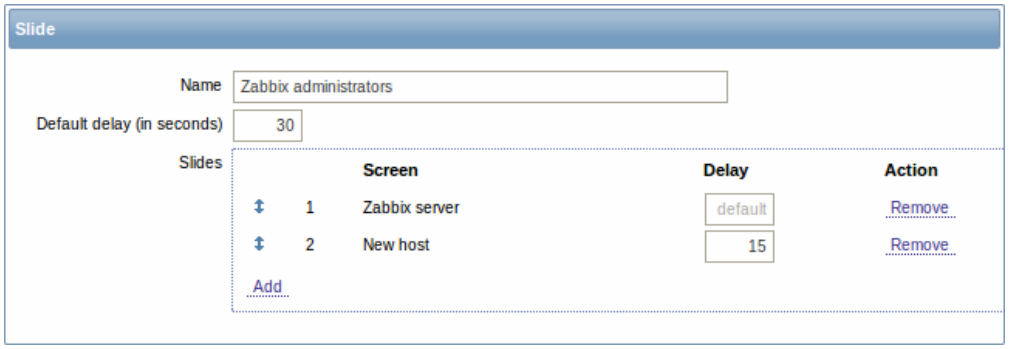
In a slide show you can configure that a number of screens are displayed one after another at set intervals.

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.

Configuration

To create a slide show, do the following:

- Go to *Configuration* → *Slide shows*
- Click on *Create slide show*
- Edit slide show attributes



PARAMETER	Description
Name	Name of the slide show.
Default delay (in seconds)	How long one screen is displayed by default, before rotating to the next, in seconds.
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.

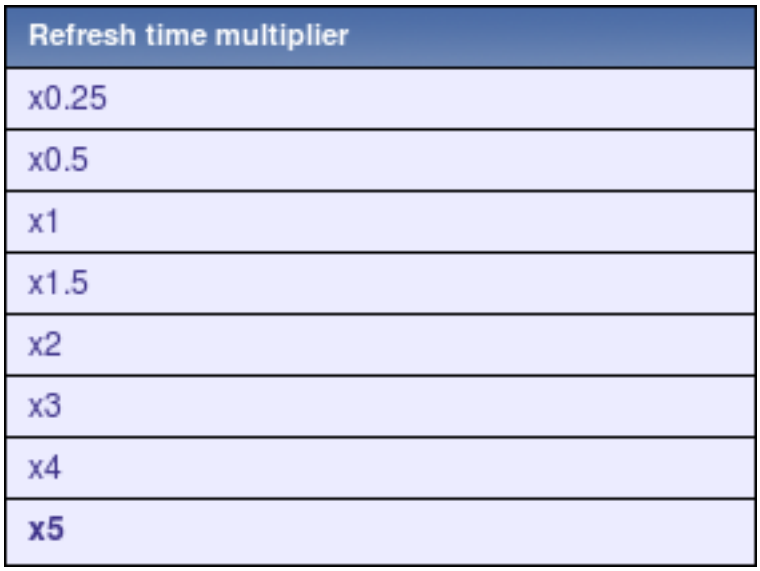
The slide show in this example consists of two screens which will be displayed in the following order:

Zabbix server ⇒ Displayed for 30 seconds ⇒ New host ⇒ Displayed for 15 seconds ⇒ Zabbix server ⇒ Displayed for 30 seconds ⇒ New host ⇒ ...

Display

Slide shows that are ready can be viewed in *Monitoring* → *Screens* and then choosing *Slide shows* from the dropdown.

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.

6 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 (*since Zabbix 2.0*)
- low-level discovery rules (*since Zabbix 2.0*)
- web scenarios (*since Zabbix 2.2*)

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.

7 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 auto-registration 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 in Zabbix.

You can configure several media types:

- E-mail
- SMS
- Jabber
- Ez Texting
- Custom alertscripts

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.

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).

Media type attributes:

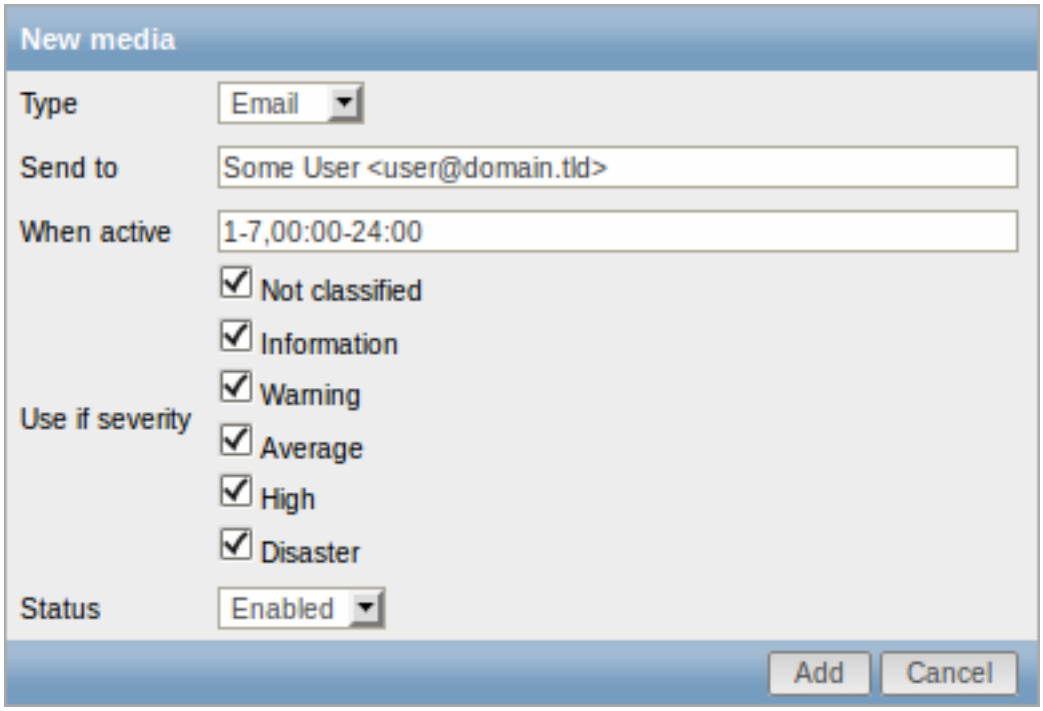
Parameter	Description
Name	Name of the media type.
Type	Select *Email* as the type.
SMTP server	Set an SMTP server to handle outgoing messages.
SMTP helo	Set a correct SMTP helo value, normally a domain name.
SMTP email	The address entered here will be used as the **From** address for the messages sent. Adding a sender display name (like ”Zabbix-HQ” in *Zabbix-HQ <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 HQ <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)

User media

To assign a specific address to the user:

- Go to *Administration→Users*

- Open the user properties form
- In Media tab, click on *Add*



User media attributes:

Parameter	Description
Type	Select *Email* as the type.
Send to	Specify the e-mail address to send the messages to. Adding a recipient display name (like “Some User” in *Some User <user@domain.tld>* in the screenshot above) with the actual e-mail address is supported since Zabbix 2.2 version. See examples and restrictions on display name and email address in media type attribute SMTP email description.
When active	You can limit the time when messages are sent, for example, 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 for non-trigger events the default severity (’Not classified’) is used, so leave it checked if you want to receive notifications for non-trigger events.
Status	Status of the user media. **Enabled** - is in use. **Disabled** - is not being used.

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).

Media type attributes:

Parameter	Description
Description	Name of the media type.
Type	Select *SMS* as the type.
GSM modem	Set the serial device name of the GSM modem.

User media

To assign a phone number to the user:

- Go to *Administration→Users*
- Open the user properties form
- In Media tab, click on *Add*

User media attributes:

Parameter	Description
Type	Select *SMS* as the type.
Send to	Specify the phone number to send messages to.
When active	You can limit the time when messages are sent, for example, 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.
Status	Status of the user media. **Enabled** - is in use. **Disabled** - is not being used.

3 Jabber

Overview

Zabbix supports sending Jabber messages.

When sending notifications, Zabbix tries to look up the Jabber SRV record first, and if that fails, it uses an address record for that domain. Among Jabber SRV records, the one with the highest priority and maximum weight is chosen. If it fails, other records are not tried.

To configure Jabber as the delivery channel for messages, you need to configure Jabber as the media type and enter the respective addresses for the users.

Configuration

To configure Jabber as the media type:

- Go to *Administration→Media types*
- Click on *Create media type* (or click on *Jabber* in the list of pre-defined media types).

Media type attributes:

Parameter	Description
Description	Name of the media type.
Type	Select *Jabber* as the type.
Jabber identifier	Enter Jabber identifier.
Password	Enter Jabber password.

User media

To assign a Jabber address to the user:

- Go to *Administration→Users*
- Open the user properties form
- In Media tab, click on *Add*

User media attributes:

Parameter	Description
Type	Select *Jabber* as the type.
Send to	Specify the address to send messages to.
When active	You can limit the time when messages are sent, for example, 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.
Status	Status of the user media. **Enabled** - is in use. **Disabled** - is not being used.

4 Ez Texting

Overview

You can use Zabbix technological partner Ez Texting for message sending.

To configure Ez Texting as the delivery channel for messages, you need to configure Ez Texting as the media type and assign recipient identification to the users.

Configuration

To configure Ez Texting as the media type:

- Go to *Administration→Media types*
- Click on *Create media type*

Media type attributes:

Parameter	Description
Description	Name of the media type.
Type	Select *Ez Texting* as the type.
Username	Enter the Ez Texting username.
Password	Enter the Ez Texting password.

Parameter	Description
Message text limit	Select the message text limit. **USA (160 characters)** **Canada (136 characters)**

User media

To assign Ez Texting recipient identification to the user:

- Go to *Administration→Users*
- Open the user properties form
- In Media tab, click on *Add*

User media attributes:

Parameter	Description
Type	Select the Ez Texting media type.
Send to	Specify the recipient to send the messages to.
When active	You can limit the time when messages are sent, for example, 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.
Status	Status of the user media. **Enabled** - is in use. **Disabled** - is not being used.

5 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. These scripts are located in the directory defined in the Zabbix server configuration file **AlertScriptsPath** variable. When alert script is executed it gets 3 command-line variables (as $1, $2 and $3 respectively):

- To
- Subject
- Message

> **Note:**
> Alert scripts are executed on the Zabbix server.

The recipient ("To") is specified in user media properties. Here is an example alert script:

```
#####!/bin/bash

to=$1
subject=$2
body=$3

cat <<EOF | mail -s "$subject" "$to"
$body
EOF
```

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*

Media type attributes:

Parameter	Description
Description	Name of the media type.
Type	Select *Script* as the type.
Script name	Enter the name of the script.

User media

To assign custom alertscripts to the user:

- Go to *Administration→Users*
- Open the user properties form
- In Media tab, click on *Add*

User media attributes:

Parameter	Description
Type	Select the custom alertscripts media type.
Send to	Specify the recipient to receive the alerts.
When active	You can limit the time when alertscripts are executed, for example, 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 activate the alertscript for.
Status	Status of the user media. **Enabled** - is in use. **Disabled** - is not being used.

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:

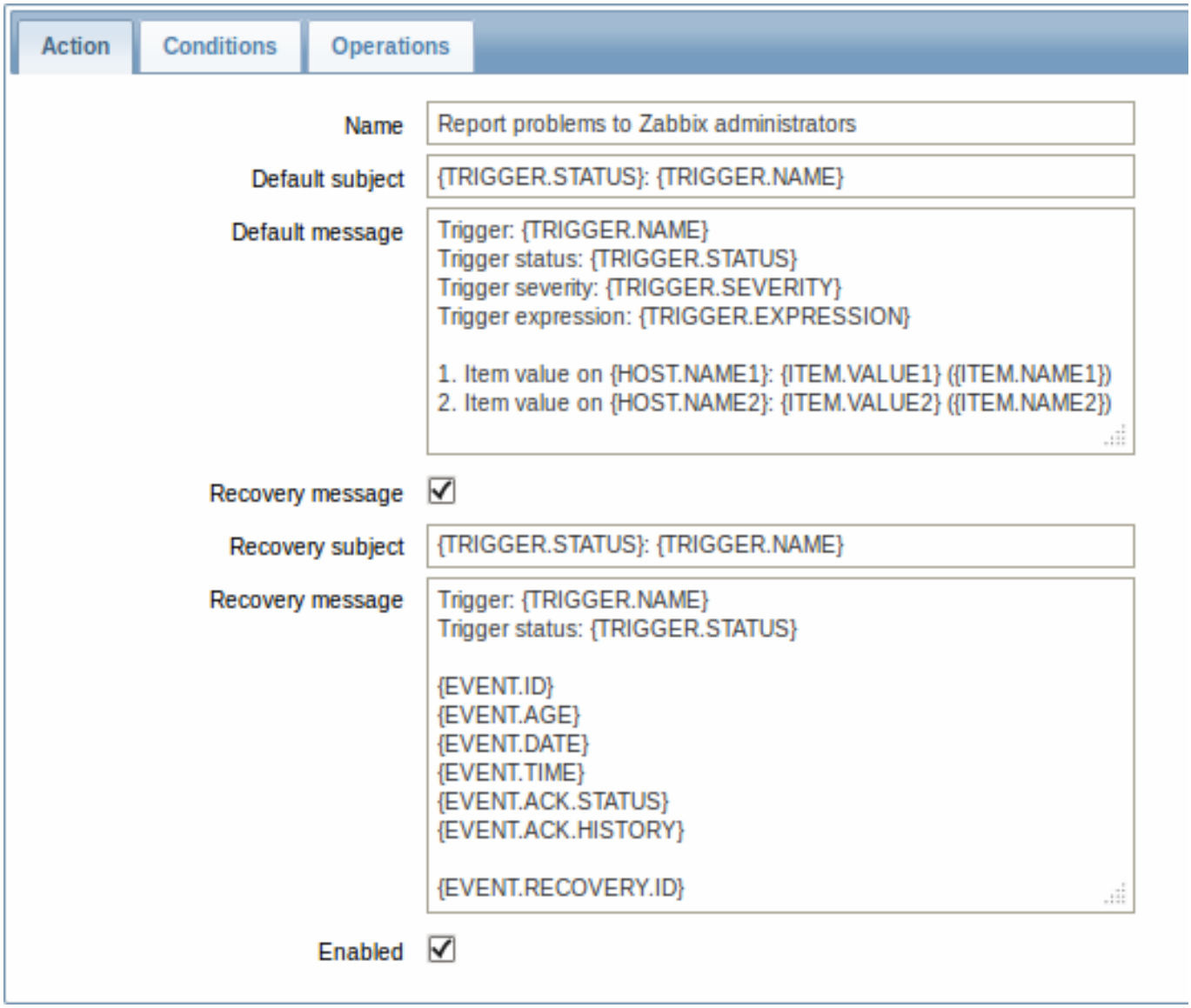
- Trigger events - when trigger status changes from *OK* to *PROBLEM* and back
- Discovery events - when network discovery takes place
- Auto registration events - when new active agents auto-register
- Internal events - when items become unsupported or triggers go into an unknown state

Configuring an action

To configure an action, do the following:

- Go to *Configuration → Actions*
- From the *Event source* dropdown select the required source
- Click on *Create action*
- Set general action attributes
- Choose the operation to carry out, in Operations tab
- Choose the conditions upon which the operation is carried out, in Conditions tab

General action attributes:



Parameter	Description
Name	Unique action name.
Default subject	Default message subject. The subject may contain macros.
Default message	Default message. The message may contain macros.
Recovery message	Mark the checkbox to turn on a *Recovery message*. *Recovery message* is a special way of getting notified for a resolved problem. If turned on, only a **single** message with a custom subject/body is sent if trigger value changes to OK. *Note*: To receive a recovery message, "Trigger value=*Problem*" must be present in action conditions; "Trigger value=*OK*", however, must not be present. (If "Trigger value=*OK*" is set, the recovery message will not work; instead you will get a full escalation of defined messages and/or remote commands in the same way as for a problem situation). *Recovery message* will be sent only to those who received any messages regarding the problem before. A recovery message inherits acknowledgment status and history from the problem event (such as when expanding {EVENT.ACK.HISTORY} and {EVENT.ACK.STATUS} macros). If using {EVENT.*} macros in a recovery message, they will refer to the problem event (not the OK event). {EVENT.RECOVERY.*} macros will **only** expanded in a recovery message and will refer to the recovery/OK event.
Recovery subject	Recovery message subject. It may contain macros.
Recovery message	Recovery message. It may contain macros.

Parameter	Description
Enabled	Mark the checkbox to enable the action. Otherwise it will be disabled.

1 Operations

Overview

You can define the following operations for all events:

- send a message
- execute a remote command (including IPMI)

For discovery events, there are additional operations available:

- add host
- remove host
- enable host
- disable host
- add to group
- delete from group
- link to template
- unlink from template

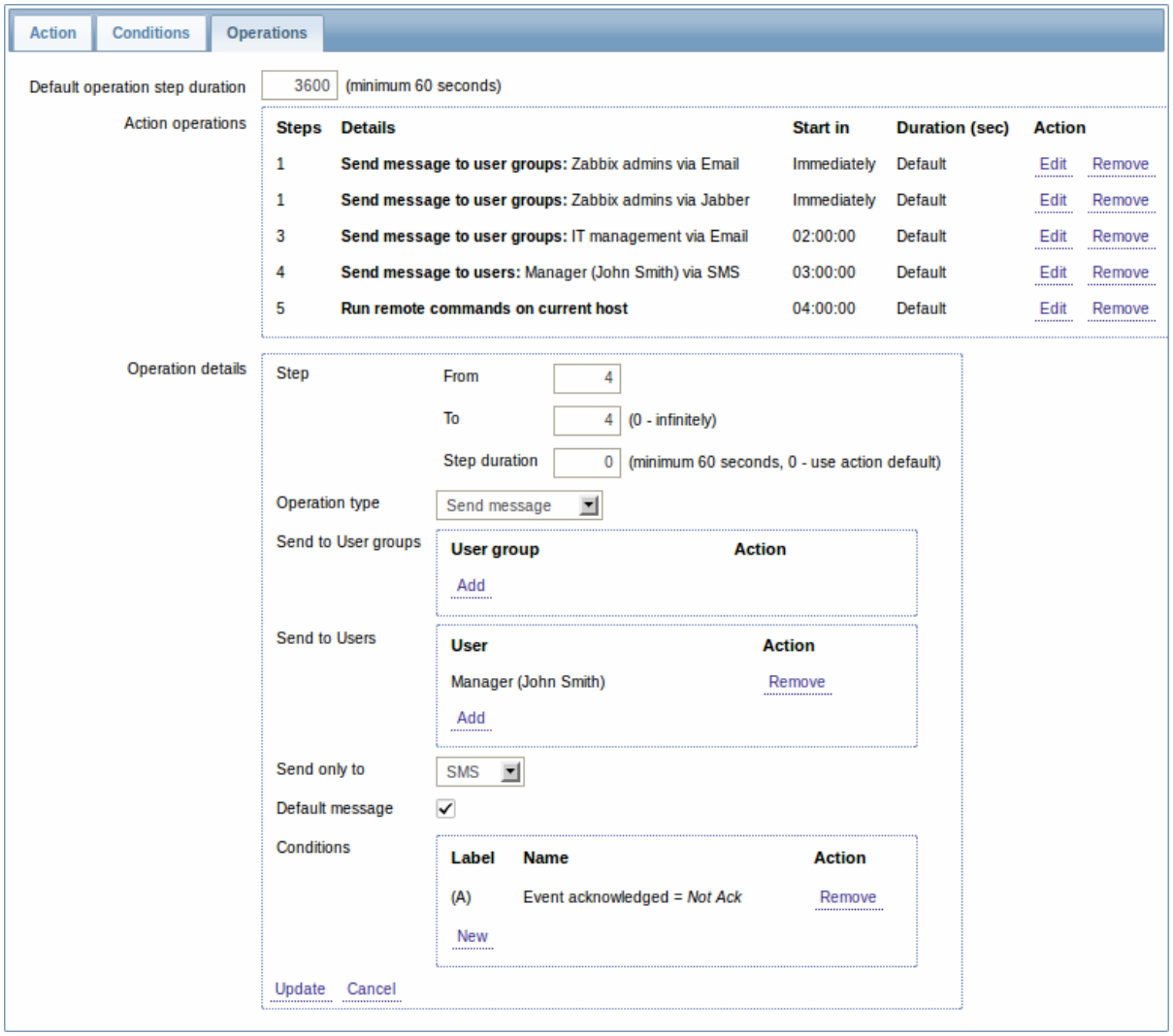
The additional operations available for auto-registration events are:

- add host
- disable host
- add to group
- link to template

Configuring an operation

To configure an operation, go to *Operations* tab in the action properties form and click on *New*. Edit the operation step and click on *Add* to add to the list of *Action operations*.

Operation attributes:



Parameter	Description
Default operation step duration	Duration of one operation step by default (minimum 60 seconds). For example, an hour-long step duration means that if an operation is carried out, an hour will pass before the next step.
Action operations	Action operations are displayed, with these details: **Steps** - escalation step(s) to which the operation is assigned **Details** - type of operation and its recipient/target. Since Zabbix 2.2, the operation list also displays the media type (e-mail, SMS, Jabber, etc) used in sending a message 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. To configure a new operation, click on *New*.

Parameter			Description
Operation details			This block is used to configure the details of an operation.
	Step		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). 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.
	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 auto-registration based events (see above).
		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.
		Default message	If selected, the default message will be used (see general action attributes).
		Subject	Subject of the custom message. The subject may contain macros.
		Message	The custom message. The message may contain macros.
		Operation type: remote command	

Parameter		Description
	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. 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 server or Zabbix agent. 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).
Conditions		Condition for performing the operation: **Not ack** - only when the event is unacknowledged **Ack** - only when the event is acknowledged.

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
- 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.

Tracking messages

You can view the status of messages sent in *Monitoring* → *Events*.

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 *Administration* → *Audit*, if you select *Actions* from the dropdown, 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.

Attention:
Remote commands are not supported to be executed on Zabbix agents monitored by Zabbix proxy, so for commands from Zabbix server to agent a direct connection is required.

Remote commands are limited to 255 characters. Multiple commands can be executed one after another by placing them on a new line. Remote commands may contain macros. See also the command execution page.

Remote commands are executed even if the target host is in maintenance.

The following tutorial provides step-by-step instructions on how to set up remote commands.

Configuration

Those remote commands that are executed on Zabbix agent (custom scripts) must be first enabled in the respective zabbix_agentd.conf.

Make sure that the **EnableRemoteCommands** parameter is set to **1** and uncommented. 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*:

- In the *Operations* tab, select the **Remote command** operation type
- Select the remote command type (IPMI, Custom script, SSH, Telnet, Global script)

- 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 (mark the respective radio button against *Execute on*).

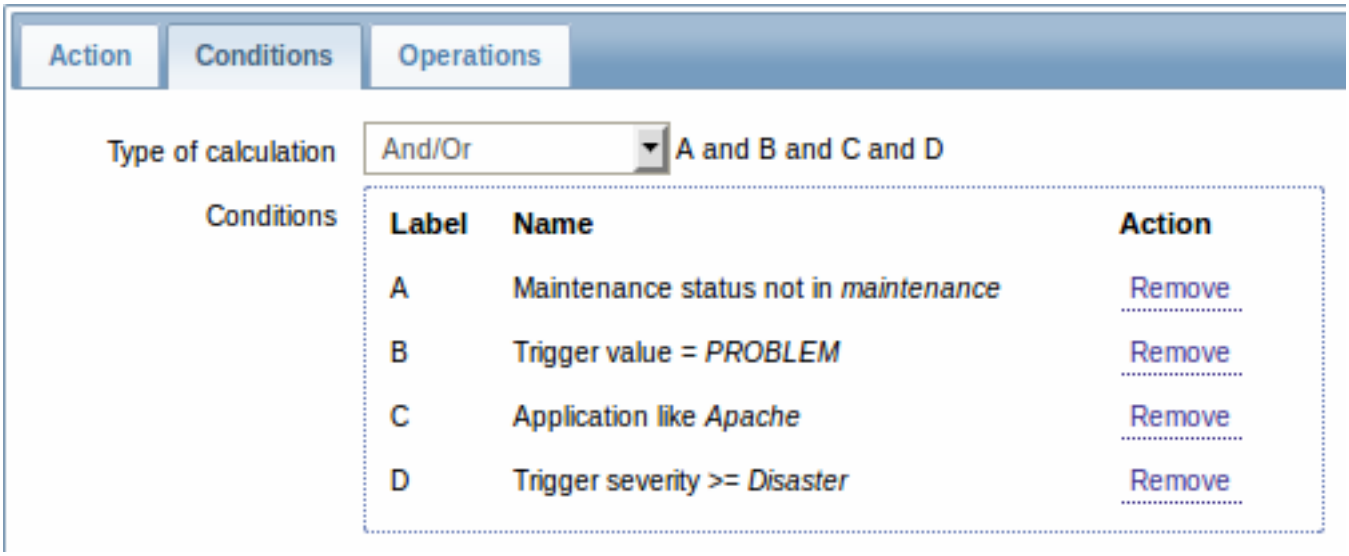
Attention:
Note the use of **sudo** - Zabbix user does not have permissions to restart system services by default. See below for hints on how to configure **sudo**.

Note:
Zabbix agent should run on the remote host and accept incoming connections. Zabbix agent executes commands in background.

Attention:
Zabbix does not check if a command has been executed successfully.

Remote commands on Zabbix agent are executed without timeout by the system.run[,nowait] key. On Zabbix server remote commands are executed with timeout as set in the TrapperTimeout parameter of zabbix_server.conf file.

- In the *Conditions* tab, define the appropriate conditions. In this example, set that the action is activated upon any disaster problems with one of Apache applications.



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.

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

For discovery events, there are additional operations available:

- add host
- remove host
- enable host
- disable host
- add to group
- delete from group
- link to template
- unlink from template

The additional operations available for auto-registration events are:

- add host
- disable host

- add to group
- link to template

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:

```
{TRIGGER.NAME}: {TRIGGER.STATUS}
```

When you receive the message, the message subject will be replaced by something like:

```
Processor load is too high on server zabbix.zabbix.com: PROBLEM
```

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/events.php?triggerid={TRIGGER.ID}&filter_set=1
```

When you receive the message, it will contain a link to all events of the problem trigger.

Example 5

Informing about values from several hosts in a trigger expression.

Message:

```
Trigger: {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:

```
Trigger: 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 acknowledgement: {EVENT.ACK.STATUS}
Event acknowledgement history: {EVENT.ACK.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}
```

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: 2014.01.02
Event age: 5m
Event acknowledgement: Yes
Event acknowledgement history: 2014.01.02 13:05:51 "John Smith (Admin)"
-acknowledged-

Recovery:

Event ID: 21896
```

```
Event value: 0
Event status: OK
Event time: 13:10:07
Event date: 2014.01.02
```

Attention:
Separate notification macros for the original problem event and recovery event are supported since Zabbix 2.2.0.

2 Conditions

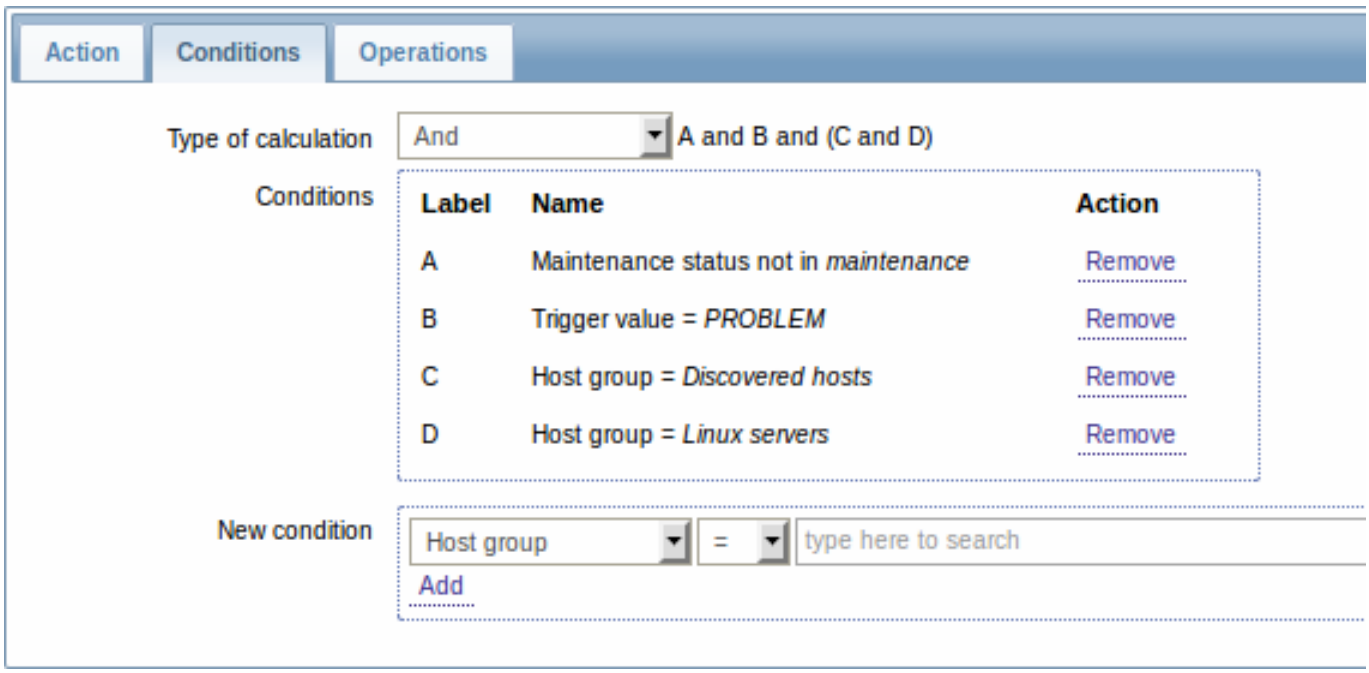
Overview

An action is executed only in case an event matches a defined set of conditions.

Configuration

To set a condition:

- Go to *Conditions* tab in the action properties form
- Select conditions from the *New condition* dropdowns and click on *Add*
- Select the type of calculation (with more than one condition)



The following conditions can be set for trigger-based actions:

Condition type	Supported operators	Description
Application	= like not like	Specify an application or an application to exclude. **=** - event belongs to a trigger of the item that is linked to the specified application. **like** - event belongs to a trigger of the item that is linked to an application containing the string. **not like** - event belongs to a trigger of the item that is linked to an application not containing the string.

Condition type	Supported operators	Description
Host group	= <>	Specify host groups or host groups to exclude. **=** - event belongs to this host group. **<>** - event does not belong to this host group.
Template	= <>	Specify templates or templates to exclude. **=** - event belongs to a trigger inherited from this template. **<>** - event does not belong to a trigger inherited from this template.
Host	= <>	Specify hosts or hosts to exclude. **=** - event belongs to this host. **<>** - event does not belong to this host.
Trigger	= <>	Specify triggers or triggers to exclude. **=** - event is generated by this trigger. **<>** - event is generated by any other trigger, except this one.
Trigger name	like not like	Specify a string in the trigger name or a string to exclude. **like** - event is generated by a trigger, containing this string in the name. Case sensitive. **not like** - this string cannot be found in the trigger name. Case sensitive. *Note*: Entered value will be compared to trigger name with all macros expanded.
Trigger severity	= <> >= <=	Specify trigger severity. **=** - equal to trigger severity **<>** - not equal to trigger severity **>=** - more or equal to trigger severity **<=** - less or equal to trigger severity
Trigger value	=	Specify a trigger value. **=** - equal to trigger value (OK or PROBLEM)
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 Time period specification page for description of the format.

Condition type	Supported operators	Description
Maintenance status	in not in	Specify a host in maintenance or not in maintenance. **in** - host is in maintenance mode. **not in** - host is not in maintenance mode. *Note*: If several hosts are involved in the trigger expression, the condition matches if at least one of the hosts is/is not in maintenance mode.

Trigger value:

- if a trigger changes status from OK to PROBLEM, trigger value is PROBLEM
- if a trigger changes status from PROBLEM to OK, trigger value is OK

Note:
When a new action for triggers is created, it gets two automatic conditions (both can be removed by the user):
- "Trigger value = *PROBLEM*" - so that notifications are sent for problems only. This means that if you configure an action without any more specific conditions, messages will be received for any problem. Having this condition by default is also important if you want to receive a single recovery message.
- "Maintenance status = not in *maintenance*" - so that notifications are not sent for hosts in maintenance.

The following conditions can be set for discovery-based events:

Condition type	Supported operators	Description
Host IP	= <>	Specify an IP address range or a range to exclude for a discovered host. **=** - host IP is in the range. **<>** - host IP is not in the range. It may have the following formats: Single IP: 192.168.1.33 Range of IP addresses: 192.168.1.1-254 (until Zabbix 2.4.4) 192.168.1-10.1-254 (from Zabbix 2.4.4) 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

Condition type	Supported operators	Description
Service type	= <>	Specify a service type of a discovered service or a service type to exclude. **=** - matches the discovered service. **<>** - 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	= <>	Specify a TCP port range of a discovered service or a range to exclude. **=** - service port is in the range. **<>** - service port is not in the range.
Discovery rule	= <>	Specify a discovery rule or a discovery rule to exclude. **=** - using this discovery rule. **<>** - using any other discovery rule, except this one.
Discovery check	= <>	Specify a discovery check or a discovery check to exclude. **=** - using this discovery check. **<>** - using any other discovery check, except this one.
Discovery object	=	Specify the discovered object. **=** - equal to discovered object (a device or a service).
Discovery status	=	**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	>= <=	Uptime for 'Host Up' and 'Service Up' events. Downtime for 'Host Down' and 'Service Down' events. **>=** - is more or equal to. Parameter is given in seconds. **<=** - is less or equal to. Parameter is given in seconds.

Condition type	Supported operators	Description
Received value	= <> >= <= like not like	Specify the value received from an agent (Zabbix, SNMP). Case sensitive string comparison. If multiple Zabbix agent or SNMP checks are configured for a rule, received values for all of them are checked (each check generates a new event which is matched against all conditions). **=** - equal to the value. **<>** - not equal to the value. **>=** - more or equal to the value. **<=** - less or equal to the value. **like** - contains the substring. Parameter is given as a string. **not like** - does not contain the substring. Parameter is given as a string.
Proxy	= <>	Specify a proxy or a proxy to exclude. **=** - using this proxy. **<>** - 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.

The following conditions can be set for actions based on active agent auto-registration:

Condition type	Supported operators	Description
Host metadata	like not like	Specify host metadata or host metadata to exclude. **like** - host metadata contains the string. **not like** - host metadata does not contain the string. Host metadata can be specified in an agent configuration file.
Host name	like not like	Specify a host name or a host name to exclude. **like** - host name contains the string. **not like** - host name does not contain the string.
Proxy	= <>	Specify a proxy or a proxy to exclude. **=** - using this proxy. **<>** - using any other proxy except this one.

The following conditions can be set for actions based on internal events:

Condition type	Supported operators	Description
Application	= like not like	Specify an application or an application to exclude. **=** - event belongs to an item that is linked to the specified application. **like** - event belongs to an item that is linked to an application containing the string. **not like** - event belongs to an item that is linked to an application not containing the string.
Event type	=	**Item in "not supported" state** - matches events where an item goes from a 'normal' to 'not supported' state **Item in "normal" state** - matches events where an item goes from a 'not supported' to 'normal' 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 ** Low-level discovery rule in "normal" state** - matches events where a low-level discovery rule goes from a 'not supported' to 'normal' state **Trigger in "unknown" state** - matches events where a trigger goes from a 'normal' to 'unknown' state ** Trigger in "normal" state** - matches events where a trigger goes from an 'unknown' to 'normal' state
Host group	= <>	Specify host groups or host groups to exclude. **=** - event belongs to this host group. **<>** - event does not belong to this host group.
Template	= <>	Specify templates or templates to exclude. **=** - event belongs to an item/trigger/low-level discovery rule inherited from this template. **<>** - event does not belong to an item/trigger/low-level discovery rule inherited from this template.

Condition type	Supported operators	Description
Host	= <>	Specify hosts or hosts to exclude. **=** - event belongs to this host. **<>** - 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 "And" calculation should not be used 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 = Oracle servers
Host group = MySQL servers
Trigger name like 'Database is down'
Trigger name like 'Database is unavailable'

is evaluated as

(Host group = Oracle servers **or** Host group = MySQL servers**) and (**Trigger name like 'Database is down' **or** Trigger name like '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).

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
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);
- proxies ("proxy" condition).

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.

3 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
- Recovery messages can be sent

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.

Miscellaneous aspects of escalation behaviour

Let’s consider what happens in different circumstances if an action contains several escalation steps.

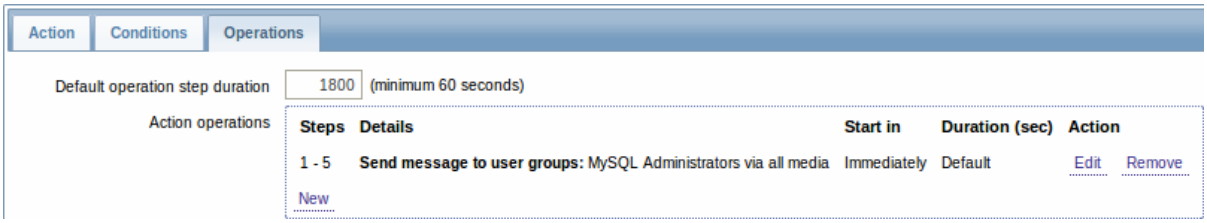
Situation	Behaviour
The host in question goes into maintenance after the initial problem notification is sent	All remaining escalation steps are executed. A maintenance cannot stop operations; maintenance has effect with regard to when actions are started/not started, not operations.
*The time period 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.
A problem starts during maintenance and continues (is not resolved) after maintenance ends	All escalation steps are executed starting from the moment maintenance ends.
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.
An action is disabled during an escalation in progress (like a message being sent)	The message in progress will be sent and then one more message on the escalation will be sent. The follow-up message will have the following text at the beginning of the message body: *NOTE: Escalation cancelled: action ’<Action name>’ disabled*. This way the recipient is informed that the escalation is cancelled and no more steps will be executed. This message is sent to the recipients defined in the following escalation step.

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 ’1800’ seconds (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).

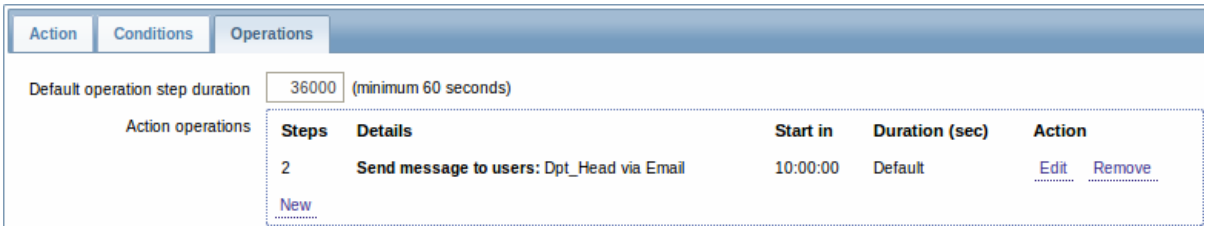
If the problem is resolved and a recovery message is configured, it will be sent to those who received at least one problem message within this escalation scenario.

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 '36000' seconds (10 hours)
- Set the escalation steps to be *From* '2' *To* '2'



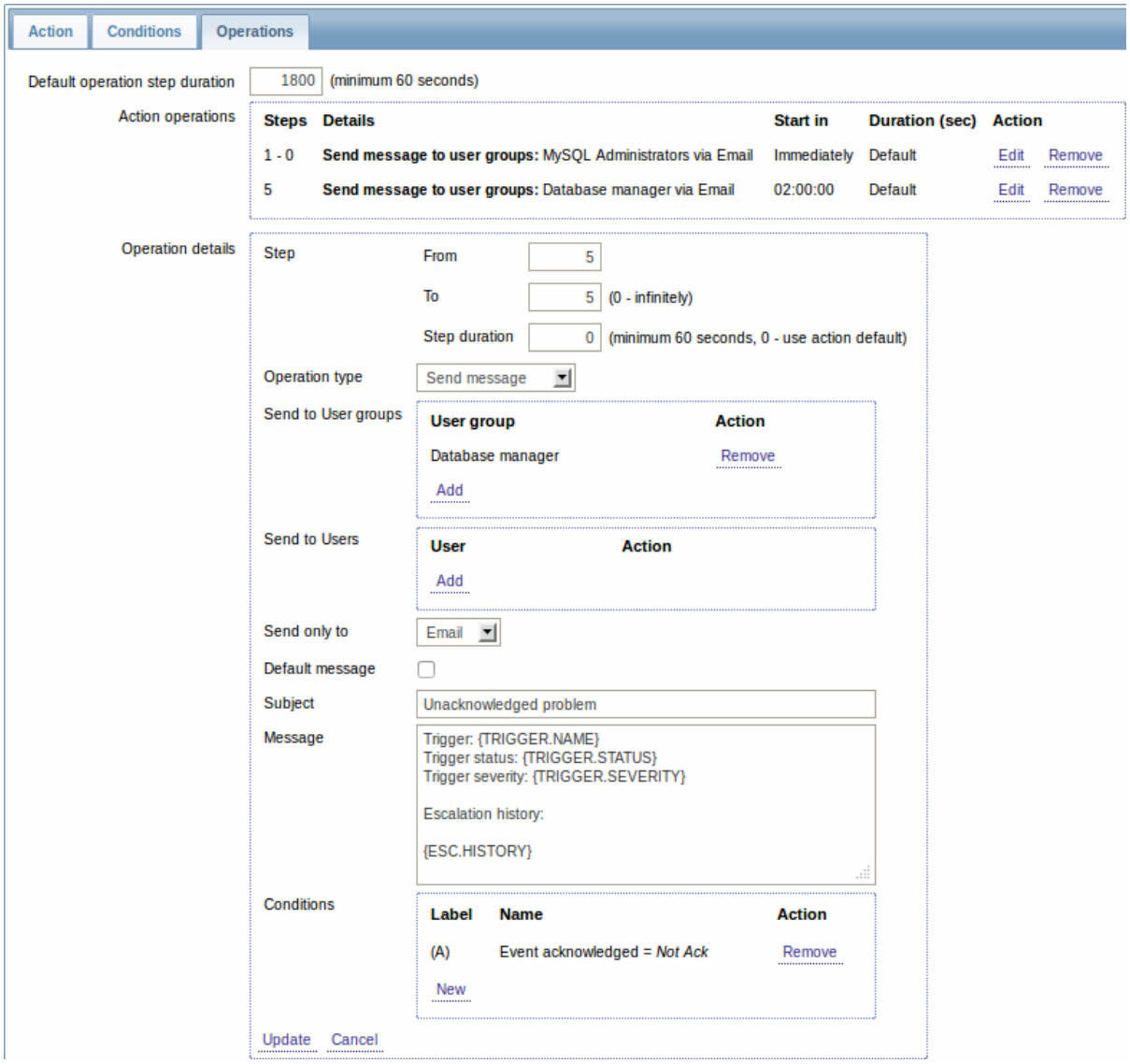
A notification will only be sent at Step 2 of the escalation scenario, or 10 hours after the problem starts.

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.



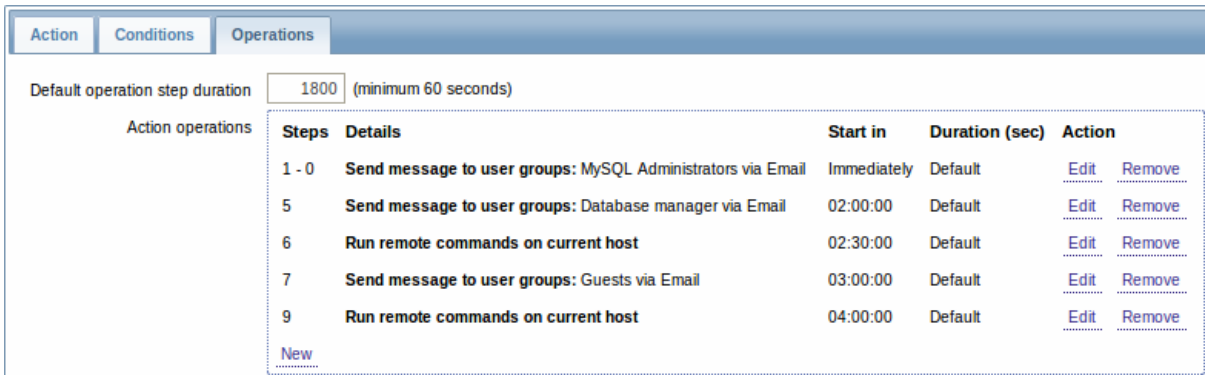
Note the use of {ESC.HISTORY} macro in the message. The macro will contain information about all previously executed steps on this escalation, such as notifications sent and commands executed.

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.

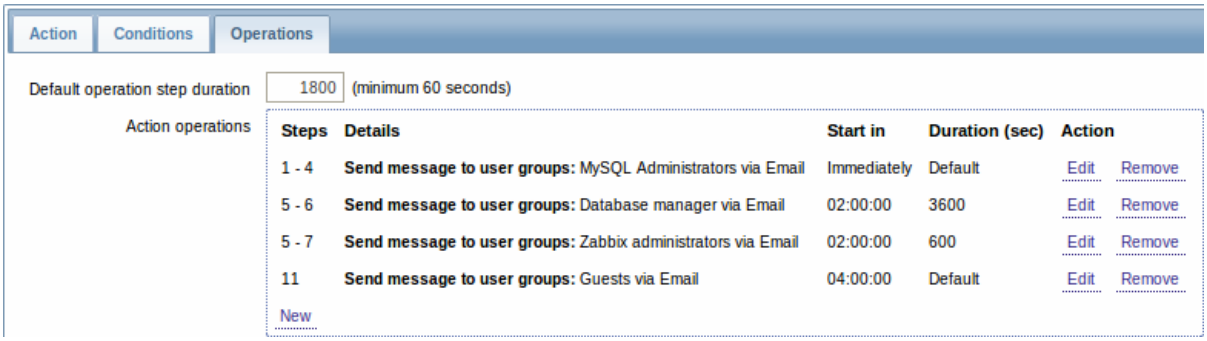
If the problem still exists, after another 30 minutes Zabbix will send a message to all guest users.

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, the shorter custom step duration of 600 seconds in the next operation overrides the longer custom step duration of 3600 seconds tried to set here)
- to Zabbix administrators at 2:00, 2:10, 2:20 after the problem starts (the custom step duration of 600 seconds 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.

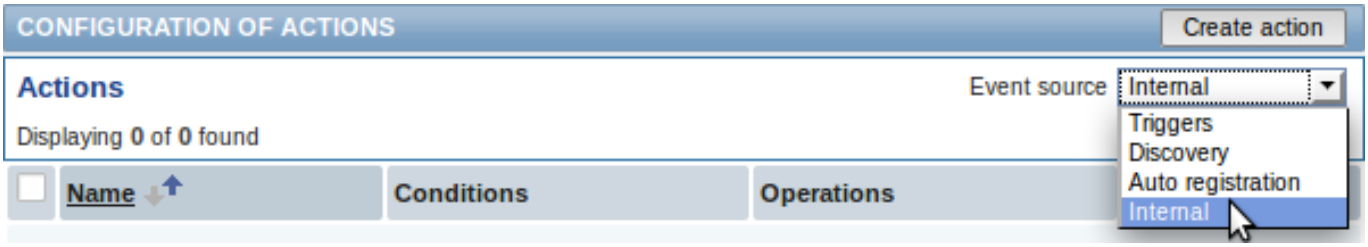
Step 1

Configure some media, such as e-mail, SMS or Jabber, 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.

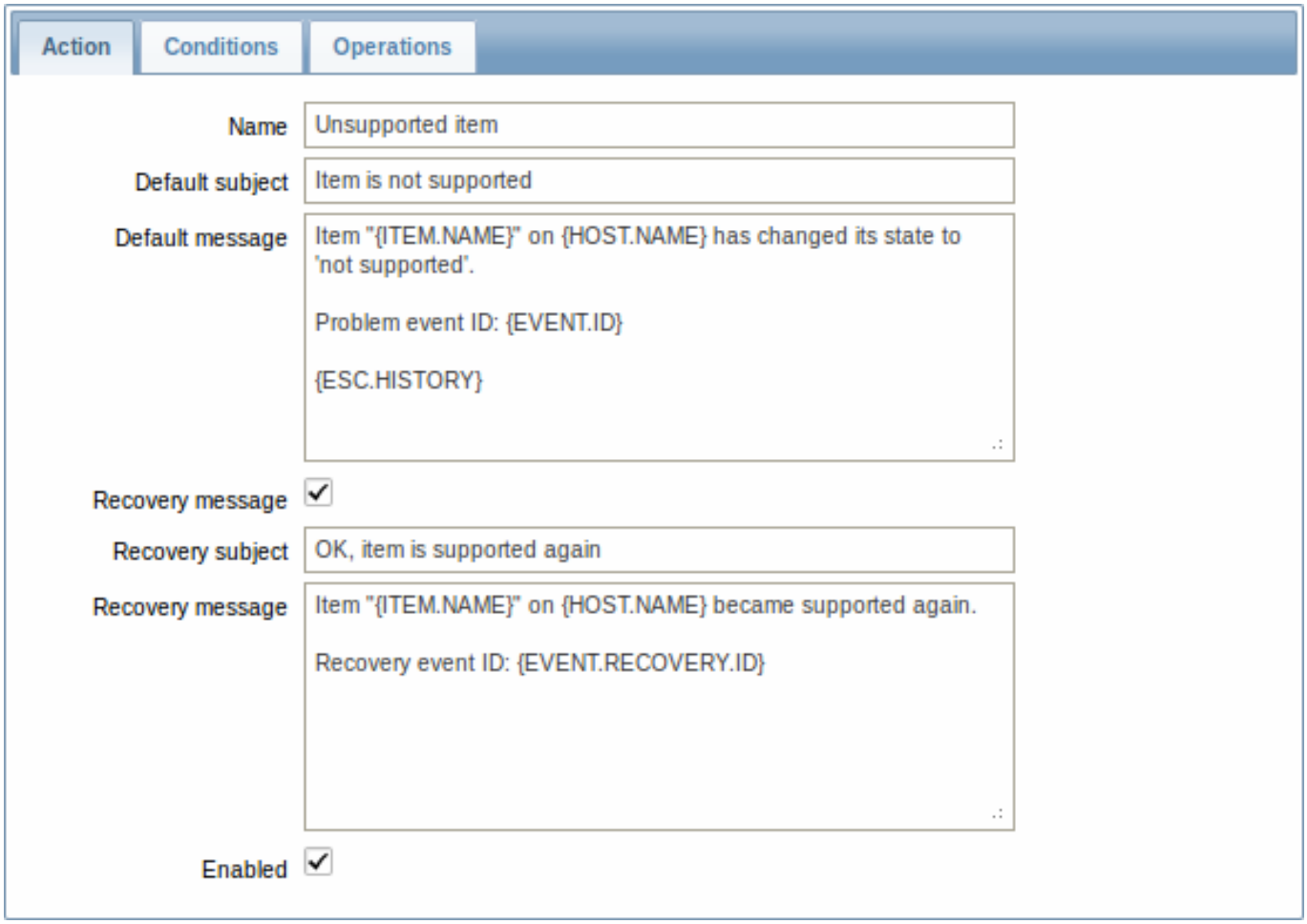
Step 2

Go to *Configuration→Actions* and select *Internal* as the event source. Click on *Create action* on the upper right to open an action configuration form.



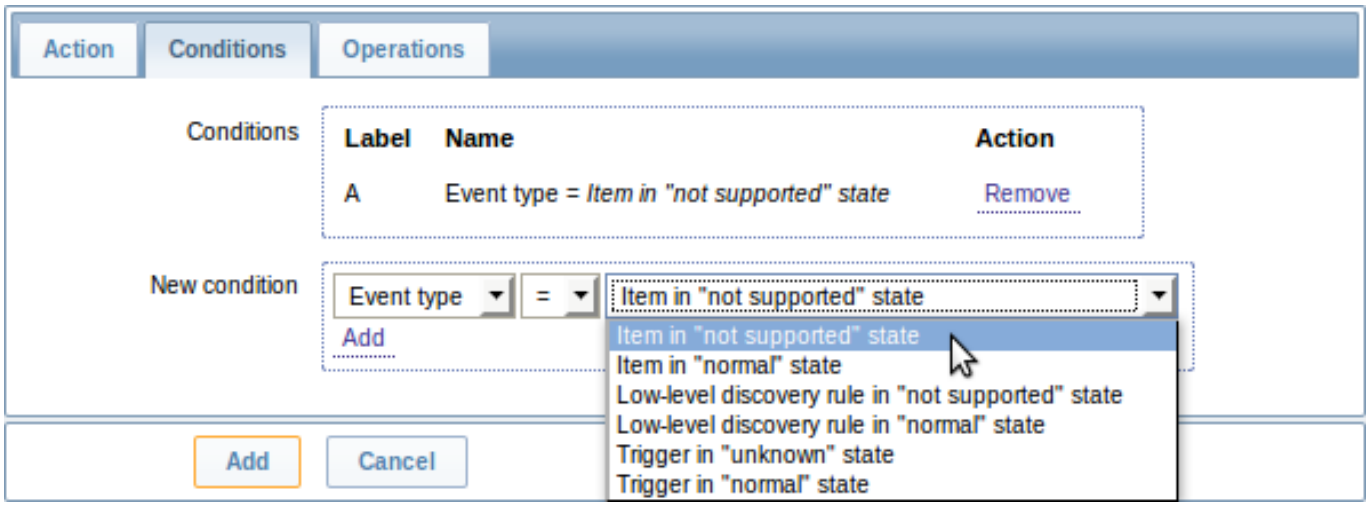
Step 3

In the **Action** tab enter a name for the action and the subject/content of problem and recovery messages.



Step 4

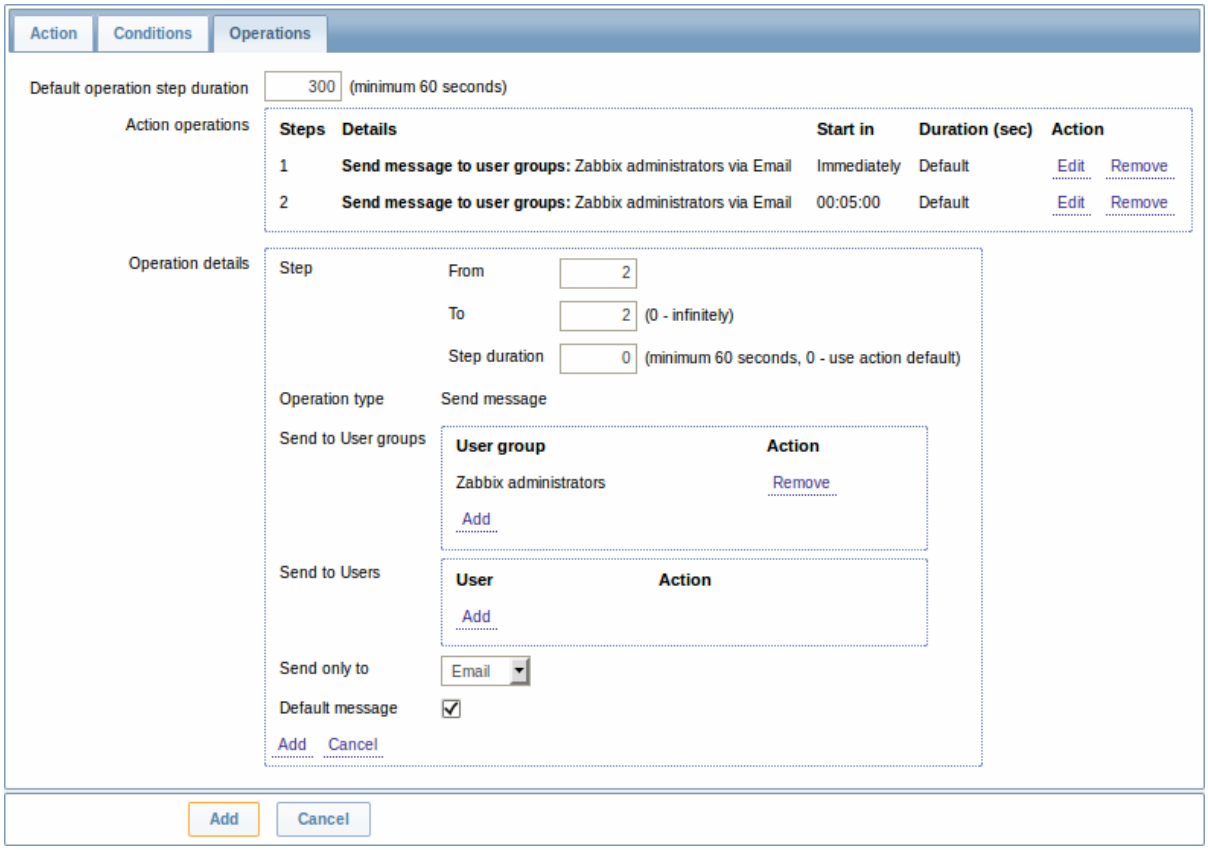
In the **Conditions** tab select *Event type* in the New condition block and select *Item in "not supported" state* as the value.



Don't forget to click on *Add* to actually list the condition in the *Conditions* block.

Step 5

In the **Operations** tab, click on *New* and select some recipients of the message (user groups/users) and the media types (or 'All') to use for delivery.



Click on *Add* in the *Operation details* block to actually list the operation in the *Action operations* block.

If you wish to receive more than one notification, set the operation step duration (interval between messages sent) and add another operation.

When finished, click on the **Add** button underneath 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.

8 Macros

Overview

Zabbix supports a number of macros which may be used in various situations. Macros are variables, identified by a **{MACRO}** syntax, and 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.

See a full list of supported macros by location.

You can also configure your own user macros.

1 User macros

Overview

For greater flexibility, Zabbix supports user macros, which can be defined on global, template and host level. These macros have a special syntax: **{$MACRO}**.

The macros can be used in:

- item names
- item key parameters
- trigger names and descriptions
- trigger expression parameters and constants (see examples)
- several other locations

The following characters are allowed in the macro names: **A-Z** , **0-9** , **_** , **.**

Zabbix substitutes 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
5. ...
6. global macros (checked last)

In other words, if a macro does not exist for a host, Zabbix will try to find it in the host templates of increasing depth. If still not found, a global macro will be used, if exists.

If Zabbix is unable to find a macro, the macro will not be substituted.

To define user macros, go to the corresponding locations 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, exporting the template to XML and importing it in another system will still allow it to work as expected.

Most common use cases of global and host macros:

1. taking advantage of templates with host specific attributes: passwords, port numbers, file names, regular expressions, etc
2. global macros for global one-click configuration changes and fine tuning

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()**.

Attention:
In trigger expressions user macros will expand if referencing a parameter or constant. They will NOT expand if referencing the host, item key, function, operator or another trigger expression.

9 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

The initial Zabbix installation has two predefined users - 'Admin' and 'guest'. The 'guest' user is used for unauthenticated users. Before you log in as 'Admin', you are 'guest'. Proceed to configuring a user in Zabbix.

1 Configuring a user

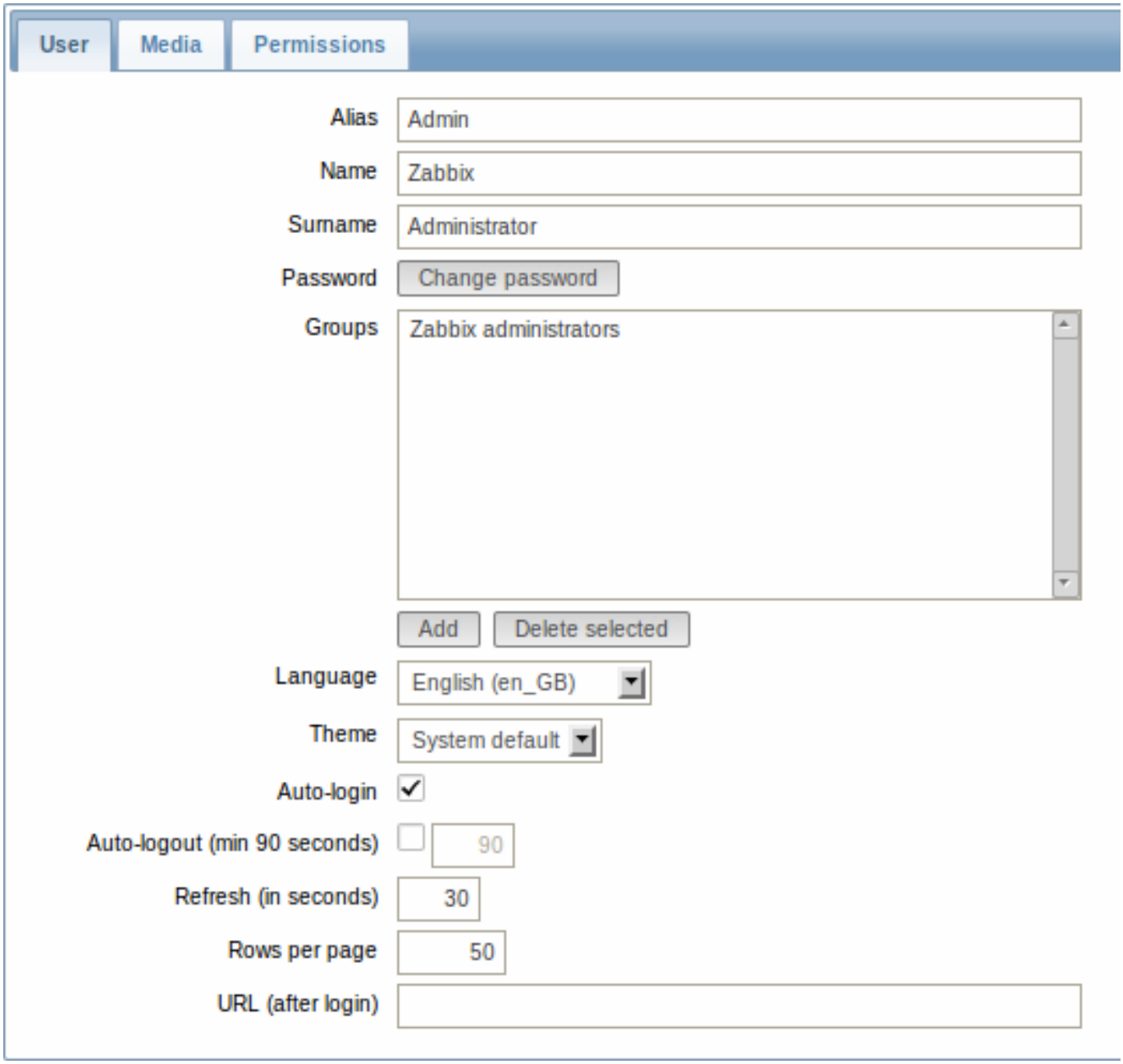
Overview

To configure a user:

- Go to *Administration* → *Users*
- Select *Users* from the dropdown to the right
- 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:



Parameter	Description
Alias	Unique username, used as the login name.
Name	User first name (optional). If not empty, visible in acknowledgement information and notification recipient information.
Surname	User second name (optional). If not empty, visible in acknowledgement information and notification recipient information.
Password	Two fields for entering the user password. With an existing password, contains a *Password* button, clicking on which opens the password fields.
Groups	List of all user groups the user belongs to. Adherence to user groups determines what host groups and hosts the user will have access to. Click on *Add* to add groups.
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 **Original Blue** - standard blue theme **Black & Blue** - alternative theme **Dark orange** - alternative theme

Parameter	Description
Auto-login	Enable if you want Zabbix to remember you and log you in automatically for 30 days. Browser cookies are used for this.
Auto-logout (min 90 seconds)	Mark the checkbox to enable automatic user logout after the set seconds of inactivity (minimum value = 90 seconds).
Refresh (in seconds)	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 to transfer you to a specific URL after successful login, for example, the status of triggers 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 and hosts 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.

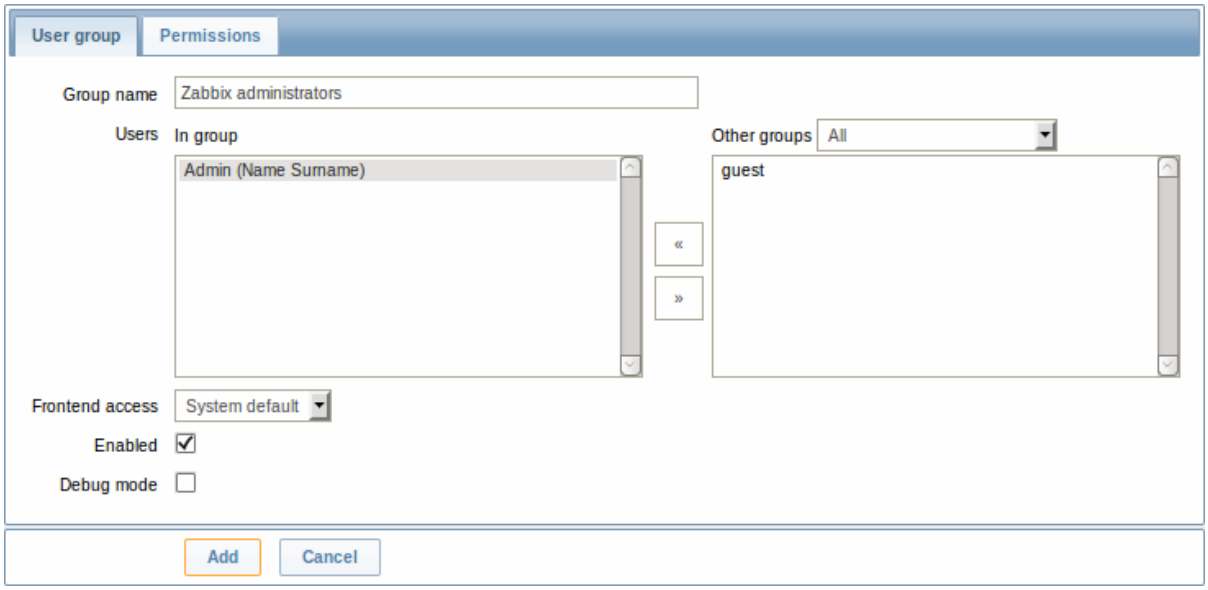
A user can belong to any amount of groups.

Configuration

To configure a user group:

- Go to *Administration → Users*
- Select *User groups* from the dropdown to the right
- 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:



Parameter	Description
Group name	Unique group name.
Users	The **In group** block contains a listing of the members of this group. To add users to the group select them in the *Other groups* block and click on «.
Frontend access	How the users of the group are authenticated. **System default** - use default authentication **Internal** - use Zabbix authentication. Ignored if HTTP authentication is set **Disabled** - access to Zabbix GUI is forbidden
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:

Composing permissions	Click on *Add* beneath the respective list to specify the host groups that the user group will have access to on the level of: **Read-write** - read-write access to a host group **Read** – read-only access to a host group **Deny** – access to a host group denied

Calculated permissions	Depending on the permissions set above, *Calculated permissions* will display all host groups and all hosts that the user group has access to on the level of: **Read-write** - host groups with read-write access **Read** - host groups with read-only access **Deny** - host groups with access denied

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.

7. IT services

Overview IT services are 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 IT services provide answers to all mentioned questions.

IT services is a hierarchy representation of monitored data.

A very simple IT service structure may look like:

```
IT 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 IT 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.

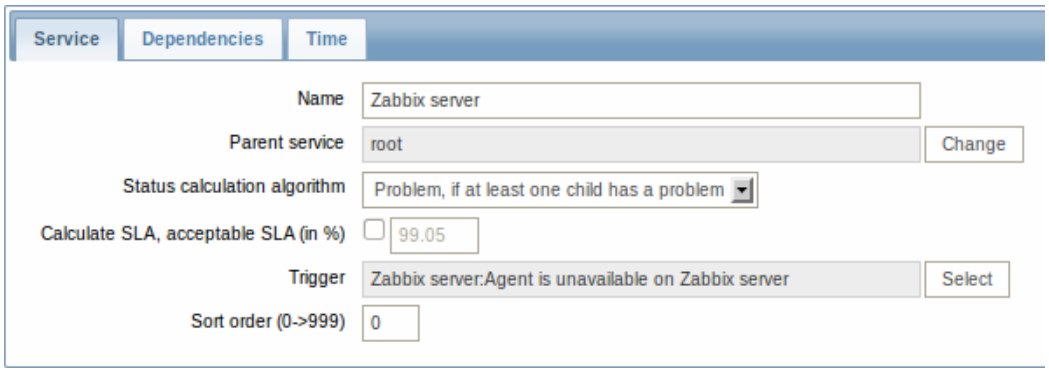
Configuration To configure IT services, go to: *Configuration → IT services*.

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 a service to add services to it or edit the service. A form is displayed where you can edit the service attributes.

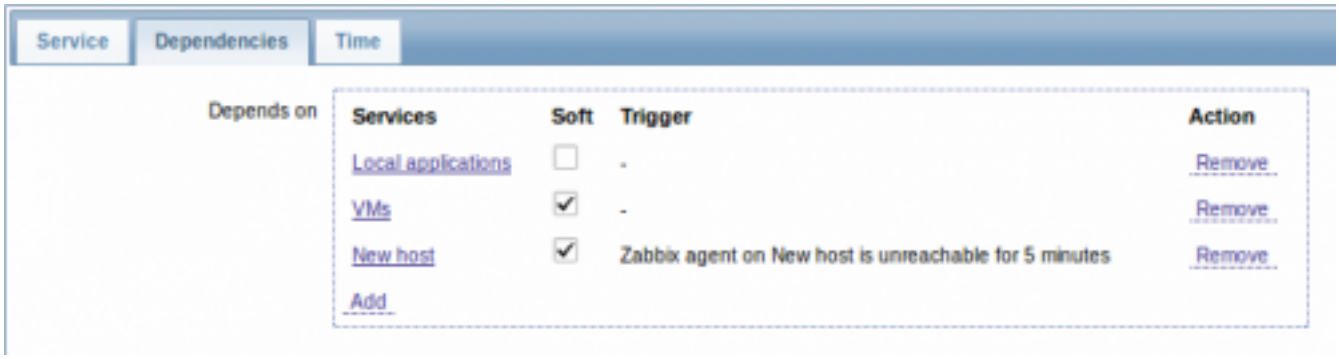
Configuring an IT service

The **Service** tab contains general service attributes:



Parameter	Description
Name	Service name.
Parent service	Parent service the service belongs to.
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.

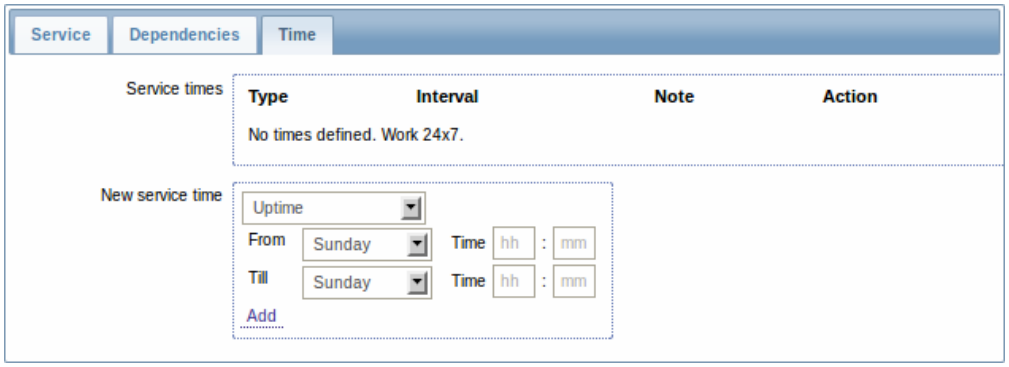


Hard and soft dependency

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 grey 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 IT 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 IT services, go to *Monitoring → IT services*.

8. 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.

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.

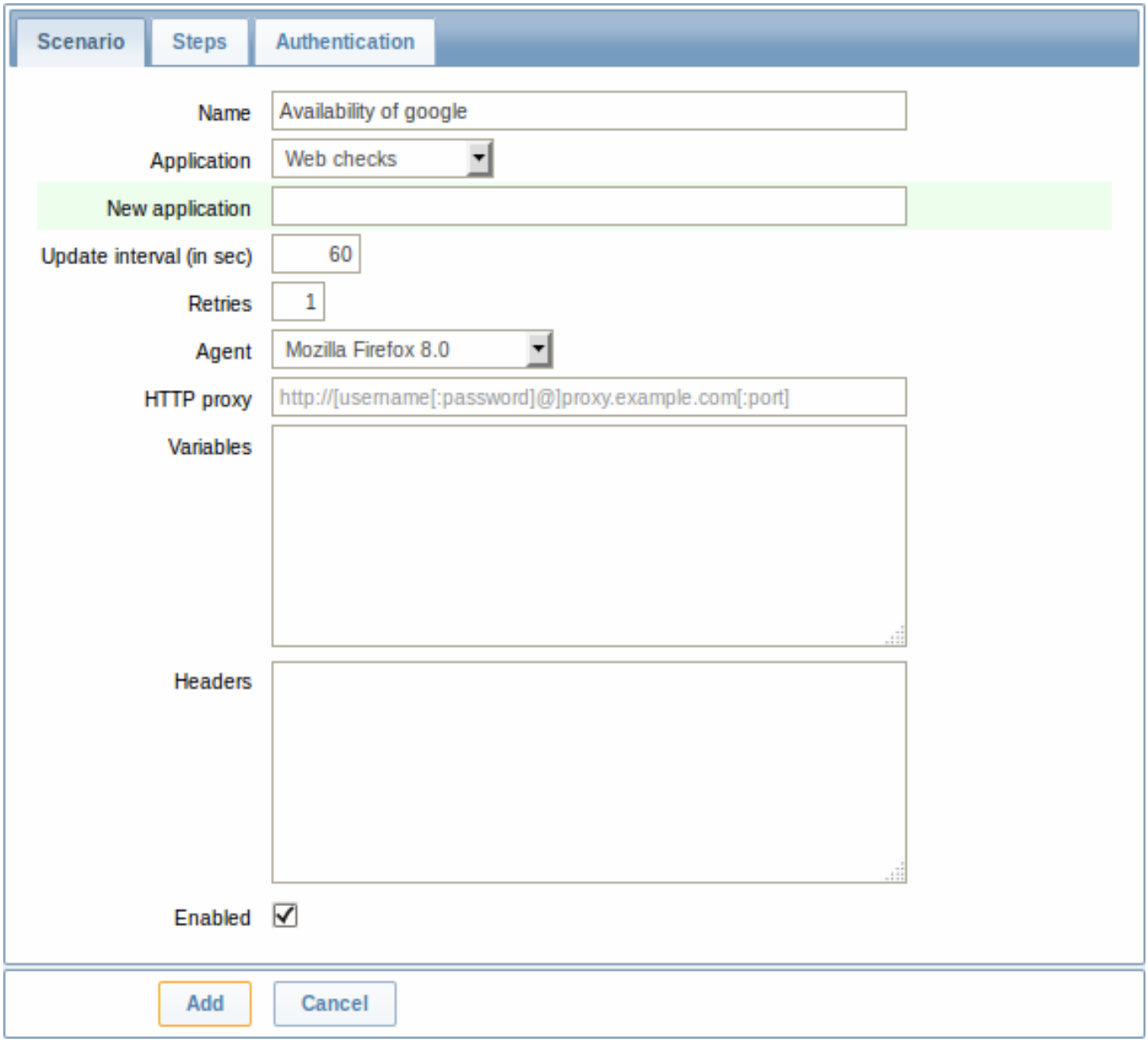
Zabbix web monitoring supports both HTTP and HTTPS. When running a web scenario, Zabbix will optionally follow redirects (see option *Follow redirects* below). Maximum number of redirects is hard-coded to 10 (using CURLOPT_MAXREDIRS cURL option). All cookies are preserved during the execution of a single scenario.

See also known issues for web monitoring using HTTPS protocol.

Configuring a web scenario To configure a web scenario:

- Go to: *Configuration* → *Hosts* (or *Templates*)
- Click on *Web* in the row of the host/template
- Click on *Create scenario* to the right (or on the scenario name to edit an existing scenario)
- Enter parameters of the scenario in the form

The **Scenario** tab allows you to configure the general parameters of a web scenario.



General parameters:

Parameter	Description
Host	Name of the host/template that the scenario belongs to.
Name	Unique scenario name. *Starting with Zabbix 2.2*, the name may contain supported macros.
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 (in sec)	How often the scenario will be executed, in seconds.
Retries	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 retries 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*.

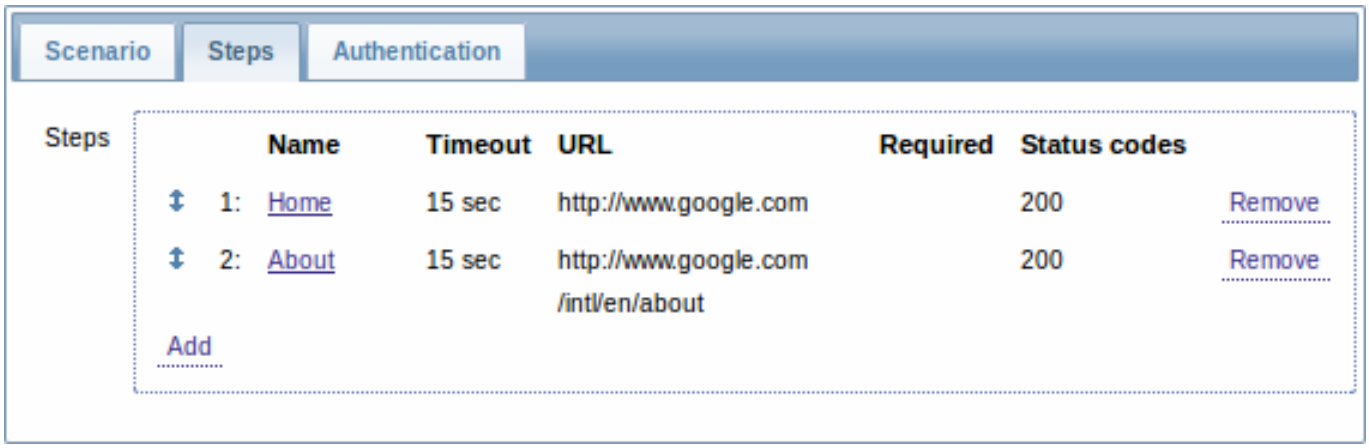
Parameter	Description
HTTP proxy	You can specify an HTTP proxy to use, using the format: *http://[username[:password]@]proxy.mycompany.com[:port]* By default, 1080 port will be used. If specified, the proxy will overwrite proxy related environment variables like http_proxy, HTTPS_PROXY. If not specified, 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. With no protocol specified, the proxy will be treated as an HTTP proxy. *Note*: 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	List of scenario-level variables (macros) 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]+) If the value part starts with *regex:* then the part after it will be treated as a regular expression that will search the web page and, if found, store the match in the variable. Note that at least one subgroup must be present so that the matched value can be extracted. The macros can then be referenced in the steps as {username}, {password} and {hostid}. Zabbix will automatically replace them with actual values. Having variables that search a webpage for a regular expression match is supported *starting with Zabbix 2.2*. `HOST.*` macros and user macros can be used in this field, *starting with Zabbix 2.2*. *Note*: Variables are not URL-encoded.
Headers	HTTP headers that will be sent when performing a request. Headers should be listed using the same syntax as they would appear in the HTTP protocol, optionally using some additional features supported by the CURLOPT_HTTPHEADER cURL option. For example: Accept-Charset: utf-8 Accept-Language: en-US Content-Type: application/xml; charset=utf-8 `HOST.*` macros and user macros can be used in this field. Specifying custom headers is supported *starting with Zabbix 2.4*.
Enabled	The scenario is active if this box is checked, otherwise - disabled.

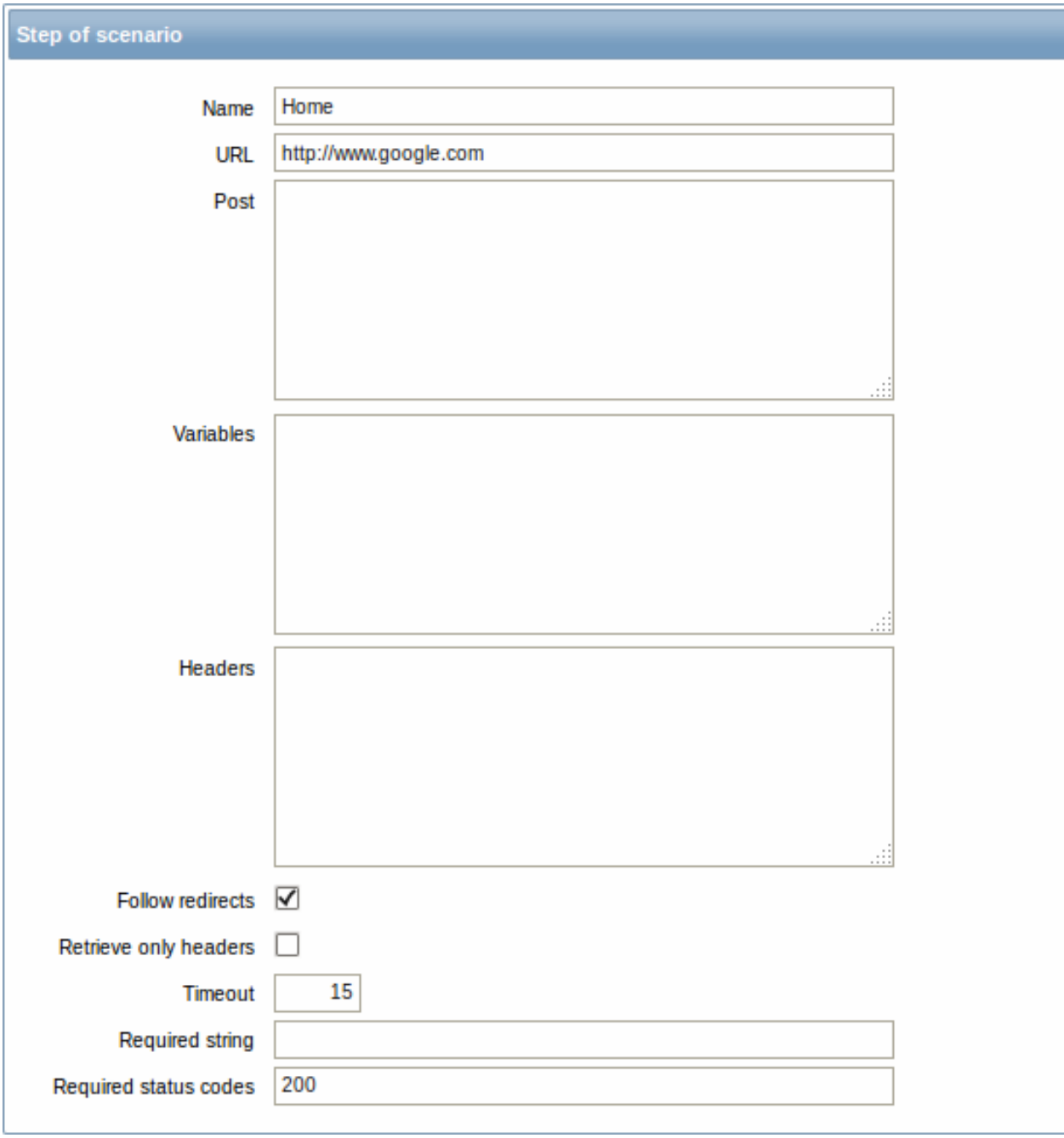
Note that when editing an existing scenario, two extra buttons are available in the form:

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*.





Configuring steps

Step parameters:

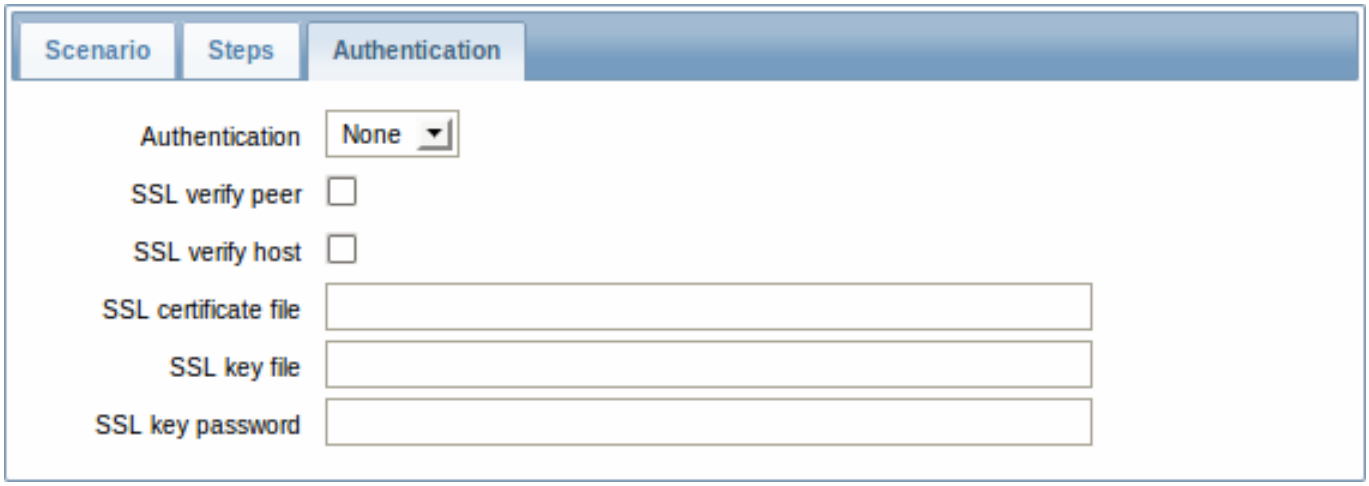
Parameter	Description
Name	Unique step name. *Starting with Zabbix 2.2*, the name may contain supported macros.
URL	URL to connect to and retrieve data. For example: http://www.zabbix.com https://www.google.com GET variables can be passed in the URL parameter. *Starting with Zabbix 2.2*, this field may contain supported macros. Limited to 2048 characters *starting with Zabbix 2.4*.
Post	HTTP POST variables, if any. For example: id=2345&userid={user} If {user} is defined as a macro of the web scenario, it will be replaced by its value when the step is executed. The information will be sent as is, variables are not URL-encoded. *Starting with Zabbix 2.2*, this field may contain supported macros.

Parameter	Description
Variables	List of step-level variables (macros) that may be used for GET and POST functions. 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 *starting with Zabbix 2.2*. *Note*: Variables are not URL-encoded.
Headers	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. Headers on the step level will overwrite the headers specified for the scenario. For example, 'User-Agent:' with no data will remove User-Agent set on scenario level. `HOST.*` macros and user macros can be used in this field. 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 only headers	Mark the checkbox to retrieve only headers from the HTTP response. This sets the CURLOPT_NOBODY cURL option. This option is supported *starting with Zabbix 2.4*.
Timeout	Zabbix will not spend more than the set amount of seconds on processing the URL. Actually this parameter defines 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. For example: 15
Required string	Required regular expressions pattern. Unless retrieved content (HTML) matches required pattern the step will fail. If empty, no check is performed. For example: Homepage of Zabbix Welcome.*admin *Note*: Referencing regular expressions created in the Zabbix frontend is not supported in this field. *Starting with Zabbix 2.2*, this field may contain supported macros.
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 is performed. For example: 200,201,210-299 *Starting with Zabbix 2.2*, user macros can be used in this field.

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 authentication** - basic authentication is used. **NTLM authentication** - NTLM (Windows NT LAN Manager) authentication is used. Selecting an authentication method will provide two additional fields for entering a user name and password. User macros can be used in user and password fields, *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 extention *.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 detailed data of defined web scenarios, go to *Monitoring* → *Web* or *Latest data*. Click on the scenario name to see more detailed statistics.

An overview of web monitoring scenarios can be viewed in *Monitoring* → *Dashboard*.

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 only be set using a runtime control option (`-R log_level_increase="http poller,N"`, where N is the process number). Providing debug level 4 is already set, extended monitoring can be started as demonstrated in the following examples:

```
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.

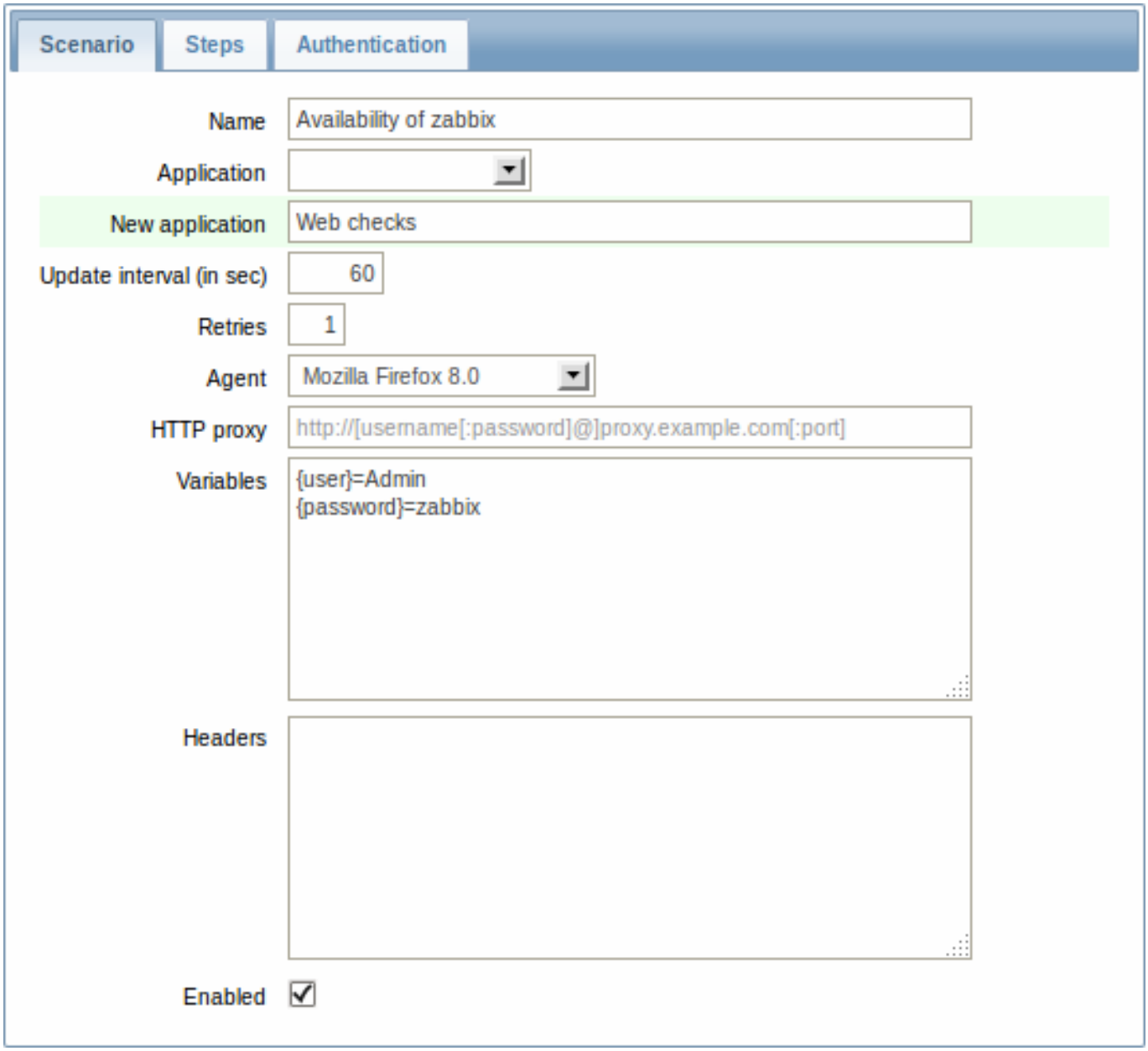
Scenario

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 scenario*.



In the new scenario form we will name the scenario as *Availability of zabbix* and create a new *Web checks* application for it.

Note that we will also create two macros, {user} and {password}.

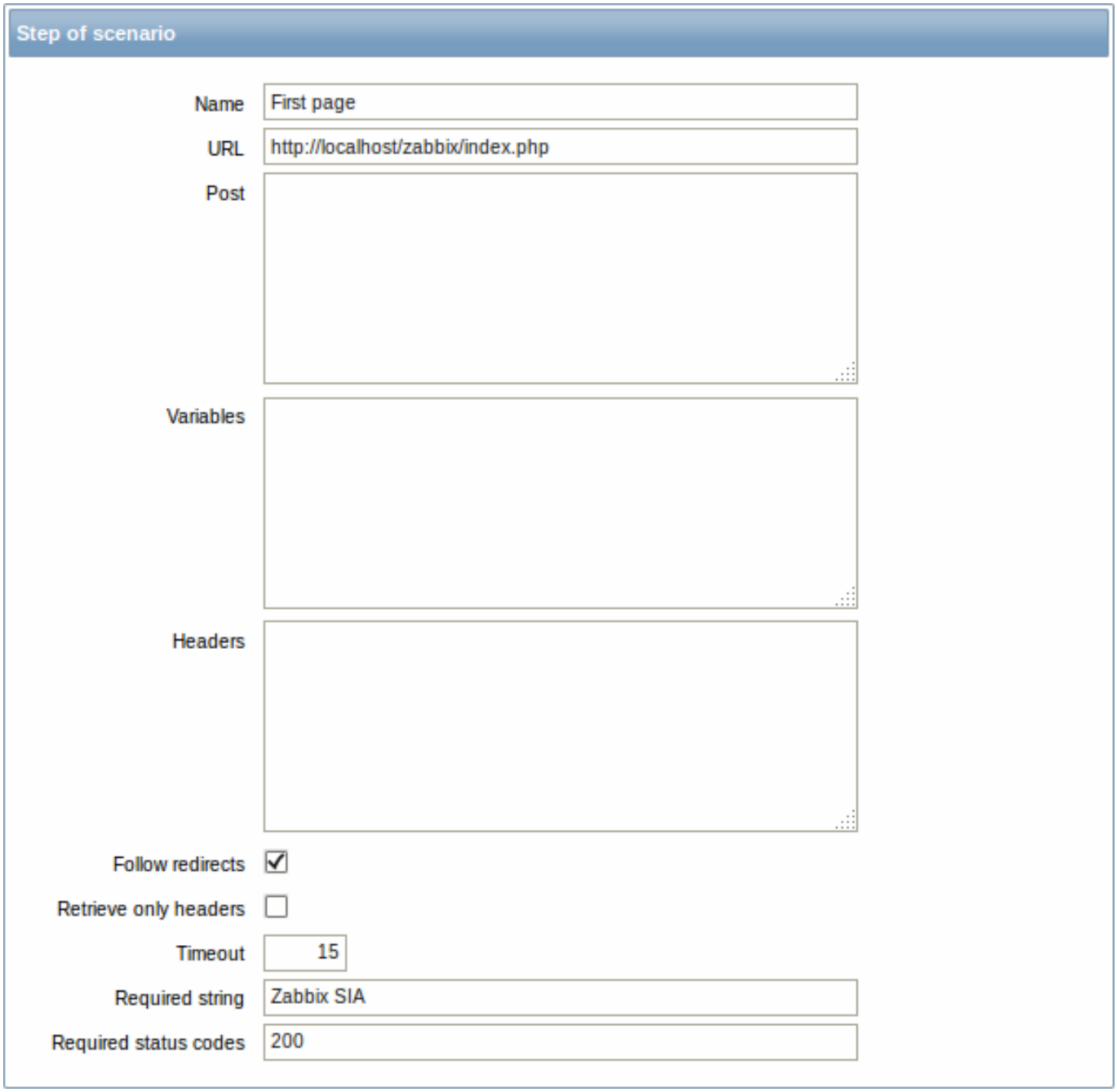
Step 2

Define steps for the scenario.

Click on *Add* button in the *Steps* tab to add individual steps.

Web scenario step 1

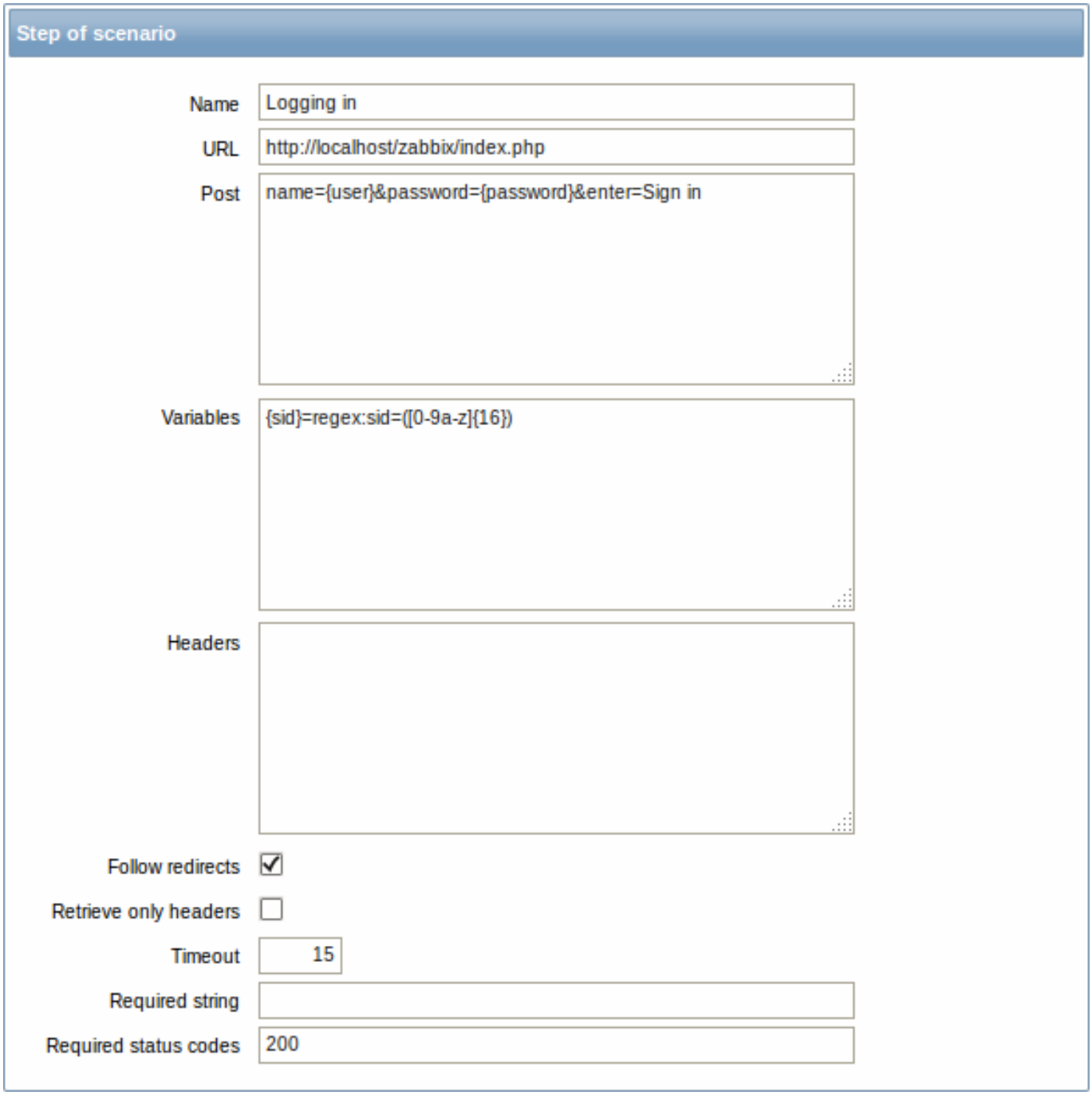
We start by checking that the first page responds correctly, returns with HTTP response code 200 and contains text ”Zabbix SIA”.



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.

All the post variables must be on a single line and concatenated with **&** symbol. Example string for logging into Zabbix frontend:

```
name=Admin&password=zabbix&enter=Sign in
```

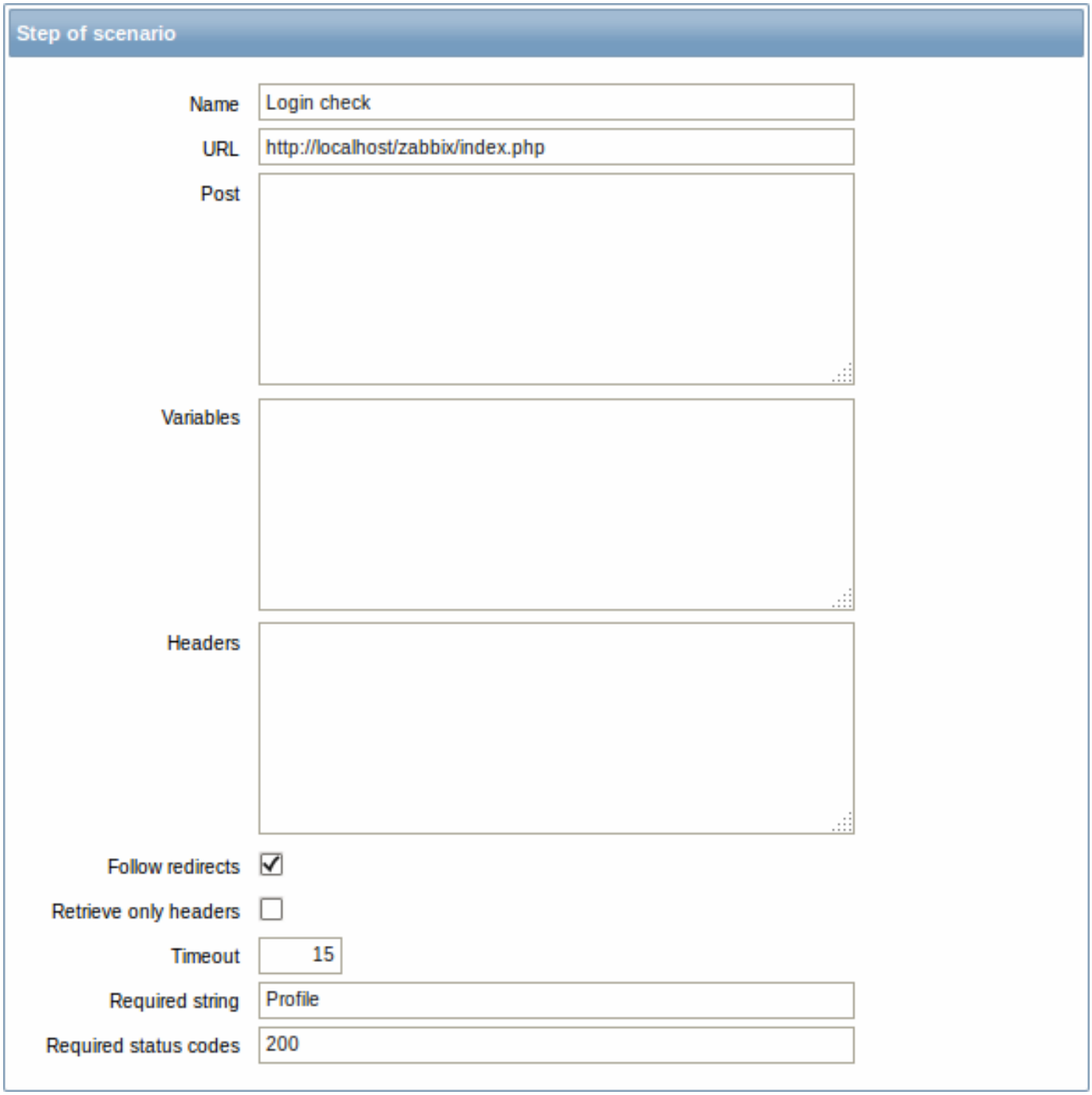
If using the macros as in this example, login string becomes:

```
name={user}&password={password}&enter=Sign in
```

Take note also of how we are getting the content of `{sid}` variable (session ID), which will be required in step 4.

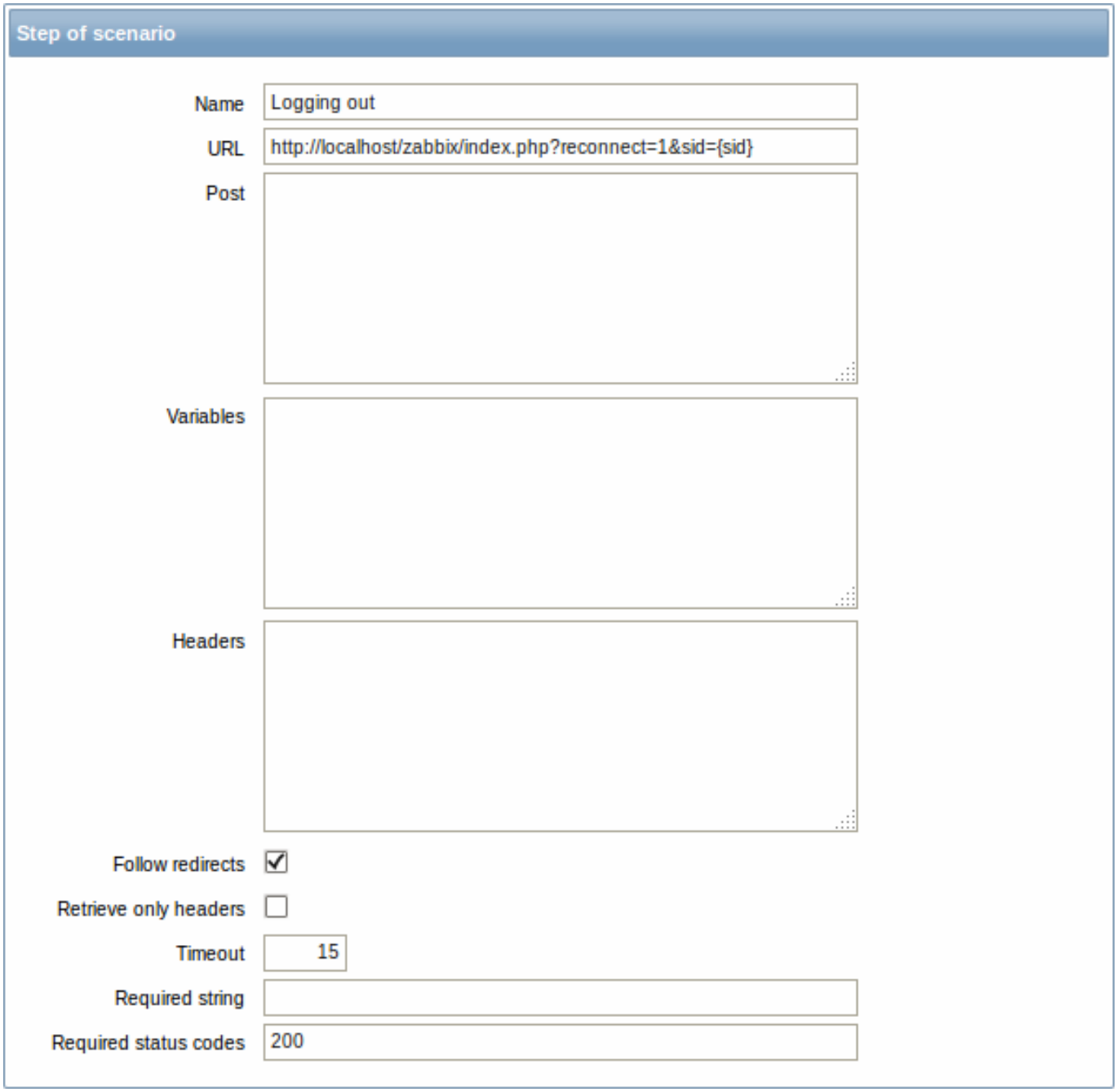
Web scenario step 3

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, **Profile** link appears in the upper right corner.



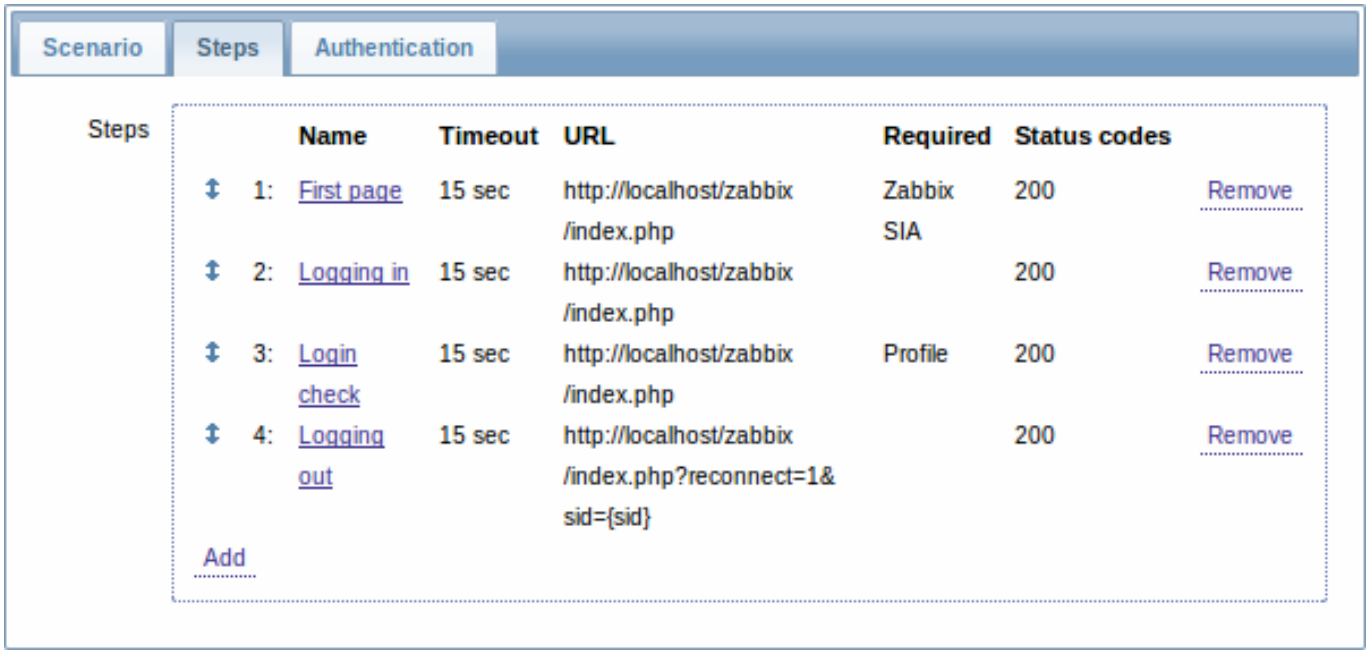
Web scenario step 4

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.



Complete configuration of steps

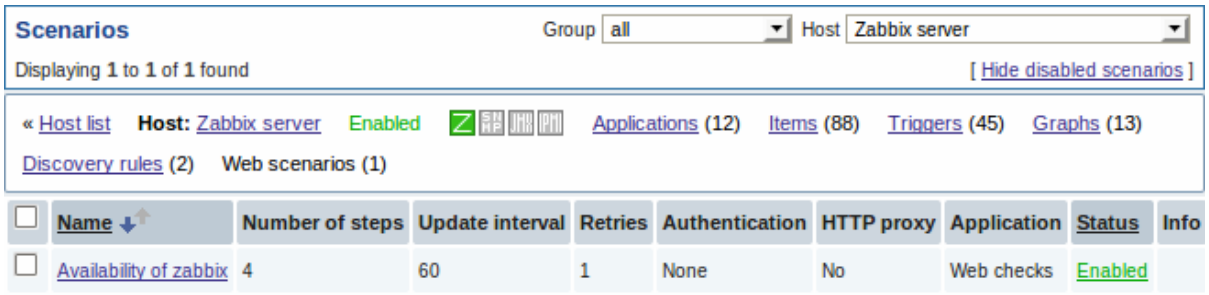
A complete configuration of web scenario steps should look like this:



Step 3

Save the finished web monitoring scenario.

The scenario will appear in *Monitoring* → *Web*:



Click on the scenario name to see more detailed statistics:

9. 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 4.1.

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.

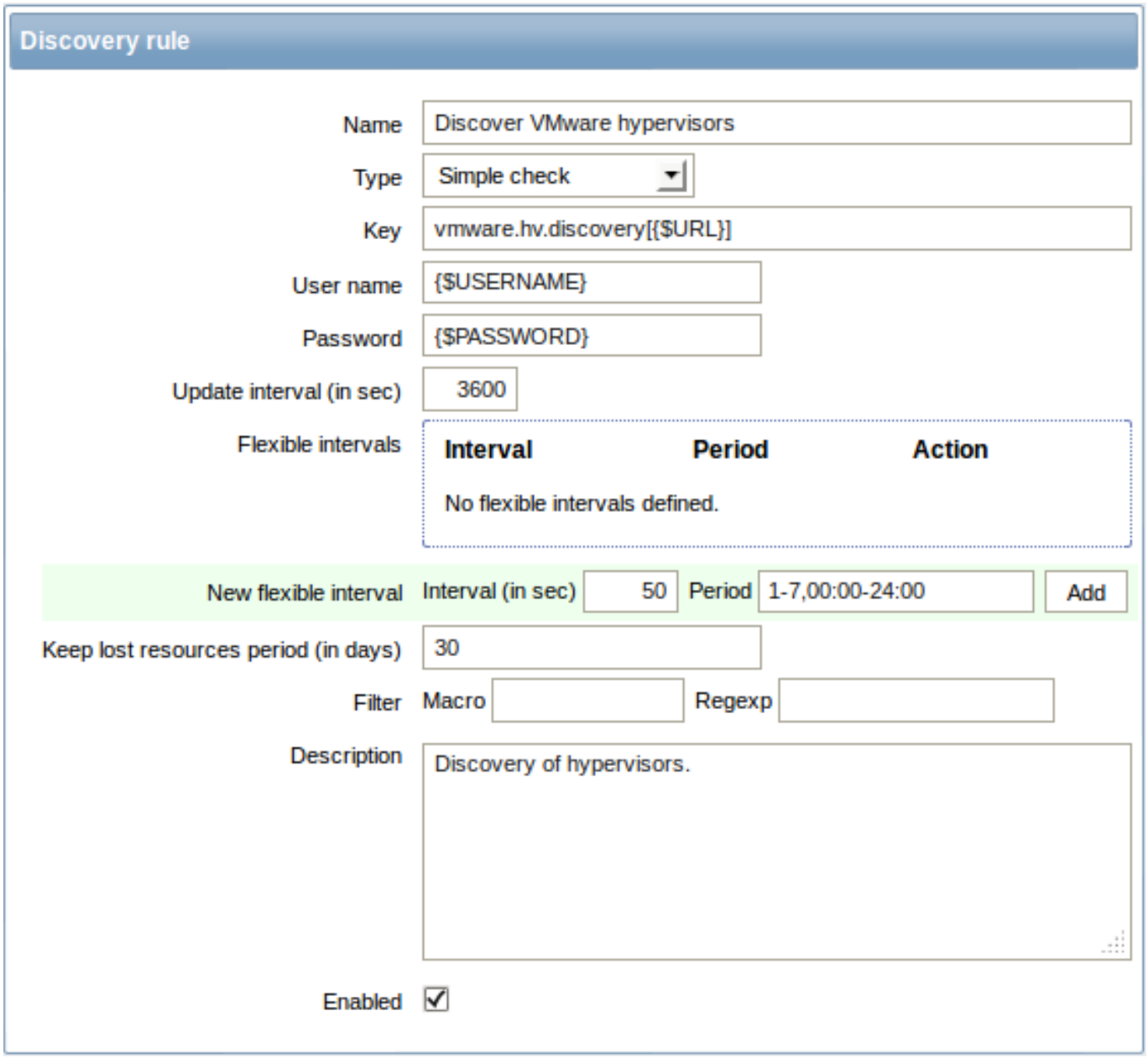
Configuration For virtual machine monitoring to work, Zabbix should be compiled with the --with-libxml2 and --with-libcurl compilation options.

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**

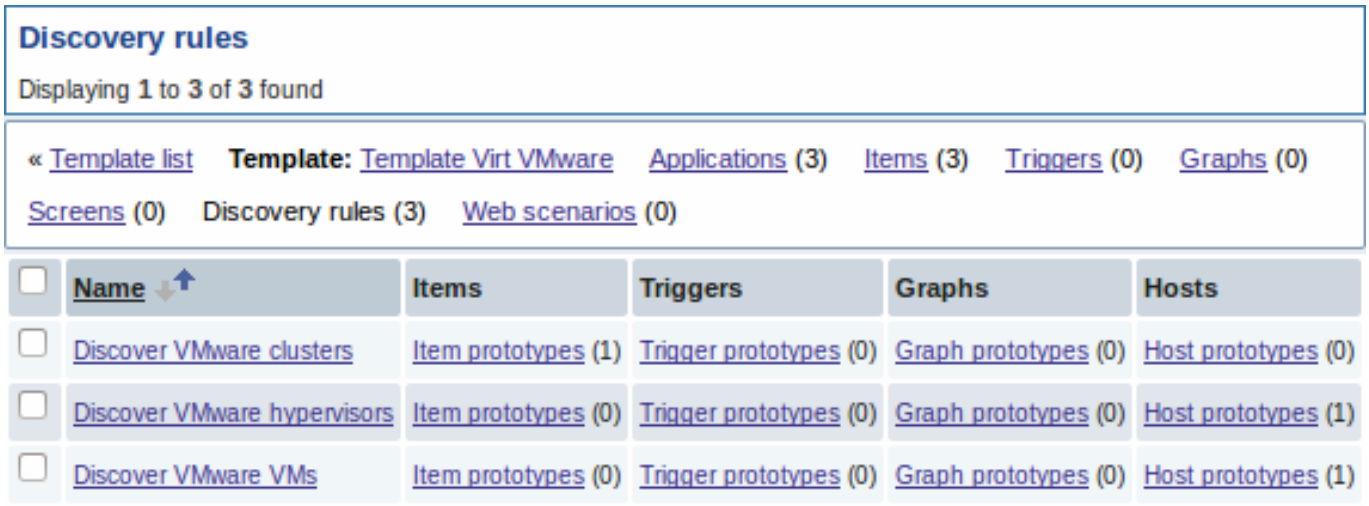
For more details, see the configuration file pages for Zabbix server and proxy.

Discovery Zabbix can use a low-level discovery rule to automatically discover VMware hypervisors and virtual machines.

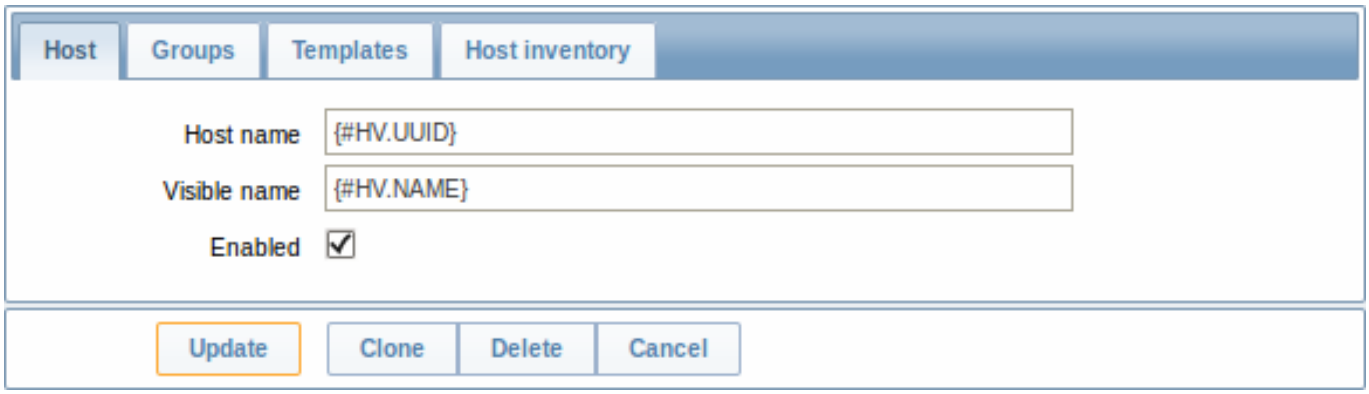


Discovery rule key in the above screenshot is *vmware.hv.discovery[{$URL}]*.

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. Host status, linkage to existing host groups and template linkage are other options that can be set.

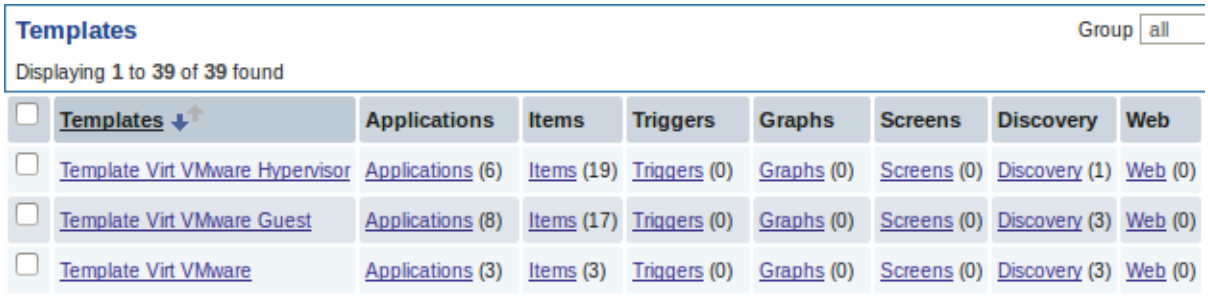


Discovered hosts are prefixed with the name of the discovery rule that created them, in the host list. Discovered hosts can be manually deleted. Discovered hosts will also be automatically deleted, based on the *Keep lost resources period (in days)* value of the discovery rule. Most of the configuration options are read-only, except for enabling/disabling the host and host inventory. Discovered hosts cannot have host prototypes of their own.

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 "*Template Virt VMware*" template should be used for VMware vCenter and ESX hypervisor monitoring. The "*Template Virt VMware Hypervisor*" and "*Template Virt 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 or manually) before importing the templates.

Host configuration To use VMware simple checks the host must have the following user macros defined:

- **{$URL}** - VMware service (vCenter or ESX hypervisor) SDK URL (https://servername/sdk)
- **{$USERNAME}** - VMware service user name
- **{$PASSWORD}** - VMware service {$USERNAME} user password

Example The following example demonstrates how to quickly setup VMware monitoring on Zabbix:

- compile zabbix server with required options (--with-libxml2 and --with-libcurl)
- set the StartVMwareCollectors option in Zabbix server configuration file to 1 or more
- create a new host
- set the host macros required for VMware authentication:

```
    {{..:..:assets:en:manual:vm_monitoring:vm_host_macros.png|}}
* Link the host to the VMware service template:
    {{..:..:assets:en:manual:vm_monitoring:vm_host_templates.png|}}
* Click on the //Add// button to save the host
```

Extended logging The data gathered by VMware collector can be logged for detailed debugging using debug level 5. This level can only be set using a runtime control option (`-R log_level_increase="vmware collector,N"`, where N is a process number). Providing debug level 4 is already set, extended logging can be started as demonstrated in the following examples:

```
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.

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 identifier.
	{#CLUSTER.NAME}	Cluster name.
vmware.hv.discovery Performs hypervisor discovery.	{#HV.UUID}	Unique hypervisor identifier.

Item key		
	{#HV.ID}	Hypervisor identifier (Host-System managed object name).
	{#HV.NAME}	Hypervisor name.
	{#CLUSTER.NAME}	Cluster name, might be empty.
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.
vmware.vm.net.if.discovery Performs virtual machine network interface discovery.	{#IFNAME}	Network interface name.
vmware.vm.vfs.dev.discovery 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.

10. Maintenance

Overview You can define maintenance periods for hosts and host groups in Zabbix. There are two maintenance types - with data collection and with no data collection.

During a maintenance "with data collection" triggers are processed as usual and events are created when required. To skip receiving notifications during such maintenance type, actions should be configured by retaining the default action condition 'Maintenance status = not in "maintenance"' - then you should not get notifications during maintenance. It's dedicated to skip problem notifications.
If a trigger generated an event during the maintenance period (as set in maintenance configuration), an additional event (the same as last event created during maintenance) will be created at the end of maintenance for the host. This way, if a problem happens during maintenance and is not resolved, a notification may be generated after the maintenance period ends.

To receive a notification during the maintenance you have to remove the default action condition about not taking actions during maintenance.

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. A proxy will always collect data regardless of the maintenance type (including "no data" maintenance). The data is later ignored by the server if 'no data collection' is set.

When "no data" maintenance ends, triggers using nodata() function will not fire before the next check during the period they are checking.

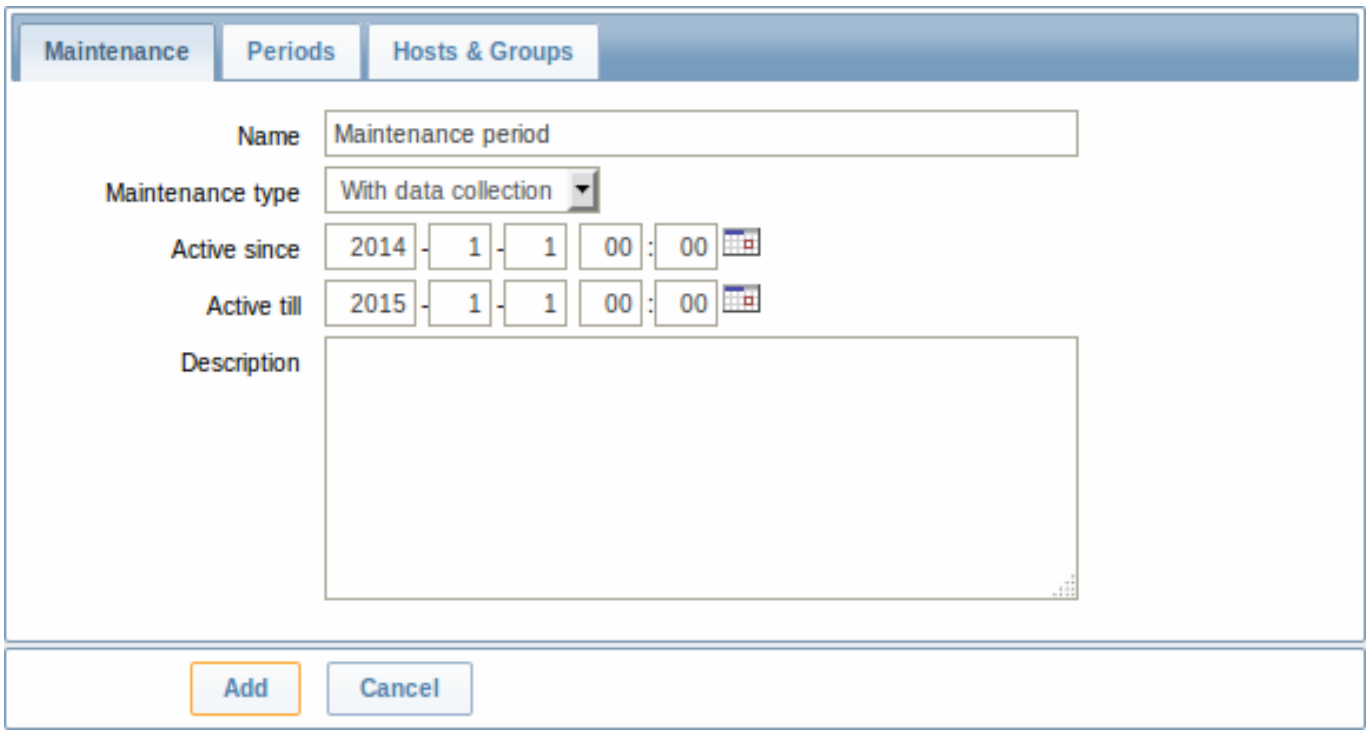
If a log item is added while a host is in maintenance and the maintenance ends, only new logfile entries since the end of the maintenance will be gathered.

Attention:
To ensure predictable behaviour of recurring maintenance periods (daily, weekly, monthly), it is required to use a common timezone for all parts of Zabbix.

Configuration To configure a maintenance period:

- Go to: *Configuration* → *Maintenance*
- Click on *Create maintenance period* (or on the name of an existing maintenance period)

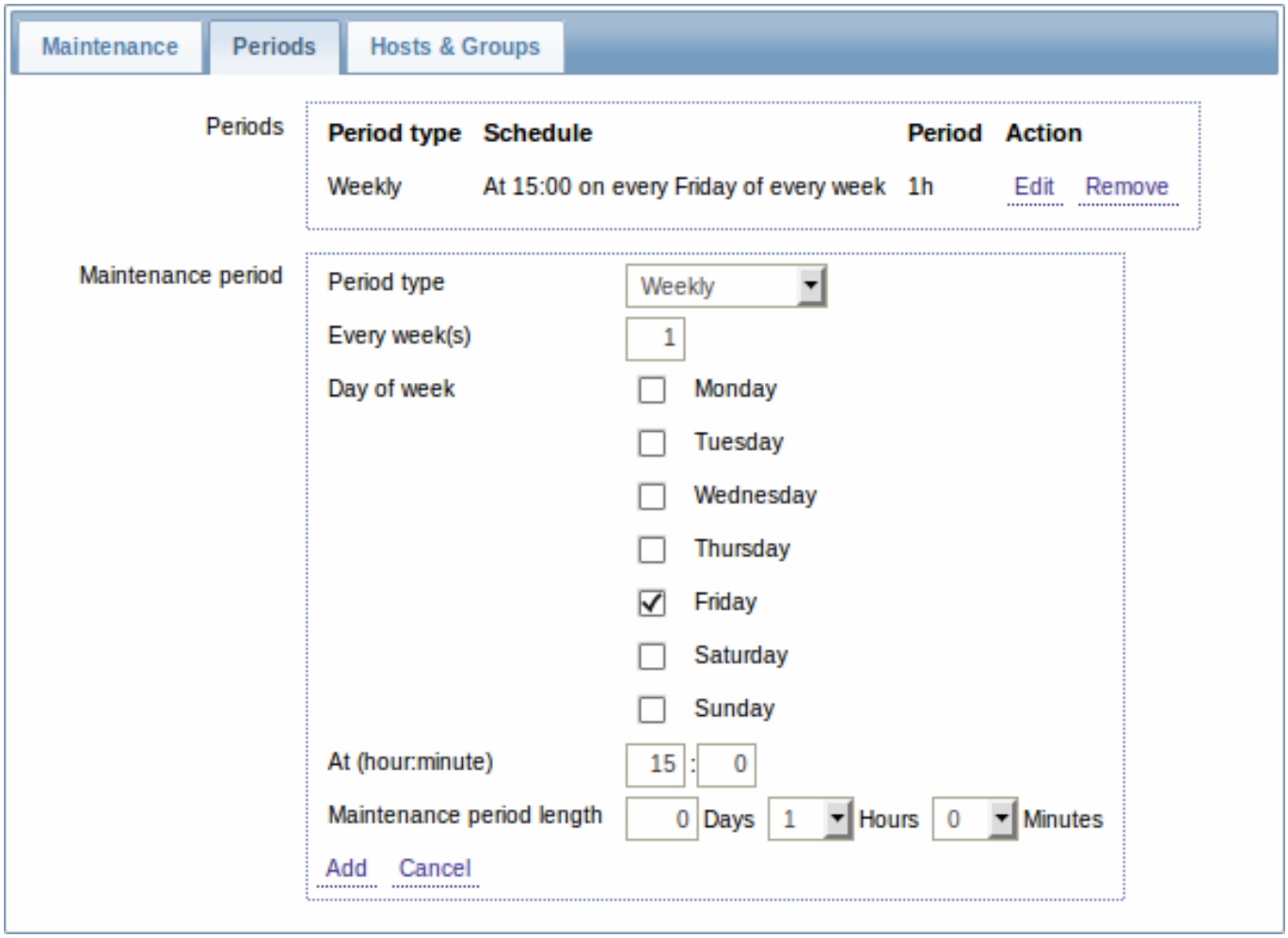
The **Maintenance** tab contains general maintenance period attributes:



Parameter	Description
Name	Name of the maintenance period.

Parameter	Description
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.

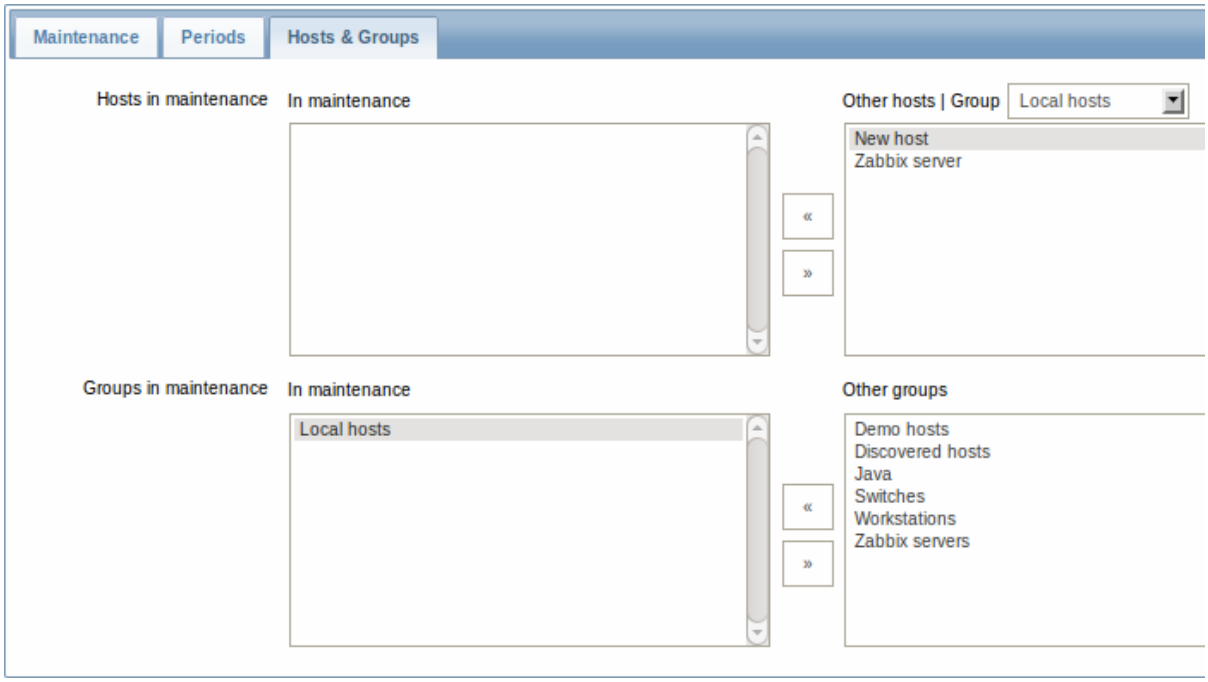
The **Periods** tab allows you to define the exact days and hours when the maintenance takes place. Clicking on *New* opens a flexible *Maintenance period* form where you can define the times - for daily, weekly, monthly or one-time maintenance.



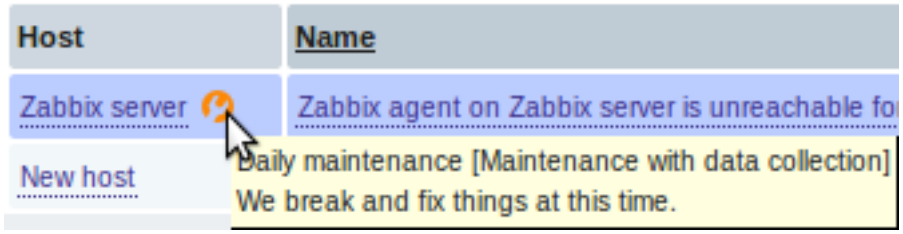
Daily and weekly periods have an *Every day/Every week* parameter, which defaults to 1. Setting it to 2 would make the maintenance take place every two days or every two weeks and so on. The starting day or week is the day or week that *Active since* time falls on.

For example, having *Active since* set to 2013-09-06 12:00 and an hour long daily recurrent period every two days at 23:00 will result in the first maintenance period starting on 2013-09-06 at 23:00, while the second maintenance period will start on 2013-09-08 at 23:00. Or, with the same *Active since* time and an hour long daily recurrent period every two days at 01:00, the first maintenance period will start on 2013-09-08 at 01:00, and the second maintenance period on 2013-09-10 at 01:00.

The **Hosts & Groups** tab allows you to select the hosts and host groups for maintenance.



Display A round orange icon with a white wrench indicates that a host is in maintenance in the *Monitoring* → *Dashboard*, *Monitoring* → *Triggers* and *Inventory* → *Hosts* → *Host inventory details* sections.



Maintenance details are displayed when the mouse pointer is positioned over the icon.

Note:
The display of hosts in maintenance in the Dashboard can be unset altogether with the dashboard filtering function.

Additionally, hosts in maintenance get an orange background in *Monitoring* → *Maps* and in *Configuration* → *Hosts* their status is displayed as 'In maintenance'.

11. Regular expressions

Overview POSIX extended regular expressions 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.

Global regular expressions There is an advanced editor for creating and testing complex regular expressions in Zabbix frontend.

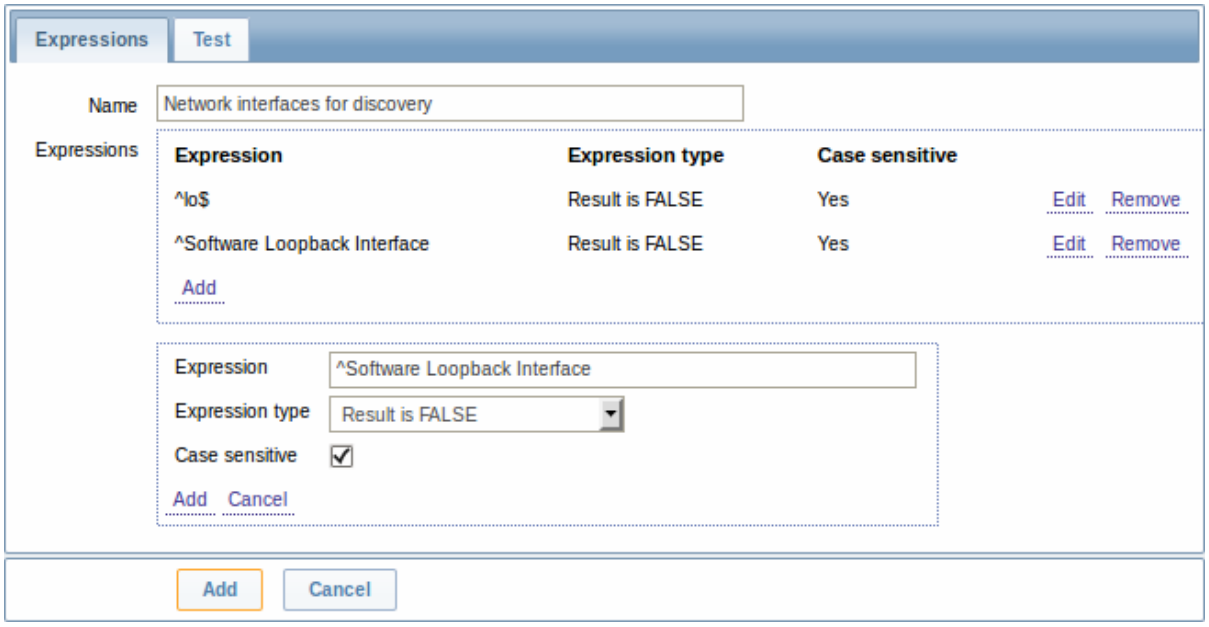
Once a regular expression has been created this way, it can be used in several places in the frontend by referring to its name, prefixed with @, for example, *@mycustomregexp*.

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.

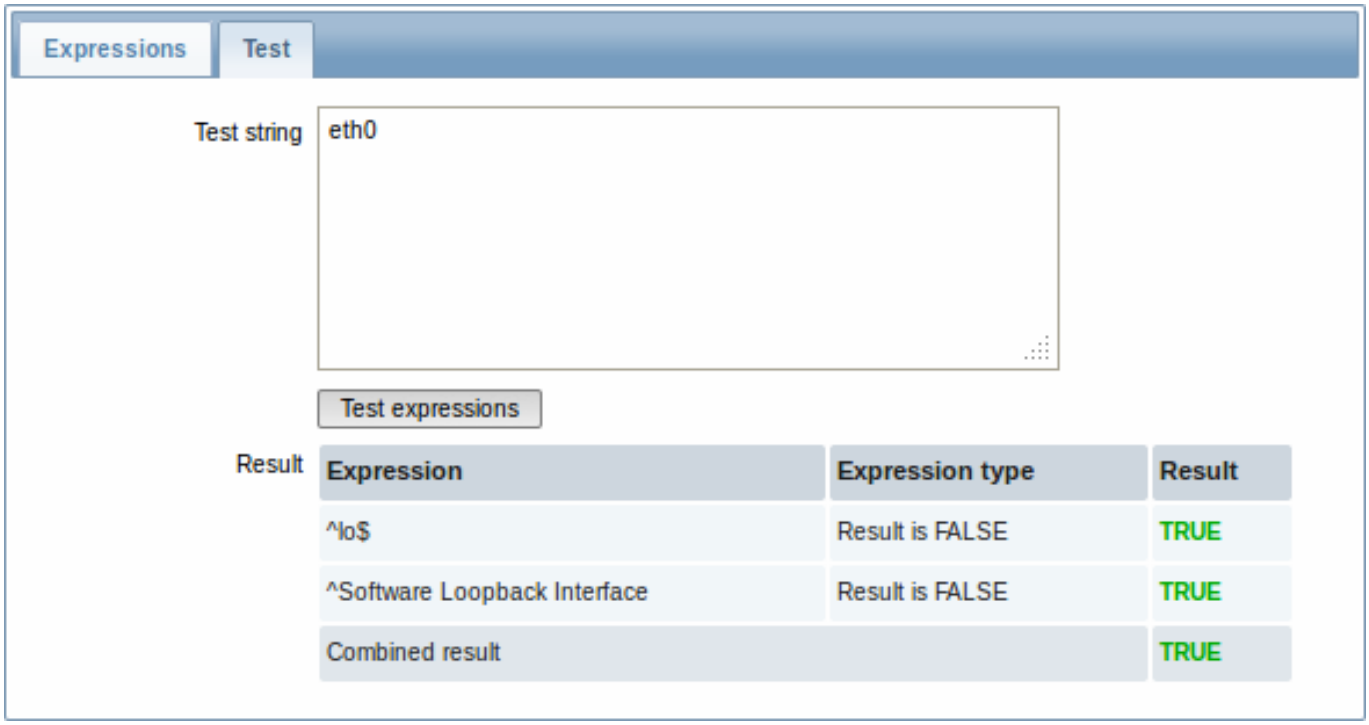


Parameter	Description
Name	Set the regular expression name. Any Unicode characters are allowed.
Expressions	Click on *Add* in the Expressions block to add a new subexpression.

Starting with 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).

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.

12. Event acknowledgment

Overview Problem events in Zabbix can be acknowledged by users.

If a user gets notified about of a problem event, they can go to Zabbix frontend, navigate from events to the acknowledgment screen 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 more coordinated way.

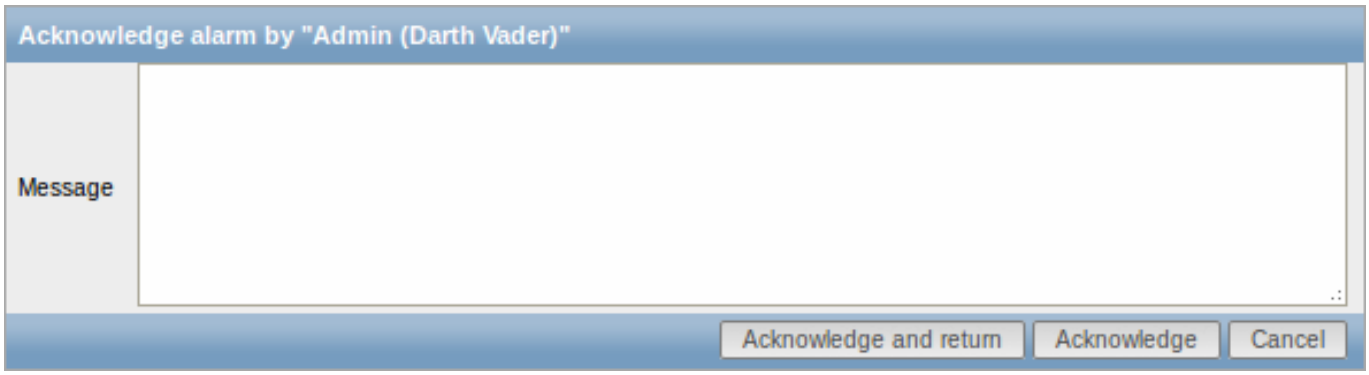
Acknowledgment status is also used when defining action operations. You can define, for example, that a notification is sent to a higher level manager only if an event is not acknowledged for some time.

To acknowledge events, a user must have at least read permission to the corresponding trigger.

Acknowledgment screen The acknowledgment status of problems is displayed in *Monitoring* → *Events*.

The *Ack* column contains either a 'Yes' or a 'No', indicating an acknowledged or an unacknowledged problem respectively. A 'Yes' may also have a number with it in brackets, indicating the number of comments for the problem so far.

Both 'Yes' and 'No' are links. Clicking them will take you to the acknowledgment screen.



To acknowledge a problem, enter your comment and click on *Acknowledge and return* or simply *Acknowledge*. 'Acknowledge and return' will take you back to the event screen.

Any previous comments for the problem are displayed above the comment area.

Display Acknowledgment information is fully displayed in the event details accessible by clicking the time of event in *Monitoring → Events*.

Acknowledgment status is displayed in the *Last 20 issues* block of *Monitoring → Dashboard*.

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 dashboard filter. It is possible to display all problem count, unacknowledged problem count as separated from the total or unacknowledged problem count only.

Acknowledgment status is displayed in *Monitoring → Triggers*. There, acknowledgment status is also used with the trigger filtering options. You can filter by unacknowledged triggers or triggers with the last event unacknowledged.

13. 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:

- sharing of templates or network maps - Zabbix users may share their configuration parameters
- integration 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 (including all directly attached items, triggers, graphs, screens, discovery rules and template linkage)
- hosts (including all directly attached items, triggers, graphs, discovery rules and template linkage)
- network maps (including all related images; map export/import is supported since *Zabbix 1.8.2*)
- images
- screens

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.
- Triggers and graphs that use web items are not 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 too.
- Empty tags for items, triggers, graphs, host/template applications, discoveryRules, itemPrototypes, triggerPrototypes, graphPrototypes are meaningless i.e. it’s the same as if it was missing. Other tags, for example, item applications, are meaningful i.e. empty tag means no applications for item, missing tag means don’t update applications.
- Import supports 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 version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>2.0</version>
    <date>2014-09-18T14:07:36Z</date>
</zabbix_export>
```

XML base format

`<?xml version="1.0" encoding="UTF-8"?>`

Default header for XML documents.

`<zabbix_export>`

Root element for Zabbix XML export.

`<version>2.0</version>`

Export version.

`<date>2014-09-18T14:07:36Z</date>`

Date when export was created in ISO 8601 long format.

Other tags are dependent on exported objects.

Groups

Frontend can export groups only with hosts or templates. When host or template is exported all groups it belongs to are exported with it automatically.

API allows to export groups independently from hosts or templates.

```
<groups>
    <group>
        <name>Zabbix servers</name>
    </group>
</groups>
```

groups/group

Parameter	Type	Description	Details
name	*string*	Group name.	

Hosts

Hosts are exported with many related objects and object relations.

Host export contains:

- host data
- host inventory data
- group relations
- template relations
- interfaces
- macros
- applications
- items
- discovery rules with all prototypes

When a host is imported and updated, it can only be linked to additional templates and never be unlinked from any.

```
<hosts>
        <host>
            <host>Zabbix server</host>
            <name>Zabbix server</name>
            <description>Zabbix monitoring server.</description>
            <proxy/>
            <status>0</status>
            <ipmi_authtype>-1</ipmi_authtype>
            <ipmi_privilege>2</ipmi_privilege>
            <ipmi_username/>
            <ipmi_password/>
            <templates/>
            <groups>
                <group>
                    <name>Zabbix servers</name>
                </group>
            </groups>
            <interfaces>
                <interface>
                    <default>1</default>
                    <type>1</type>
                    <useip>1</useip>
                    <ip>127.0.0.1</ip>
                    <dns/>
                    <port>20001</port>
                    <interface_ref>if1</interface_ref>
                </interface>
            </interfaces>
            <applications>
                <application>
                    <name>Memory</name>
                </application>
                <application>
                    <name>Zabbix agent</name>
                </application>
            </applications>
            <items>
                <item>
                    <name>Agent ping</name>
                    <type>0</type>
                    <snmp_community/>
                    <multiplier>0</multiplier>
                    <snmp_oid/>
                    <key>agent.ping</key>
                    <delay>60</delay>
                    <history>7</history>
                    <trends>365</trends>
                    <status>0</status>
                    <value_type>3</value_type>
                    <allowed_hosts/>
                    <units/>
                    <delta>0</delta>
                    <snmpv3_securityname/>
                    <snmpv3_securitylevel>0</snmpv3_securitylevel>
                    <snmpv3_authpassphrase/>
                    <snmpv3_privpassphrase/>
                    <formula>1</formula>
                    <delay_flex/>
                    <params/>
                    <ipmi_sensor/>
                    <data_type>0</data_type>
                    <authtype>0</authtype>
```

```
                <username/>
                <password/>
                <publickey/>
                <privatekey/>
                <port/>
                <description>The agent always returns 1 for this item. It could be used in combination
                <inventory_link>0</inventory_link>
                <applications>
                    <application>
                        <name>Zabbix agent</name>
                    </application>
                </applications>
                <valuemap>
                    <name>Zabbix agent ping status</name>
                </valuemap>
                <interface_ref>if1</interface_ref>
            </item>
            <item>
                <name>Available memory</name>
                <type>0</type>
                <snmp_community/>
                <multiplier>0</multiplier>
                <snmp_oid/>
                <key>vm.memory.size[available]</key>
                <delay>60</delay>
                <history>7</history>
                <trends>365</trends>
                <status>0</status>
                <value_type>3</value_type>
                <allowed_hosts/>
                <units>B</units>
                <delta>0</delta>
                <snmpv3_securityname/>
                <snmpv3_securitylevel>0</snmpv3_securitylevel>
                <snmpv3_authpassphrase/>
                <snmpv3_privpassphrase/>
                <formula>1</formula>
                <delay_flex/>
                <params/>
                <ipmi_sensor/>
                <data_type>0</data_type>
                <authtype>0</authtype>
                <username/>
                <password/>
                <publickey/>
                <privatekey/>
                <port/>
                <description>Available memory is defined as free+cached+buffers memory.</description>
                <inventory_link>0</inventory_link>
                <applications>
                    <application>
                        <name>Memory</name>
                    </application>
                </applications>
                <valuemap/>
                <interface_ref>if1</interface_ref>
            </item>
        </items>
        <discovery_rules>
            <discovery_rule>
                <name>Mounted filesystem discovery</name>
                <type>0</type>
```

```
                    <snmp_community/>
                    <snmp_oid/>
                    <key>vfs.fs.discovery</key>
                    <delay>3600</delay>
                    <status>0</status>
                    <allowed_hosts/>
                    <snmpv3_securityname/>
                    <snmpv3_securitylevel>0</snmpv3_securitylevel>
                    <snmpv3_authpassphrase/>
                    <snmpv3_privpassphrase/>
                    <delay_flex/>
                    <params/>
                    <ipmi_sensor/>
                    <authtype>0</authtype>
                    <username/>
                    <password/>
                    <publickey/>
                    <privatekey/>
                    <port/>
                    <filter>{#FSTYPE}:@File systems for discovery</filter>
                    <lifetime>30</lifetime>
                    <description>Discovery of file systems of different types as defined in global regular
                    <item_prototypes>
                        <item_prototype>
                            <name>Free disk space on $1</name>
                            <type>0</type>
                            <snmp_community/>
                            <multiplier>0</multiplier>
                            <snmp_oid/>
                            <key>vfs.fs.size[{#FSNAME},free]</key>
                            <delay>60</delay>
                            <history>7</history>
                            <trends>365</trends>
                            <status>0</status>
                            <value_type>3</value_type>
                            <allowed_hosts/>
                            <units>B</units>
                            <delta>0</delta>
                            <snmpv3_securityname/>
                            <snmpv3_securitylevel>0</snmpv3_securitylevel>
                            <snmpv3_authpassphrase/>
                            <snmpv3_privpassphrase/>
                            <formula>1</formula>
                            <delay_flex/>
                            <params/>
                            <ipmi_sensor/>
                            <data_type>0</data_type>
                            <authtype>0</authtype>
                            <username/>
                            <password/>
                            <publickey/>
                            <privatekey/>
                            <port/>
                            <description/>
                            <inventory_link>0</inventory_link>
                            <applications>
                                <application>
                                    <name>Filesystems</name>
                                </application>
                            </applications>
                            <valuemap/>
                            <interface_ref>if1</interface_ref>
```

```
                    </item_prototype>
                </item_prototypes>
                <trigger_prototypes>
                    <trigger_prototype>
                        <expression>{Zabbix server 2:vfs.fs.size[{#FSNAME},pfree].last()}<20</expressi
                        <name>Free disk space is less than 20% on volume {#FSNAME}</name>
                        <url/>
                        <status>0</status>
                        <priority>2</priority>
                        <description/>
                        <type>0</type>
                    </trigger_prototype>
                </trigger_prototypes>
                <graph_prototypes>
                    <graph_prototype>
                        <name>Disk space usage {#FSNAME}</name>
                        <width>600</width>
                        <height>340</height>
                        <yaxismin>0.0000</yaxismin>
                        <yaxismax>0.0000</yaxismax>
                        <show_work_period>0</show_work_period>
                        <show_triggers>0</show_triggers>
                        <type>2</type>
                        <show_legend>1</show_legend>
                        <show_3d>1</show_3d>
                        <percent_left>0.0000</percent_left>
                        <percent_right>0.0000</percent_right>
                        <ymin_type_1>0</ymin_type_1>
                        <ymax_type_1>0</ymax_type_1>
                        <ymin_item_1>0</ymin_item_1>
                        <ymax_item_1>0</ymax_item_1>
                        <graph_items>
                            <graph_item>
                                <sortorder>0</sortorder>
                                <drawtype>0</drawtype>
                                <color>C80000</color>
                                <yaxisside>0</yaxisside>
                                <calc_fnc>2</calc_fnc>
                                <type>2</type>
                                <item>
                                    <host>Zabbix server 2</host>
                                    <key>vfs.fs.size[{#FSNAME},total]</key>
                                </item>
                            </graph_item>
                            <graph_item>
                                <sortorder>1</sortorder>
                                <drawtype>0</drawtype>
                                <color>00C800</color>
                                <yaxisside>0</yaxisside>
                                <calc_fnc>2</calc_fnc>
                                <type>0</type>
                                <item>
                                    <host>Zabbix server 2</host>
                                    <key>vfs.fs.size[{#FSNAME},free]</key>
                                </item>
                            </graph_item>
                        </graph_items>
                    </graph_prototype>
                </graph_prototypes>
                <interface_ref>if1</interface_ref>
            </discovery_rule>
        </discovery_rules>
```

```
            <macros>
                <macro>
                    <macro>{$M1}</macro>
                    <value>m1</value>
                </macro>
                <macro>
                    <macro>{$M2}</macro>
                    <value>m2</value>
                </macro>
            </macros>
            <inventory/>
        </host>
    </hosts>
```

hosts/host

Parameter	Type	Description	Details
host	*string*	Host name.	
name	*string*	Visible host name.	
description	*string*	Host description.	
status	*int*	Host status.	
proxy	*int*	Proxy name.	
ipmi_authtype	*int*	IPMI authentication type.	
ipmi_privilege	*int*	IPMI privilege.	
ipmi_username	*string*	IPMI username.	
ipmi_password	*string*	IPMI password.	

hosts/host/groups/group

Parameter	Type	Description	Details
name	*string*	Group name.	

hosts/host/templates/template

Parameter	Type	Description	Details
name	*string*	Template technical name.	

hosts/host/interfaces/interface

Column name	Type	Description
default	integer	Interface status: 0 - Not default interface 1 - Default interface
type	integer	Interface type: 1 - agent 2 - SNMP 3 - IPMI 4 - JMX
useip	integer	How to connect to the host: 0 – connect to the host using DNS name 1 – connect to the host using IP address
ip	varchar	IP address, can be either IPv4 or IPv6.
dns	varchar	DNS name.
port	varchar	Port number.
interface_ref	varchar	Interface reference name to be used in items.

hosts/host/applications/application

Parameter	Type	Description	Details
name	*string*	Application name.	

hosts/host/items/item

Parameter	Type	Description
type	*int*	Item type: 0 - Zabbix agent 1 - SNMPv1 2 - Trapper 3 - Simple check 4 - SNMPv2 5 - Internal 6 - SNMPv3 7 - Active check 8 - Aggregate 9 - HTTP test (web monitoring scenario step) 10 - External 11 - Database monitor 12 - IPMI 13 - SSH 14 - telnet 15 - Calculated 16 - JMX 17 - SNMP trap
snmp_community	*string*	SNMP Community name
snmp_oid	*string*	SNMP OID
port	*int*	Item custom port
name	*string*	Item name
key	*string*	Item key
delay	*int*	Check interval
history	*int*	How long to keep item history (days)
trends	*int*	How long to keep item trends (days)
status	*int*	Item status
value_type	*int*	Value type
trapper_hosts	*string*	
units	*string*	Value units
multiplier	*int*	Value multiplier
delta	*int*	Store values as delta
snmpv3_securityname	*string*	SNMPv3 security name
snmpv3_securitylevel	*int*	SNMPv3 security level
snmpv3_authpassphrase	*string*	SNMPv3 authentication phrase
snmpv3_privpassphrase	*string*	SNMPv3 private phrase
formula	*string*	
delay_flex	*string*	Flexible delay
params	*string*	
ipmi_sensor	*string*	IPMI sensor
data_type	*int*	
authtype	*int*	
username	*string*	
password	*string*	
publickey	*string*	
privatekey	*string*	
interface_ref	*varchar*	Reference to host interface
description	*string*	Item description
inventory_link	*int*	Host inventory field number, that will be updated with the value returned by the item
applications		Item applications
valuemap		Value map assigned to item

hosts/host/items/item/applications/application

Parameter	Type	Description	Details
name	*string*	Application name.	

14. 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
- 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', consecutively.
Service Lost	The service is 'down' after it was 'up'.
Service Down	The service is 'down', consecutively.
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', consecutively.
Host Lost	All services of a host are 'down' after at least one was 'up'.
Host Down	All services of a host are 'down', consecutively.

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.

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

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.

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.

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

Starting with Zabbix 2.4.0, 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.
- 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.

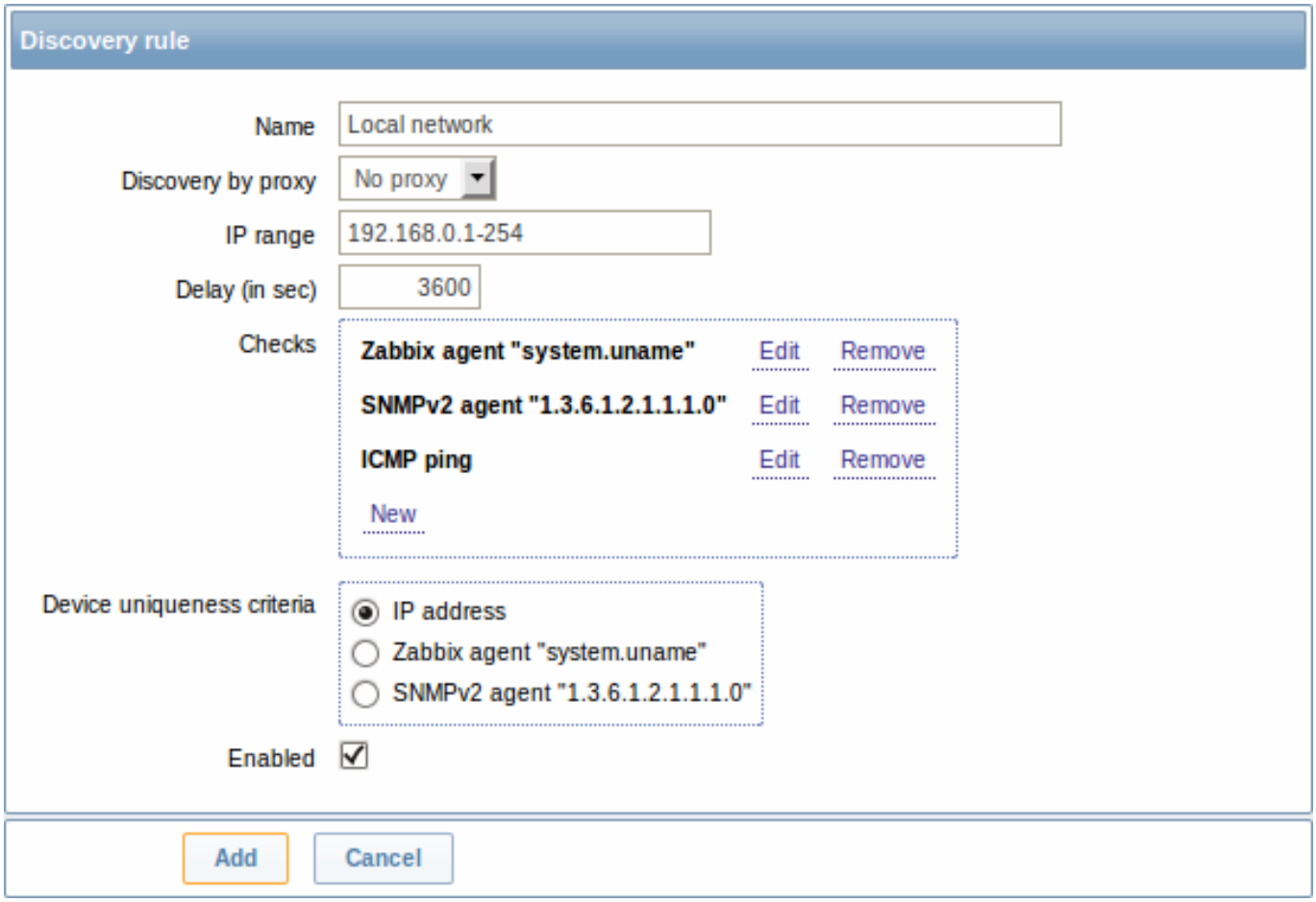
Configuring a network discovery rule

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



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: 192.168.1.1-255 (until Zabbix 2.4.4), 192.168.1-10.1-255 (from Zabbix 2.4.4). 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
Delay (in sec)	This parameter defines how often Zabbix will execute the rule. Delay is measured after the execution of previous discovery instance ends so there is no overlap.

Parameter	Description
Checks	Zabbix will use this list of checks for discovery. 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. 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** - no processing of multiple single-IP devices. If a device with the same IP already exists it will be considered already discovered and a new host will not be added. **Type of discovery check** - either SNMP or Zabbix agent check.
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.

Changing proxy setting

Since Zabbix 2.2.0 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

A real life scenario

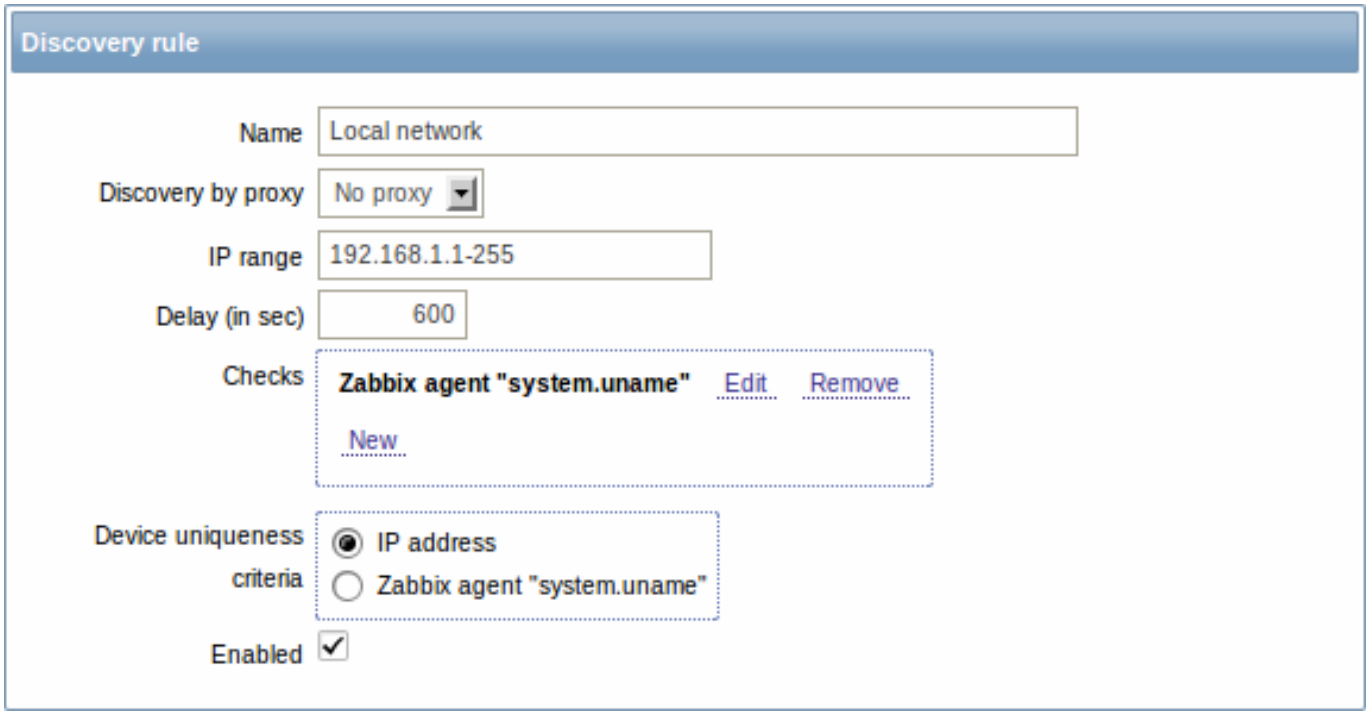
In this example we would like to set up network discovery for the local network having an IP range of 192.168.1.1-192.168.1.255.

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_Linux* for Linux hosts
- use *Template_Windows* for Windows hosts

Step 1

Defining a network discovery rule for our IP range.



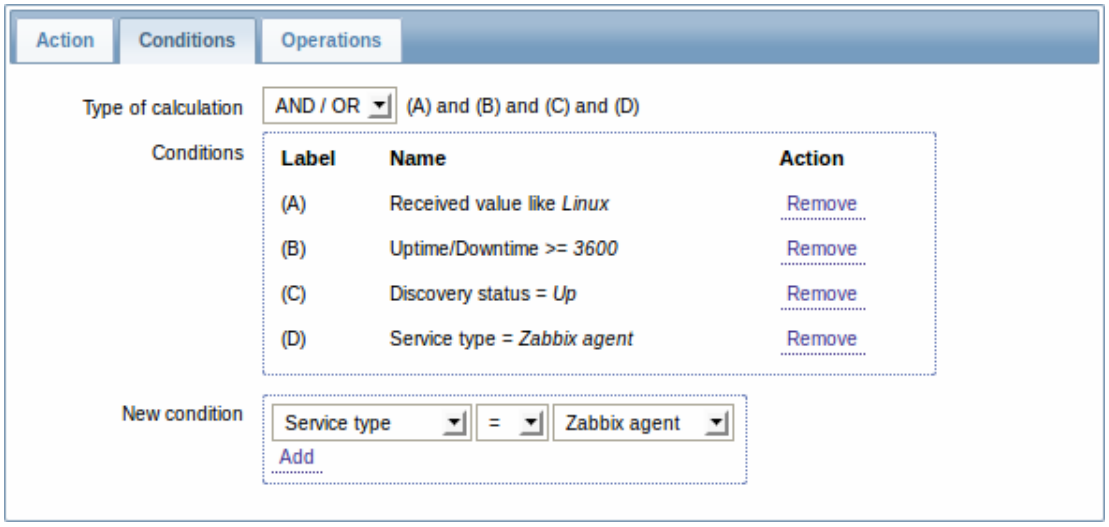
Zabbix will try to discover hosts in the IP range of 192.168.1.1-192.168.1.255 by connecting to Zabbix agents and getting the value from **system.uname** key. The value received from the agent can be used to apply different actions for different operating systems. For example, link Windows servers to Template_Windows, Linux servers to Template_Linux.

The rule will be executed every 10 minutes (600 seconds).

With this rule is added, Zabbix will automatically start the discovery and generating discovery-based events for further processing.

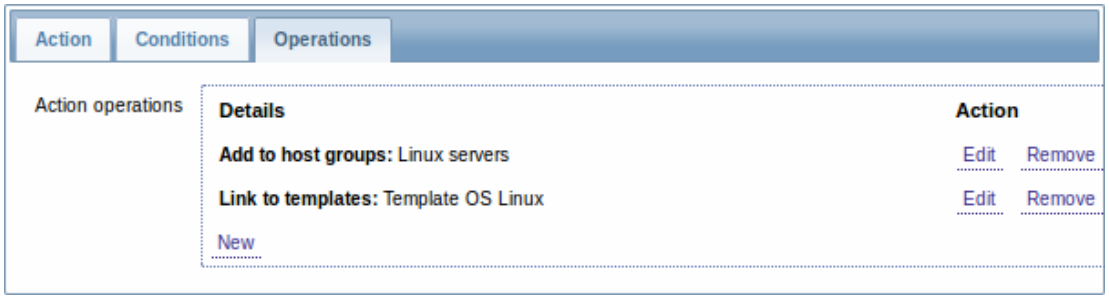
Step 2

Defining an 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

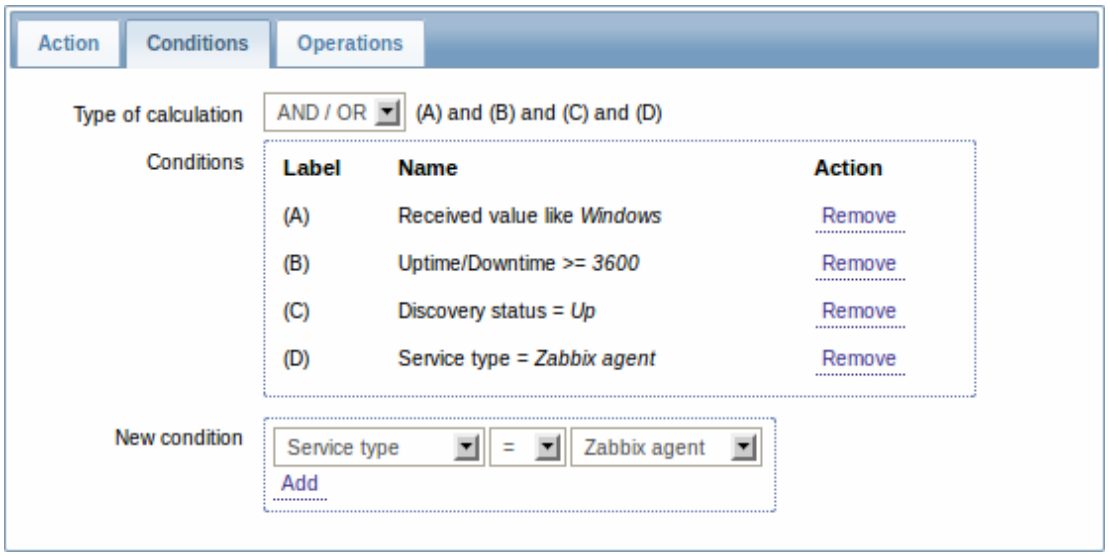


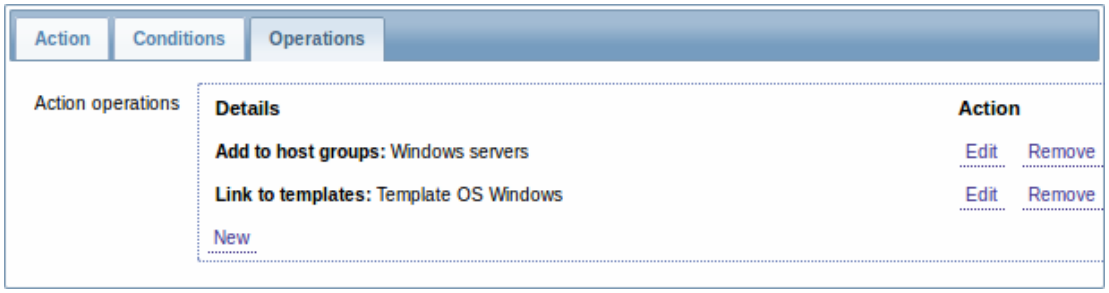
The action will execute the following operations:

- add the discovered host to the ”Linux servers” group (and also add host if it wasn’t added previously)
- link host to the ”Template_Linux” template. Zabbix will automatically start monitoring the host using items and triggers from ”Template_Linux”.

Step 3

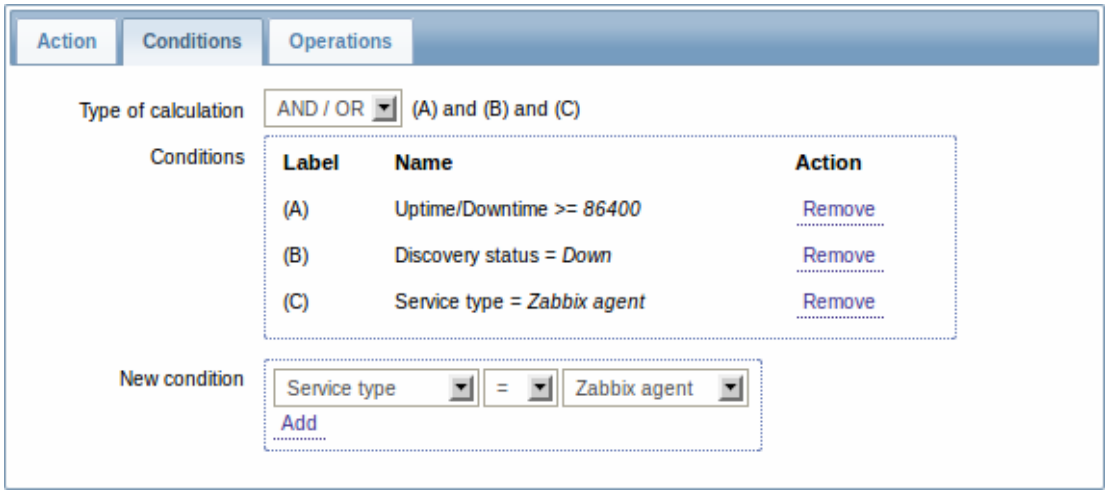
Defining an action for adding the discovered Windows servers to the respective group/template.

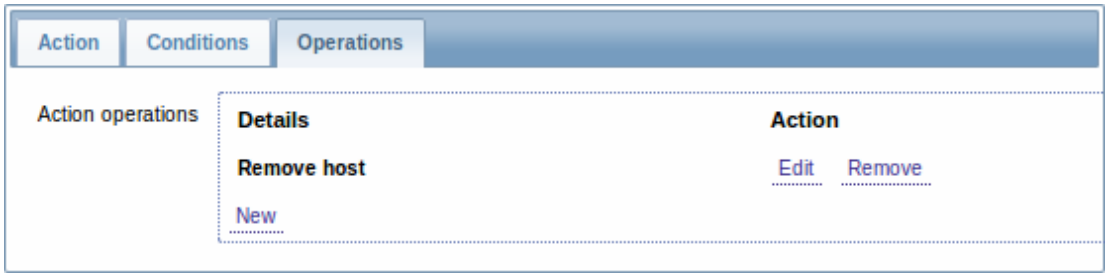




Step 4

Defining an 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 auto-registration

Overview

It is possible to allow active Zabbix agent auto-registration, after which the server can start monitoring them. This way new hosts can be added for monitoring without configuring them manually on the server.

Auto registration 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 auto-registration 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 auto-registered 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.

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 auto-registration

When server receives an auto-registration request from an agent it calls an action. An action of event source ”Auto registration” must be configured for agent auto-registration.

Note:
Setting up network discovery is not required to have active agents auto-register.

In the Zabbix frontend, go to *Configuration → Actions*, select *Auto registration* as the event source and click on *Create action*:

- In the Action tab, give your action a name
- In the Conditions tab, optionally specify conditions. 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 groups’ (for example, *Discovered hosts*), ’Link to templates’, etc.

Note:
If the hosts that will be auto-registering 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.

Using host metadata

When agent is sending an auto-registration 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.

<note:important>An auto-registration 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 auto-registered 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 auto-registration, which might not be enough. In order to make sure the proper template is applied to the host you should use host metadata.

Agent configuration

The first thing to do is configuring 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.

Frontend configuration

Now you need to configure the frontend. Create 2 actions. The first action:

- Name: Linux host autoregistration
- Conditions: Host metadata like *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 like *Windows*
- Operations: Link to templates: Template OS Windows

Example 2

Using host metadata to allow some basic protection against unwanted hosts registering.

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 some hard-to-guess secret text.

Do not forget to restart the agent after making any changes to the configuration file.

Frontend configuration

Create an action in the frontend, using the above mentioned hard-to-guess secret code to disallow unwanted hosts:

- Name: Auto registration action Linux
- Conditions:

```
    * Type of calculation: AND
    * Condition (A): Host metadata like //Linux//
    * Condition (B): Host metadata like //21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
* 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 are transmitted in plain text.

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.

In Zabbix, four types of item discovery are supported out of the box:

- discovery of file systems;
- discovery of network interfaces;
- discovery of CPUs and CPU cores;
- discovery of SNMP OIDs.

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 specific 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”.

> **Note:**
> Low-level discovery items ”vfs.fs.discovery” and ”net.if.discovery” are supported since Zabbix agent version 2.0.
> Discovery item ”system.cpu.discovery” is supported since Zabbix agent version 2.4.
> Discovery of SNMP OIDs is supported since Zabbix server and proxy version 2.0.

Note:
Return values of a low-level discovery rule are limited to 2048 bytes on a Zabbix proxy run with IBM DB2 database. This limit does not apply to Zabbix server as return values are processed without being stored in a database.

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.

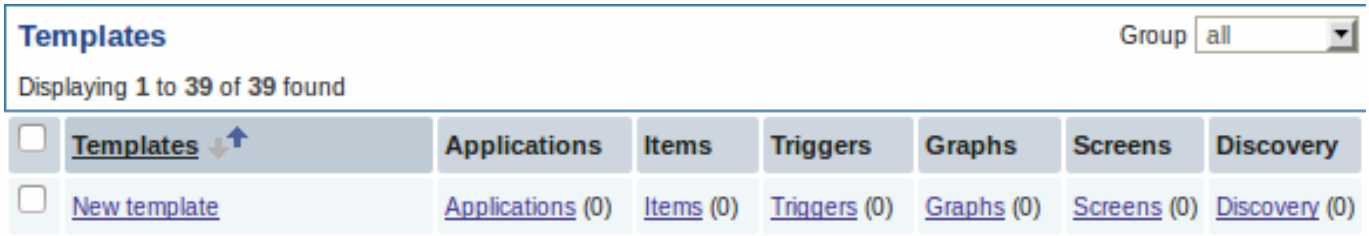
When the server receives a value for a discovery item, it looks at the macro → value pairs and for each pair generates real items, triggers, and graphs, based on their prototypes. In the example with "net.if.discovery" above, the server would generate one set of items, triggers, and graphs for the loopback interface "lo", and another set for interface "eth0".

The following sections illustrate the process described above in detail and serve as a how-to for performing discovery of file systems, network interfaces, and SNMP OIDs. The last section describes the JSON format for discovery items and gives an example of how to implement your own file system discoverer as a Perl script.

3.1 Discovery of file systems

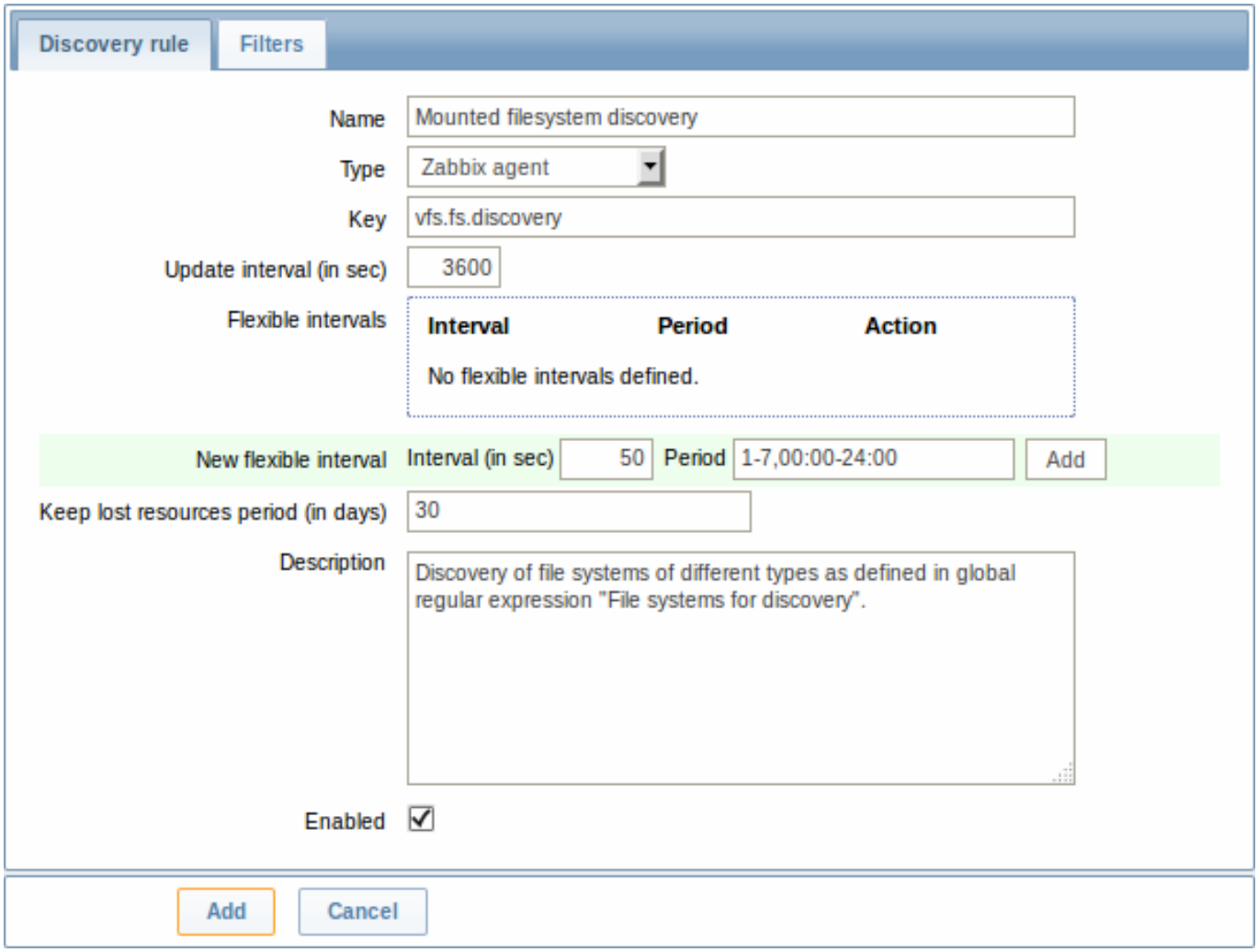
To configure the discovery of file systems, do the following:

- 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 form with the following details

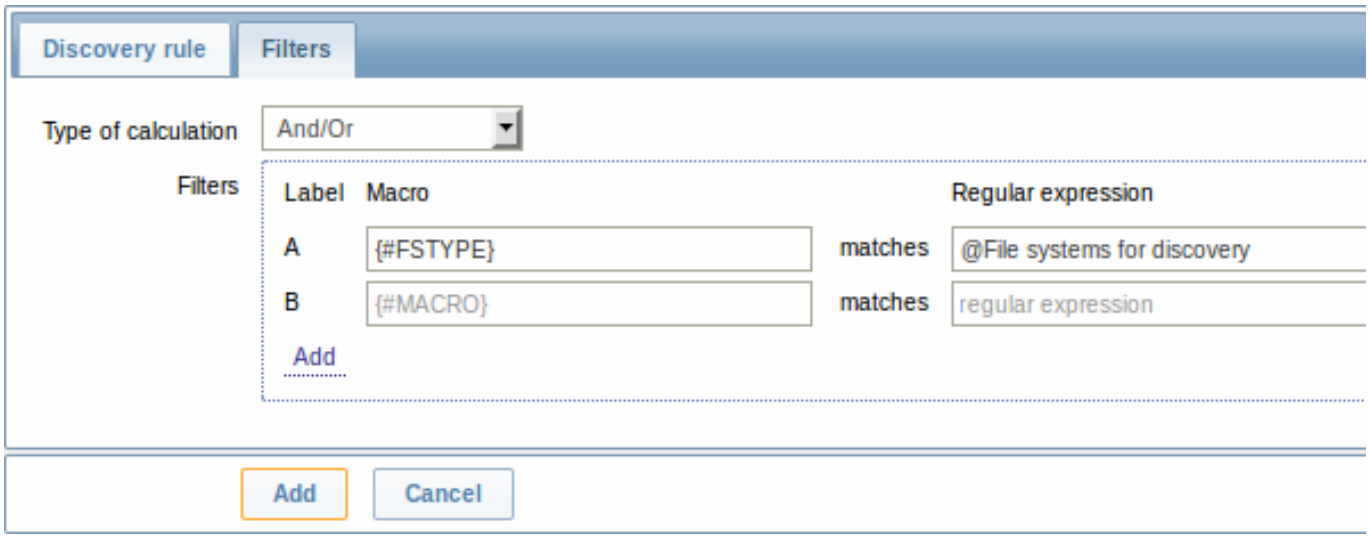
The **Discovery rule** tab contains general discovery rule attributes:



Parameter	Description
Name	Name of discovery rule.
Type	The type of check to perform discovery; should be *Zabbix agent* or *Zabbix agent (active)* for file system discovery.
Key	An item with "vfs.fs.discovery" key is built into the Zabbix agent on many platforms (see supported item key list for details), and will return a JSON with the list of file systems present on the computer and their types.
Update interval (in sec)	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. *Note*: If set to '0', the item will not be polled. However, if a flexible interval also exists with a non-zero value, the item will be polled during the flexible interval duration.
Flexible intervals	You can create exceptions to *Update interval*. For example: Interval: **0**, Period: **6-7,00:00-24:00** - will disable the polling at the weekend. Otherwise default update interval will be used. Up to seven flexible intervals can be defined. If multiple flexible intervals overlap, the smallest *Interval* value is used for the overlapping period. See Time period specification page for description of the *Period* format. *Note*: If set to '0', the item will not be polled during the flexible interval duration and will resume polling according to the *Update interval* once the flexible interval period is over.

Parameter	Description
Keep lost resources period (in days)	This field allows you to specify for how many days the discovered entity will be retained (won't be deleted) once its discovery status becomes "Not discovered anymore" (max 3650 days). *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.

The **Filters** tab contains discovery rule filter definitions:

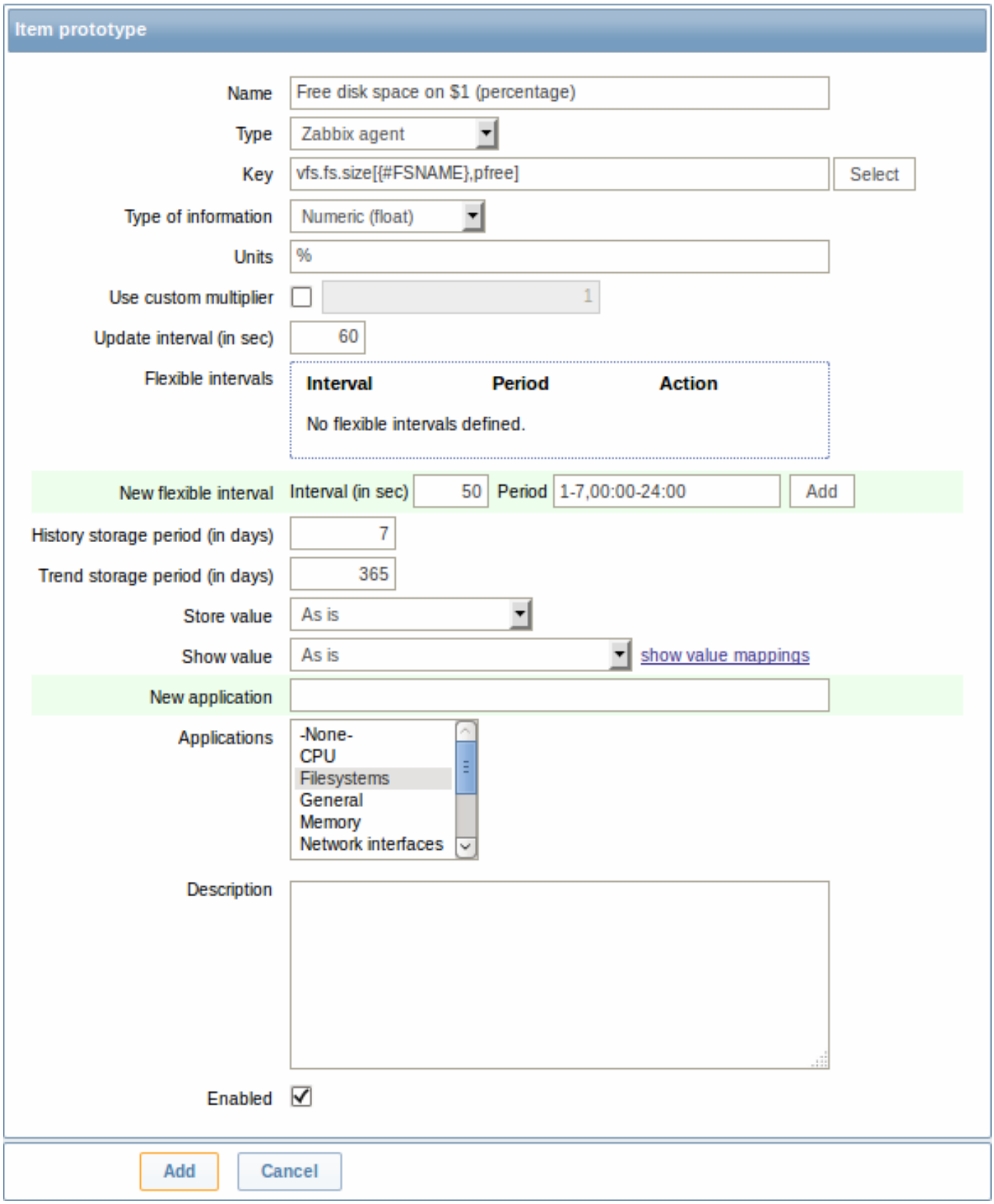


Parameter	Description
Type of calculation	The following options for calculating filters are available: **And** - all filters must be passed; **Or** - enough if one filter is passed; **And/Or** - uses *And* with different macro names and *Or* with the same macro name; **Custom expression** - a user-defined formula for calculating filters. The formula must include all filters in the list (represented as uppercase letters A, B, C, ...) and may include spaces, tabs, brackets ( ), **and** (case sensitive), **or** (case sensitive). Limited to 255 symbols. The custom calculation is the same as in action conditions.
Filters	A filter can be used to generate only real items, triggers, and graphs for certain file systems. It expects a POSIX Extended Regular Expression. 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"). 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` 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.

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.

Note:
Discovery rule history is not preserved.

Once a rule is created, go to the items for that rule and press "Create prototype" to create an item prototype. Note how macro {#FSNAME} is used where a file system name is required. When the discovery rule is processed, this macro will be substituted with the discovered file system.



Note:
If an item prototype is created with a *Disabled* status, it will be added to a discovered entity, but in a disabled state.

We can create several item prototypes for each file system metric we are interested in:

Item prototypes of Mounted filesystem discovery

Displaying **1** to **5** of **5** found

« Template list **Template:** Template OS Linux « Discovery list **Discovery:** Mounted filesystem discovery Item pr

Trigger prototypes (2) Graph prototypes (1) Host prototypes (0)

Name	Key	Interval	History	Trends	Type
Free disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},free]	60	7	365	Zabbix agent
Free disk space on {#FSNAME} (percentage)	vfs.fs.size[{#FSNAME},pfree]	60	7	365	Zabbix agent
Free inodes on {#FSNAME} (percentage)	vfs.fs.inode[{#FSNAME},pfree]	60	7	365	Zabbix agent
Total disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},total]	3600	7	365	Zabbix agent
Used disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},used]	60	7	365	Zabbix agent

Then, we create trigger prototypes in a similar way:

Trigger prototype

Name: Free disk space is less than 20% on volume {#FSNAME}

Expression: {Template OS Linux:vfs.fs.size[{#FSNAME},pfree].last(0)}<20 Add

Expression constructor

Multiple PROBLEM events generation:

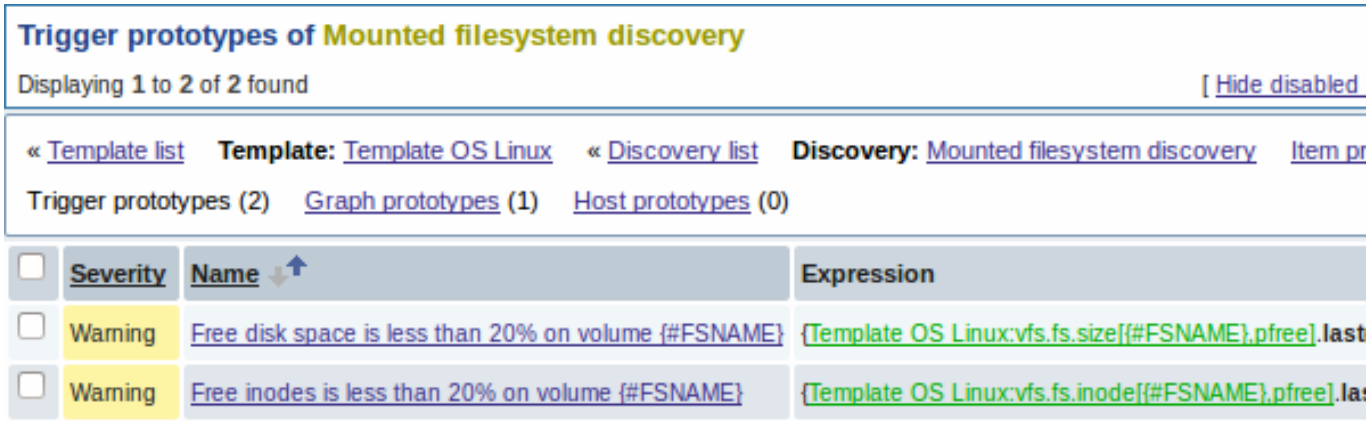
Description:

URL:

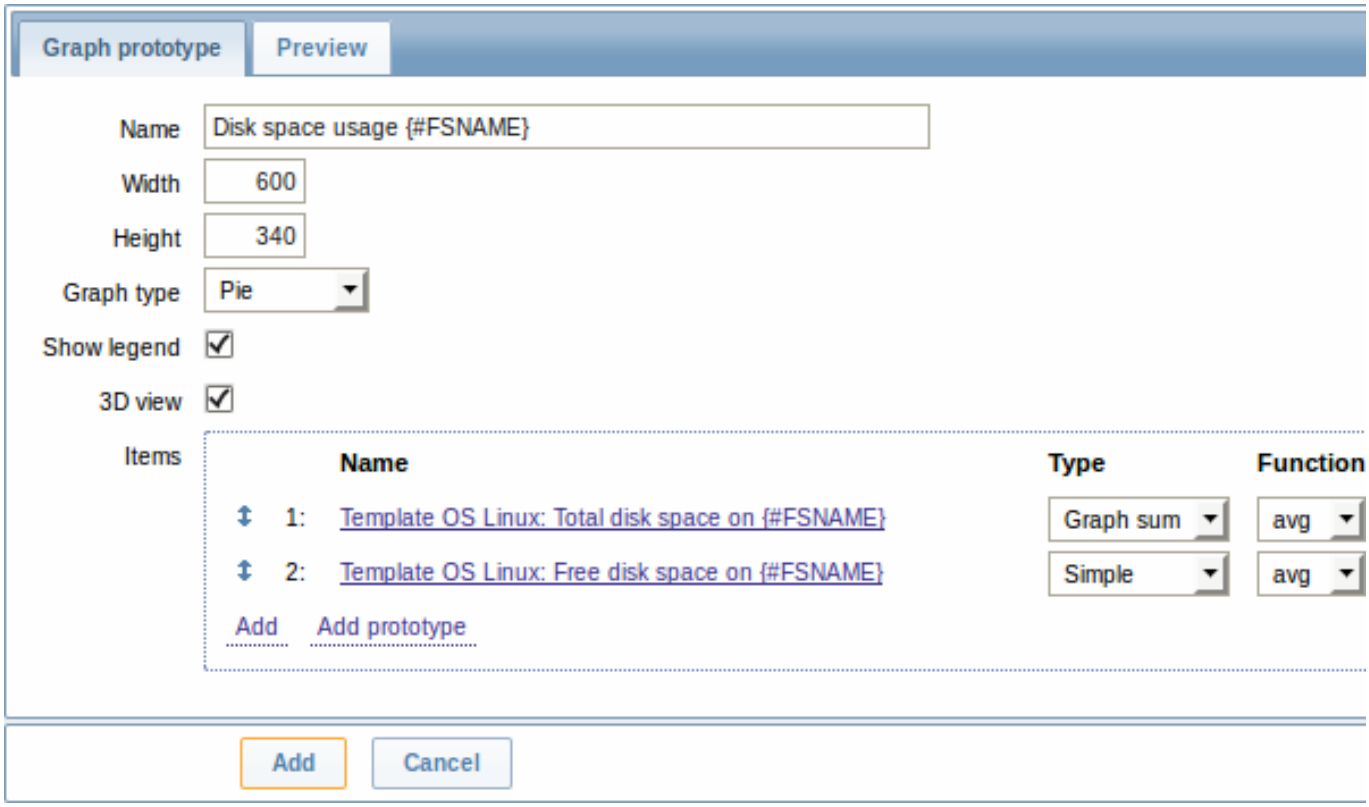
Severity: Not classified | Information | **Warning** | Average | High | Disaster

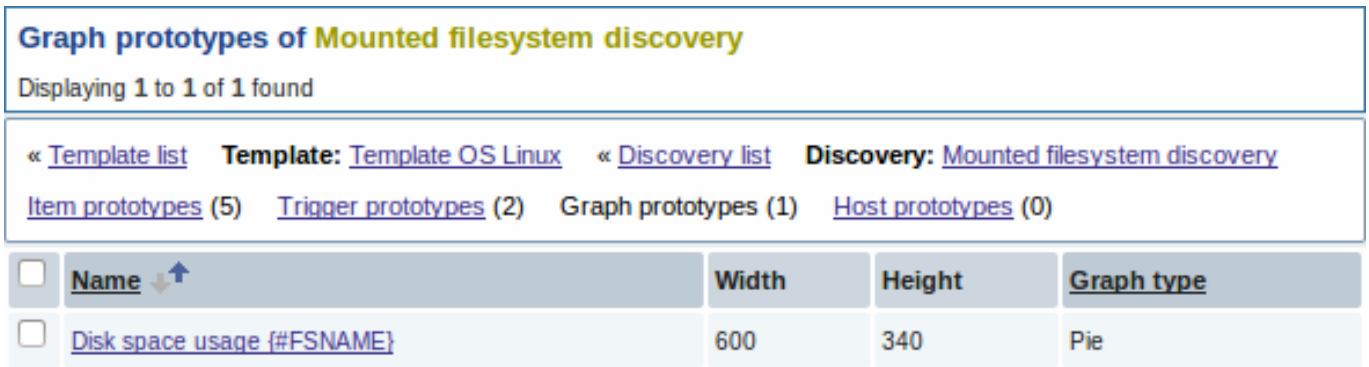
Enabled:

Add Cancel

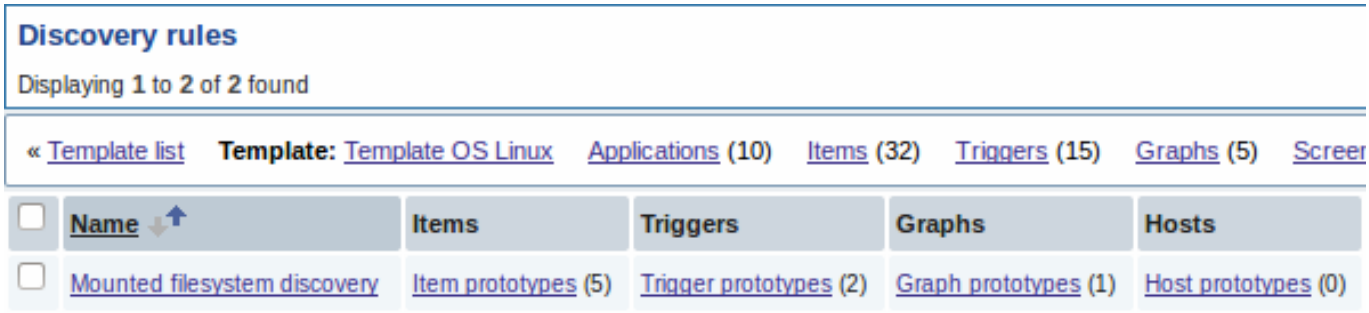


And graph prototypes too:



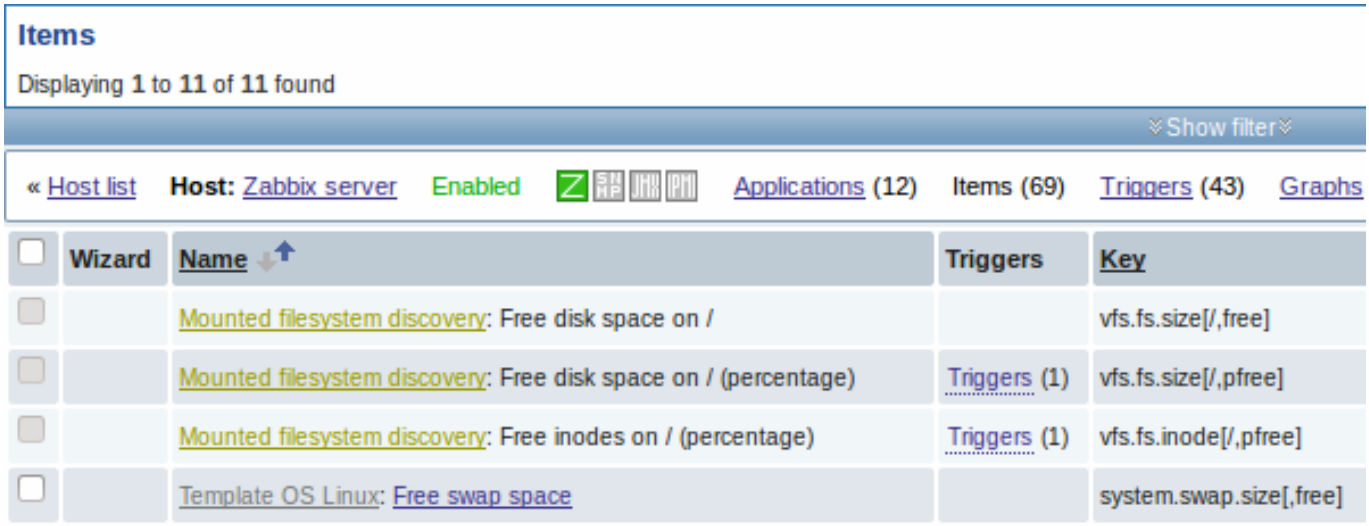


Finally, we have created a discovery rule that looks like shown below. It has five item prototypes, two trigger prototypes, and one graph prototype.



Note: For configuring host prototypes, see the section about host prototype configuration in virtual machine monitoring.

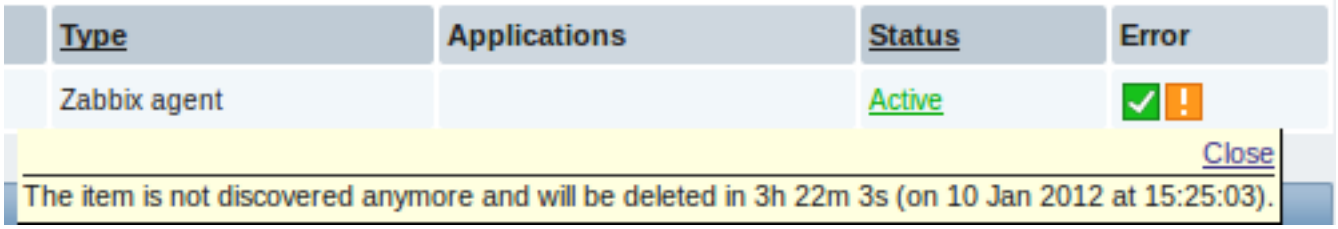
The screenshots below illustrate how discovered items, triggers, and graphs look like in the host's configuration. Discovered entities are prefixed with a golden link to a discovery rule they come from.



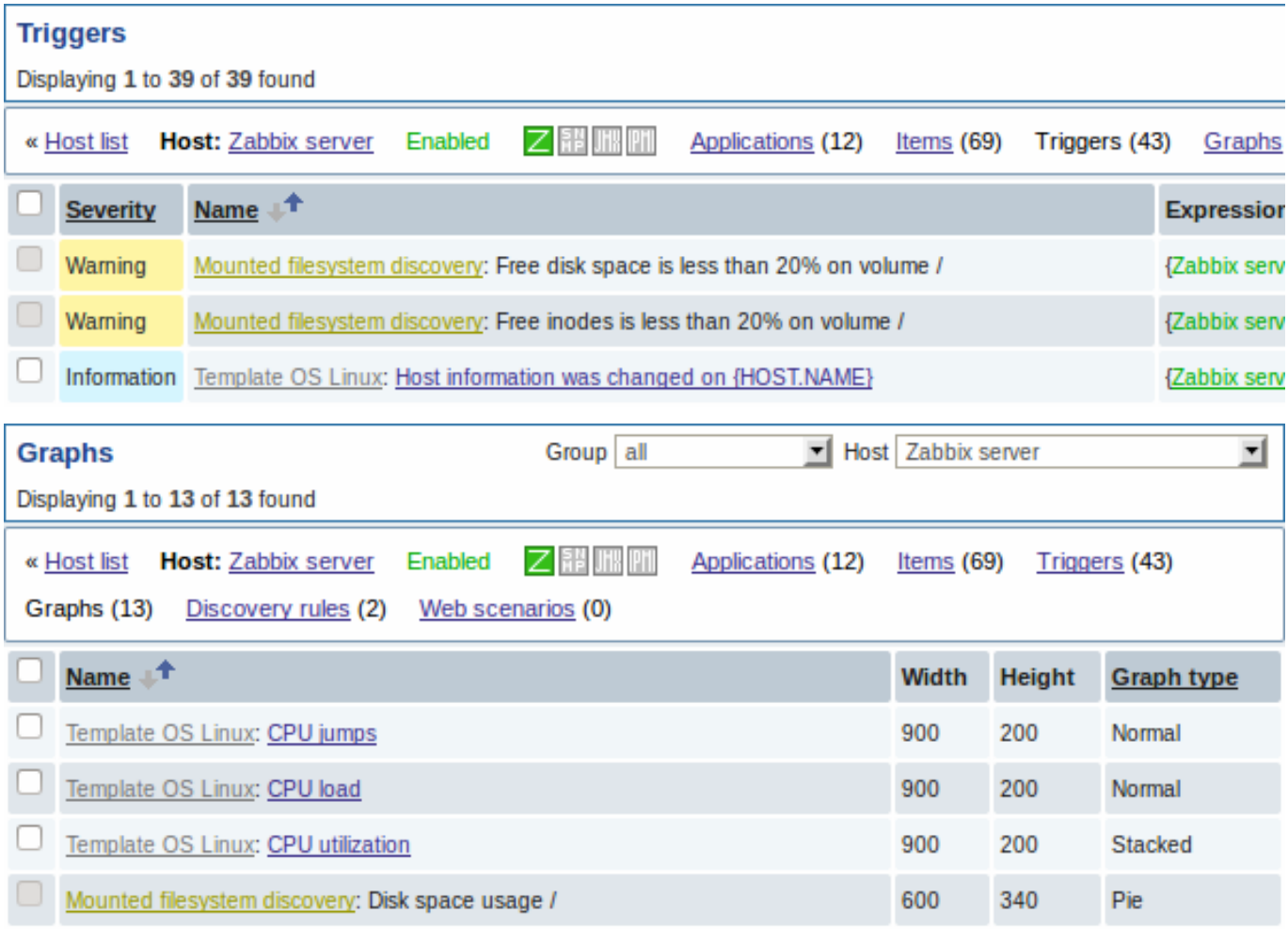
Note that discovered entities will not be created in case there are already existing entities with the same uniqueness criteria, for example, an item with the same key or graph with the same name.

Items (similarly, triggers and graphs) created by a low-level discovery rule cannot be manually deleted. However, they 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', an orange 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.



3.2 Discovery of network interfaces

Discovery of network interfaces is done in exactly the same way as discovery of file systems, except that you use the discovery rule key "net.if.discovery" instead of "vfs.fs.discovery" and use macro {#IFNAME} instead of {#FSNAME} in filter and item/trigger/graph prototypes.

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]".

See above for more information about the filter.

3.3 Discovery of CPUs and CPU cores

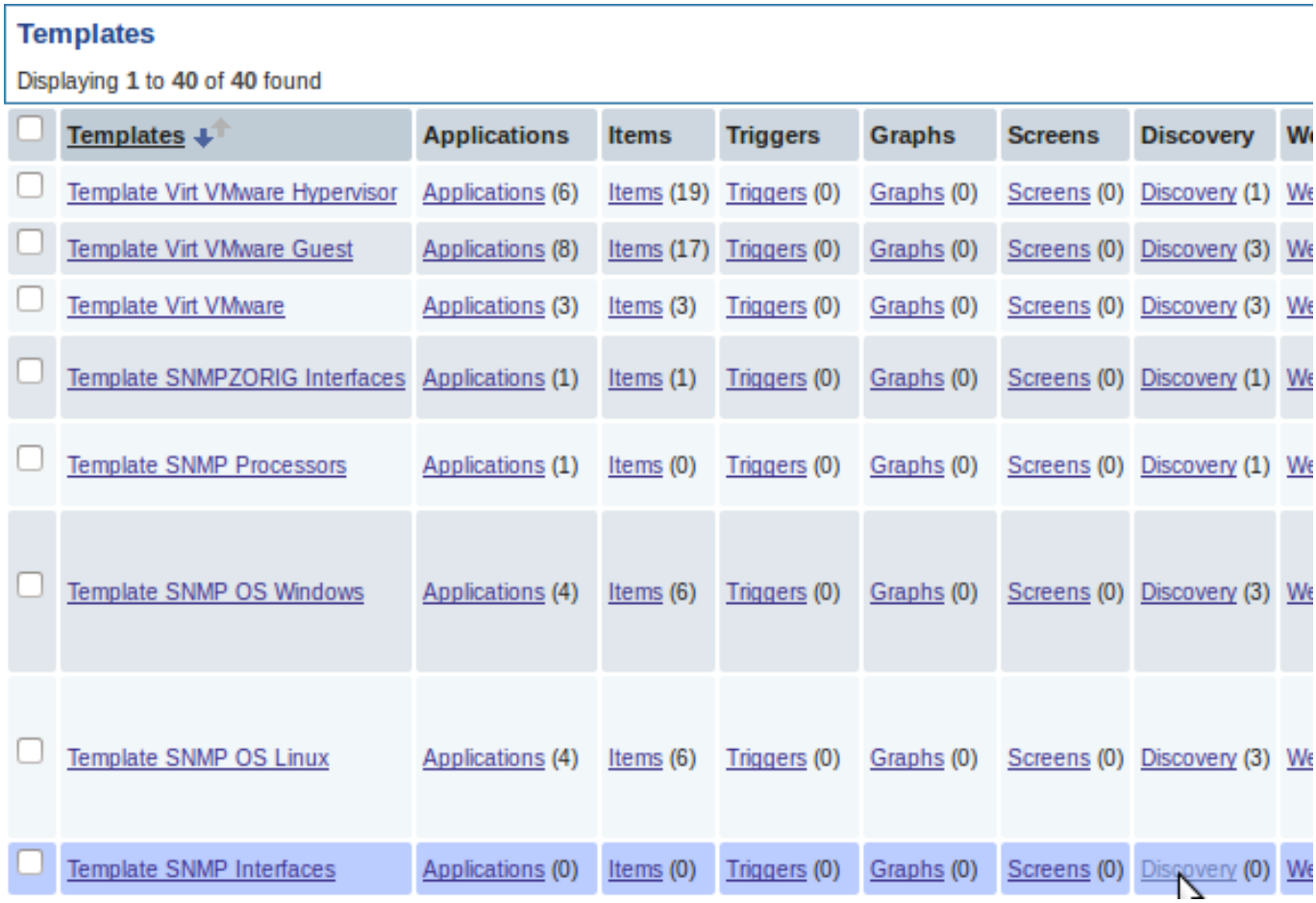
Discovery of CPUs and CPU cores is done in a similar fashion as network interface discovery with the exception being that the discovery rule key is "system.cpu.discovery". This discovery key returns two macros - {#CPU.NUMBER} and {#CPU.STATUS} identifying the CPU order number and status respectively. To note, 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>]" or "system.hw.cpu[{#CPU.NUMBER}, <info>]".

3.4 Discovery of SNMP OIDs

In this example, we will perform SNMP discovery on a switch. First, go to "Configuration" → "Templates".



To edit discovery rules for a template, click on the link in the ”Discovery” column.

Then, press ”Create rule” and fill the form with the details in the screenshot below.

Unlike file system and network interface discovery, the item does not necessarily have to have ”snmp.discovery” key - item type of SNMP agent is sufficient.

Also, unlike the previous examples, this discovery item will generate two macros for each discovered entity: {#SNMPINDEX} and {#SNMPVALUE}. In case you would like to filter out loopback interfaces from returned values you could put ”{#SNMPVALUE}” into filter “Macro” and ”^([^l]|l$)[^o]?” regular expression into “Regexp” text fields. See above for more information about the filter.

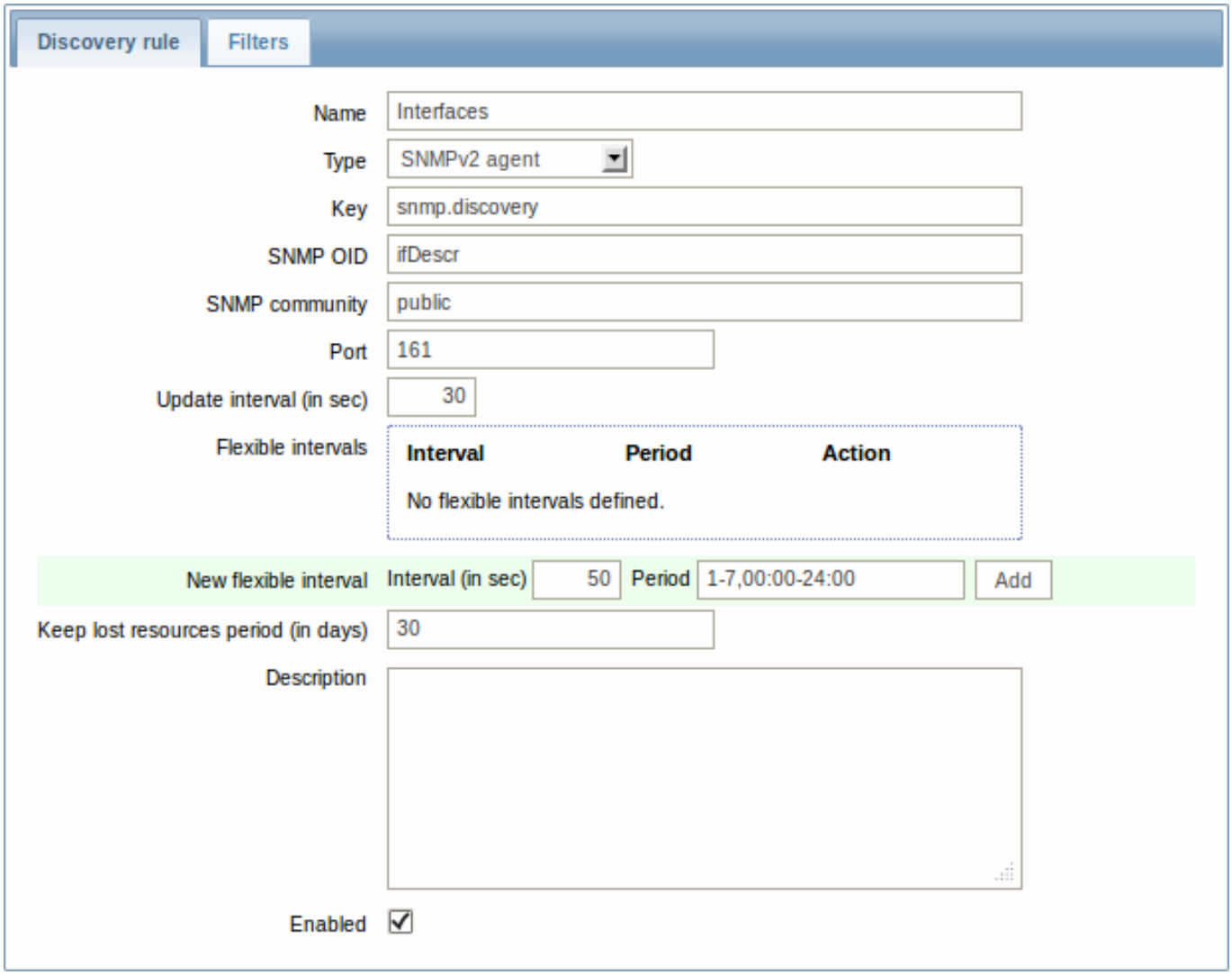
In ”SNMP OID” field, we have to put an OID that is capable of generating meaningful values for these macros.

To understand what we mean, let us perform snmpwalk on our switch:

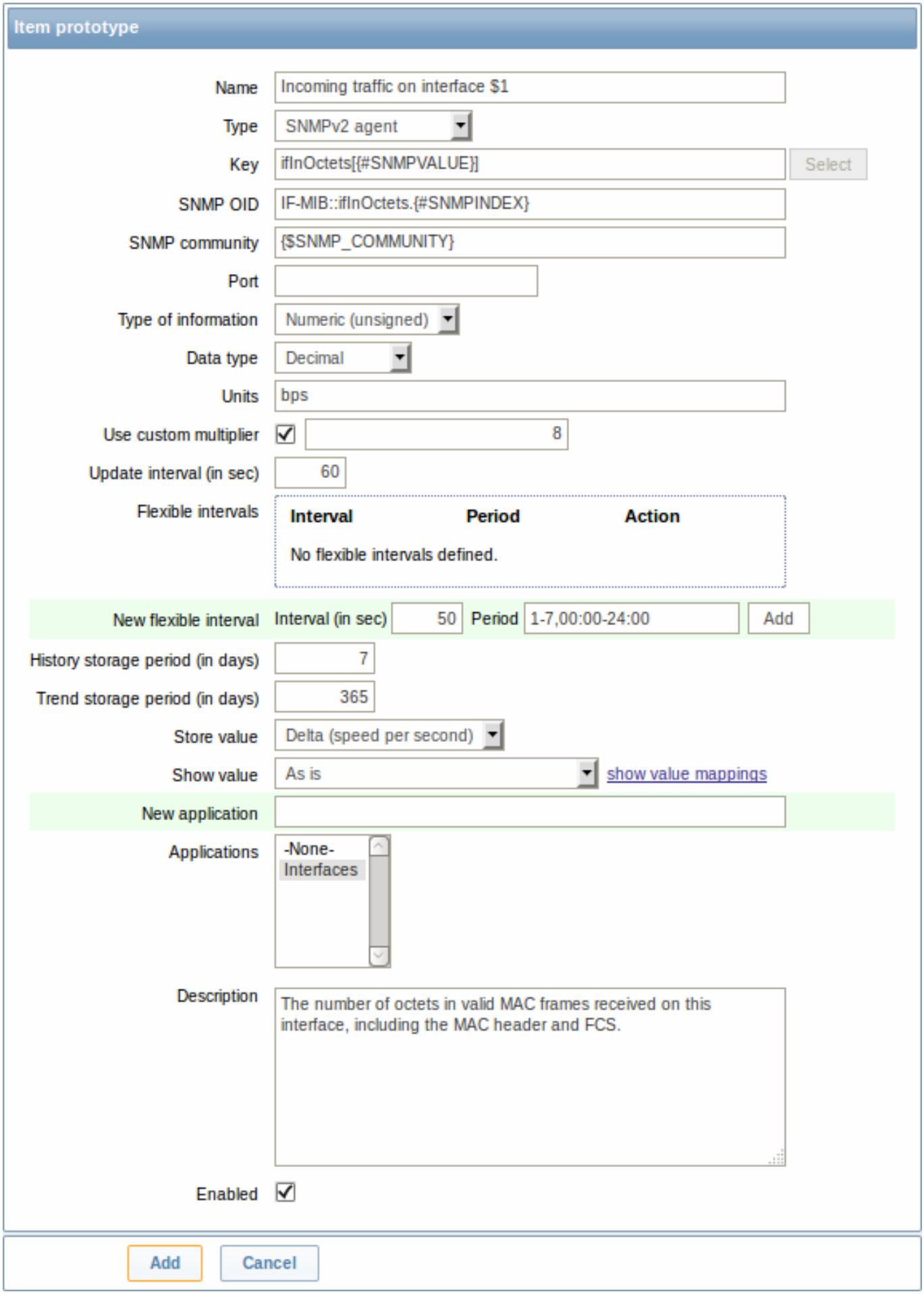
```
$ snmpwalk -v 2c -c public 192.168.1.1 IF-MIB::ifDescr
IF-MIB::ifDescr.1 = STRING: WAN
IF-MIB::ifDescr.2 = STRING: LAN1
IF-MIB::ifDescr.3 = STRING: LAN2
```

Macro {#SNMPINDEX} takes its value from the part of the OID that is after ifDescr (in this example: 1, 2, 3). Macro {#SNMPVALUE} comes from the value of the corresponding OID (here: WAN, LAN1, LAN2). Thus, our ”snmp.discovery” item would return three sets of macro → value pairs:

```
{#SNMPINDEX} → 1   {#SNMPVALUE} → WAN
{#SNMPINDEX} → 2   {#SNMPVALUE} → LAN1
{#SNMPINDEX} → 3   {#SNMPVALUE} → LAN2
```



The following screenshot illustrates how we can use these macros in item prototypes:



Again, creating as many item prototypes as needed:

Item prototypes of Network interfaces

Displaying **1** to **8** of **8** found

« Template list **Template:** Template SNMP Interfaces « Discovery list **Discovery:** Network interfaces Item proto

Trigger prototypes (1) Graph prototypes (1) Host prototypes (0)

Name	Key	Interval	History	Trends	Type
Admin status of interface {#SNMPVALUE}	ifAdminStatus[{#SNMPVALUE}]	60	7	365	SNMPv2
Alias of interface {#SNMPVALUE}	ifAlias[{#SNMPVALUE}]	3600	7		SNMPv2
Description of interface {#SNMPVALUE}	ifDescr[{#SNMPVALUE}]	3600	7		SNMPv2
Inbound errors on interface {#SNMPVALUE}	ifInErrors[{#SNMPVALUE}]	60	7	365	SNMPv2
Incoming traffic on interface {#SNMPVALUE}	ifInOctets[{#SNMPVALUE}]	60	7	365	SNMPv2
Operational status of interface {#SNMPVALUE}	ifOperStatus[{#SNMPVALUE}]	60	7	365	SNMPv2
Outbound errors on interface {#SNMPVALUE}	ifOutErrors[{#SNMPVALUE}]	60	7	365	SNMPv2
Outgoing traffic on interface {#SNMPVALUE}	ifOutOctets[{#SNMPVALUE}]	60	7	365	SNMPv2

As well as trigger prototypes:

Trigger prototype

Name: Operational status was changed on {HOST.NAME} interface {#SNMI

Expression: {Template SNMP Interfaces:ifOperStatus[{#SNMPVALUE}].diff(0)}=1 Add

Expression constructor

Multiple PROBLEM events generation:

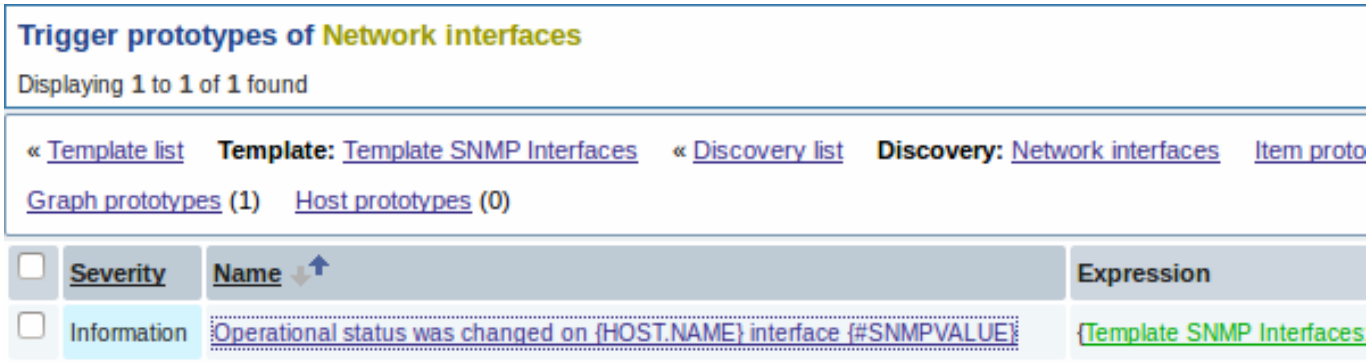
Description:

URL:

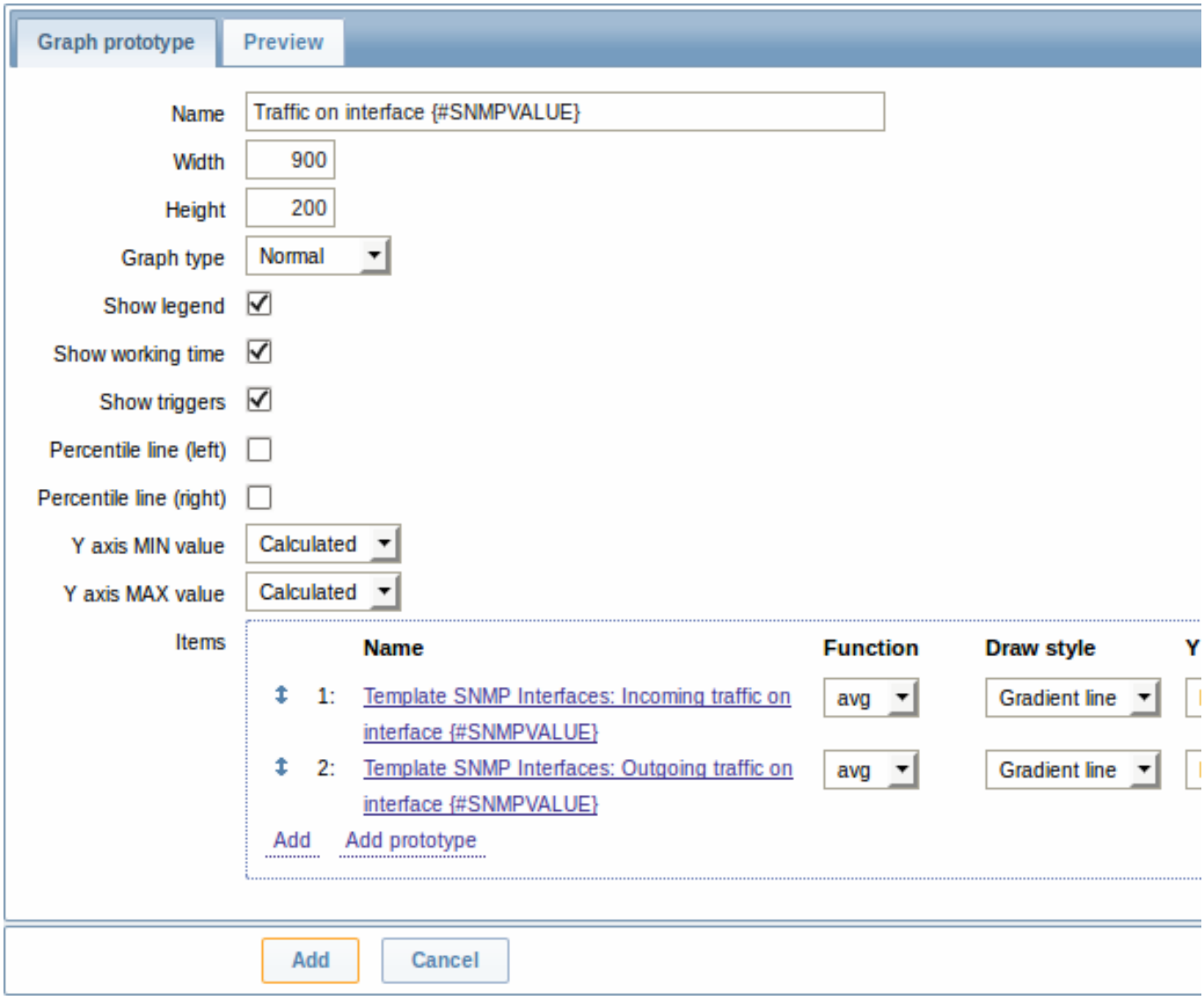
Severity: Not classified | Information | Warning | Average | High | Disaster

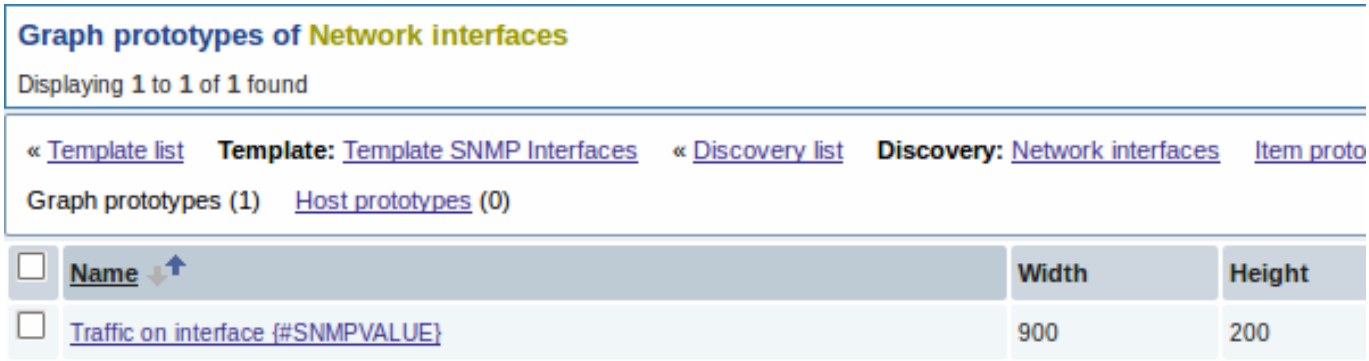
Enabled:

Add Cancel



And graph prototypes:





A summary of our discovery rule:

Discovery rules

Displaying **1** to **1** of **1** found

« Template list **Template:** Template SNMP Interfaces Applications (1) Items (1) Triggers (0) Graphs (0) Screens (0)
Discovery rules (1) Web scenarios (0)

Name	Items	Triggers	Graphs	Hosts	Key	Interval	Type	Status
Network interfaces	Item prototypes (8)	Trigger prototypes (1)	Graph prototypes (1)	Host prototypes (0)	ifDescr	3600	SNMPv2 agent	Enabled

When server runs, it will create real items, triggers, and graphs, based on the values "snmp.discovery" returns. In host's configuration they will be prefixed with a golden link to a discovery rule they come from.

Items

Displaying **1** to **50** of **246** found

« Host list **Host:** HP Procurve switch Enabled Applications (2) Items (246) Triggers (30) G

1 | 2 | 3

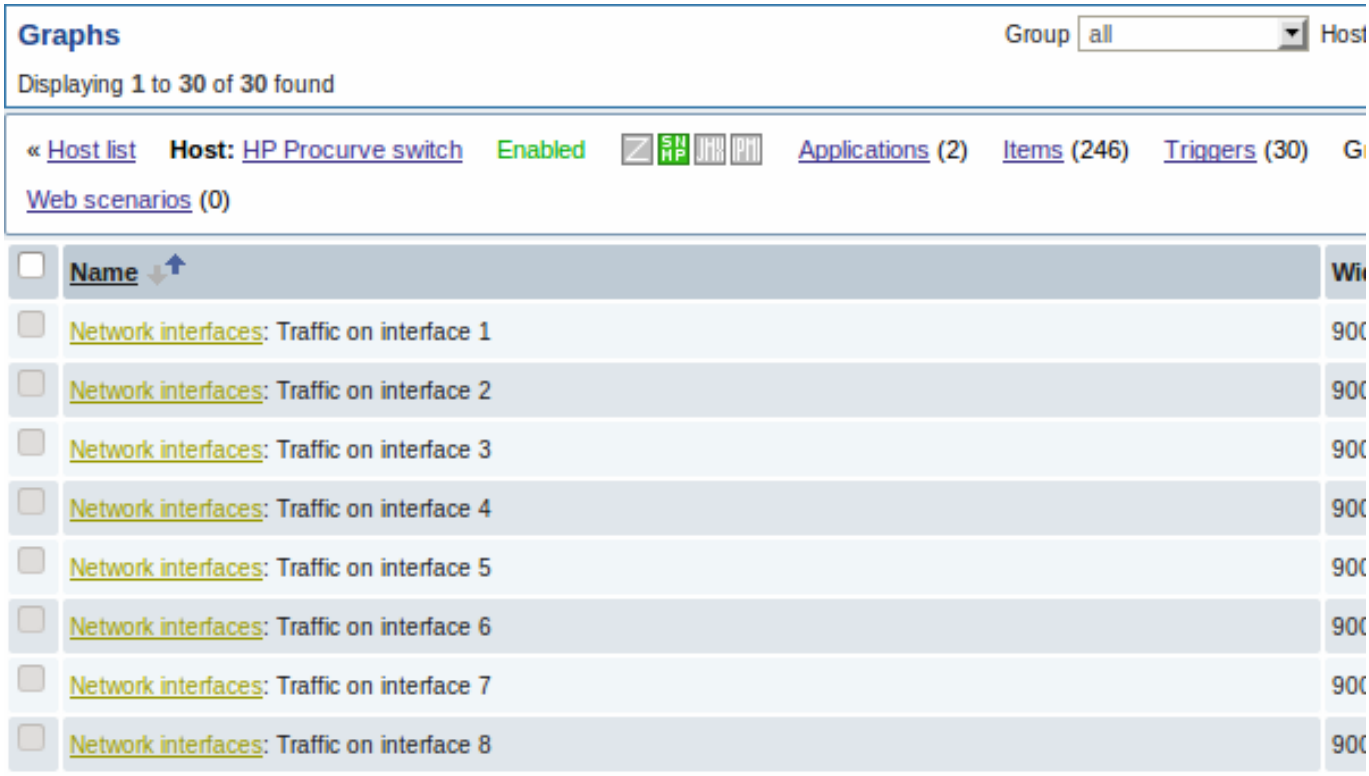
Wizard	Name	Triggers	Key
	Network interfaces: Admin status of interface 1		ifAdmin
	Network interfaces: Admin status of interface 2		ifAdmin
	Network interfaces: Admin status of interface 3		ifAdmin
	Network interfaces: Admin status of interface 4		ifAdmin
	Network interfaces: Admin status of interface 5		ifAdmin
	Network interfaces: Admin status of interface 6		ifAdmin
	Network interfaces: Admin status of interface 7		ifAdmin
	Network interfaces: Admin status of interface 8		ifAdmin

Triggers

Displaying **1** to **30** of **30** found

« Host list **Host:** HP Procurve switch Enabled Applications (2) Items (246) Triggers (30) G

Severity	Name
Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 1
Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 2
Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 3
Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 4
Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 5
Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 6
Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 7
Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 8



3.5 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";
print "\t\"data\":[\n\n";

for (`cat /proc/mounts`)
{
    ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;
    $fsname =~ s!/!\\/!g;

    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\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.

```
{
  "data":[

  { "{#FSNAME}":"\/",                          "{#FSTYPE}":"rootfs"   },
  { "{#FSNAME}":"\/sys",                       "{#FSTYPE}":"sysfs"    },
  { "{#FSNAME}":"\/proc",                      "{#FSTYPE}":"proc"     },
  { "{#FSNAME}":"\/dev",                       "{#FSTYPE}":"devtmpfs" },
  { "{#FSNAME}":"\/dev\/pts",                  "{#FSTYPE}":"devpts"   },
  { "{#FSNAME}":"\/",                          "{#FSTYPE}":"ext3"     },
  { "{#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.

15. 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[1]	**Yes**
Local administration	No
Ready for embedded hardware	**Yes**
One way TCP connections	**Yes**
Centralised configuration	**Yes**
Generates notifications	No

Note:
[1] Automatic DB creation feature only works with SQLite. Other databases require a manual setup.

1 Proxies

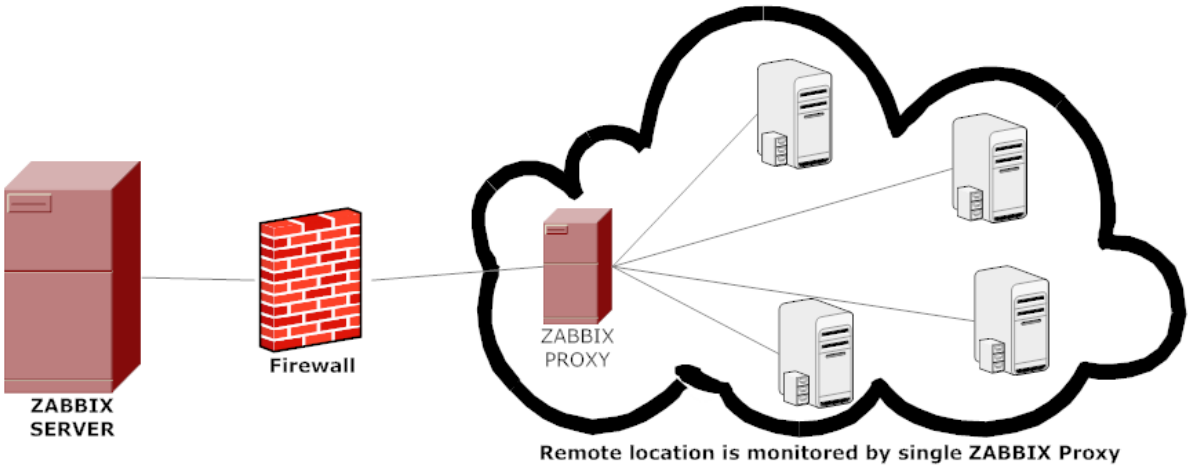
Overview

A Zabbix proxy can collect performance and availability data on behalf of the Zabbix server. This way, a proxy can take on itself some of the load of collecting data and offload the Zabbix server.

Also, using a proxy is the easiest way of implementing centralized and distributed monitoring, when all agents and proxies report to one Zabbix server and all data is collected centrally.

A Zabbix proxy can be used to:

- Monitor remote locations
- Monitor locations having unreliable communications
- Offload the Zabbix server when monitoring thousands of devices
- Simplify the maintenance of distributed monitoring



The proxy requires only one TCP connection to the Zabbix server. This way it is easier to get around a firewall as you only need to configure one firewall rule.

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**

Function		Supported by proxy
	Internal checks	**Yes**
	SSH checks	**Yes**
	Telnet checks	**Yes**
	External checks	**Yes**
Built-in web monitoring		**Yes**
Network discovery		**Yes**
Low-level discovery		**Yes**
Calculating triggers		*No*
Processing events		*No*
Sending alerts		*No*
Remote commands		*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.

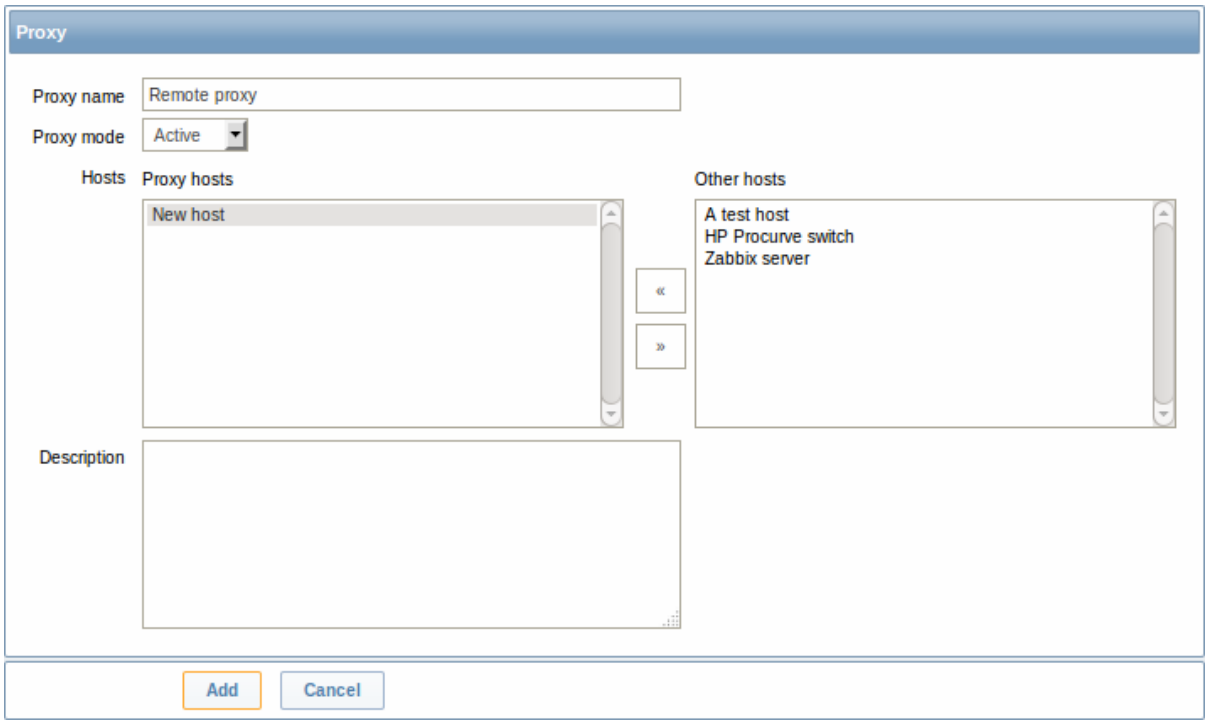
Configuration

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*

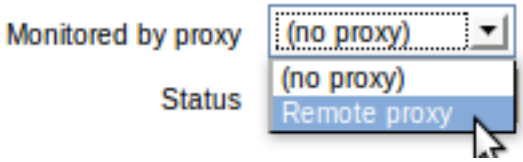


Parameter	Description
Proxy name	Enter the proxy name. It must be the same name as in the *Hostname* parameter in the proxy configuration file.

Parameter	Description
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 (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.
Hosts	Add hosts to be monitored by the proxy.
Description	Enter the proxy description.

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.

16. 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 and so on.

1 Frontend sections

1 Monitoring

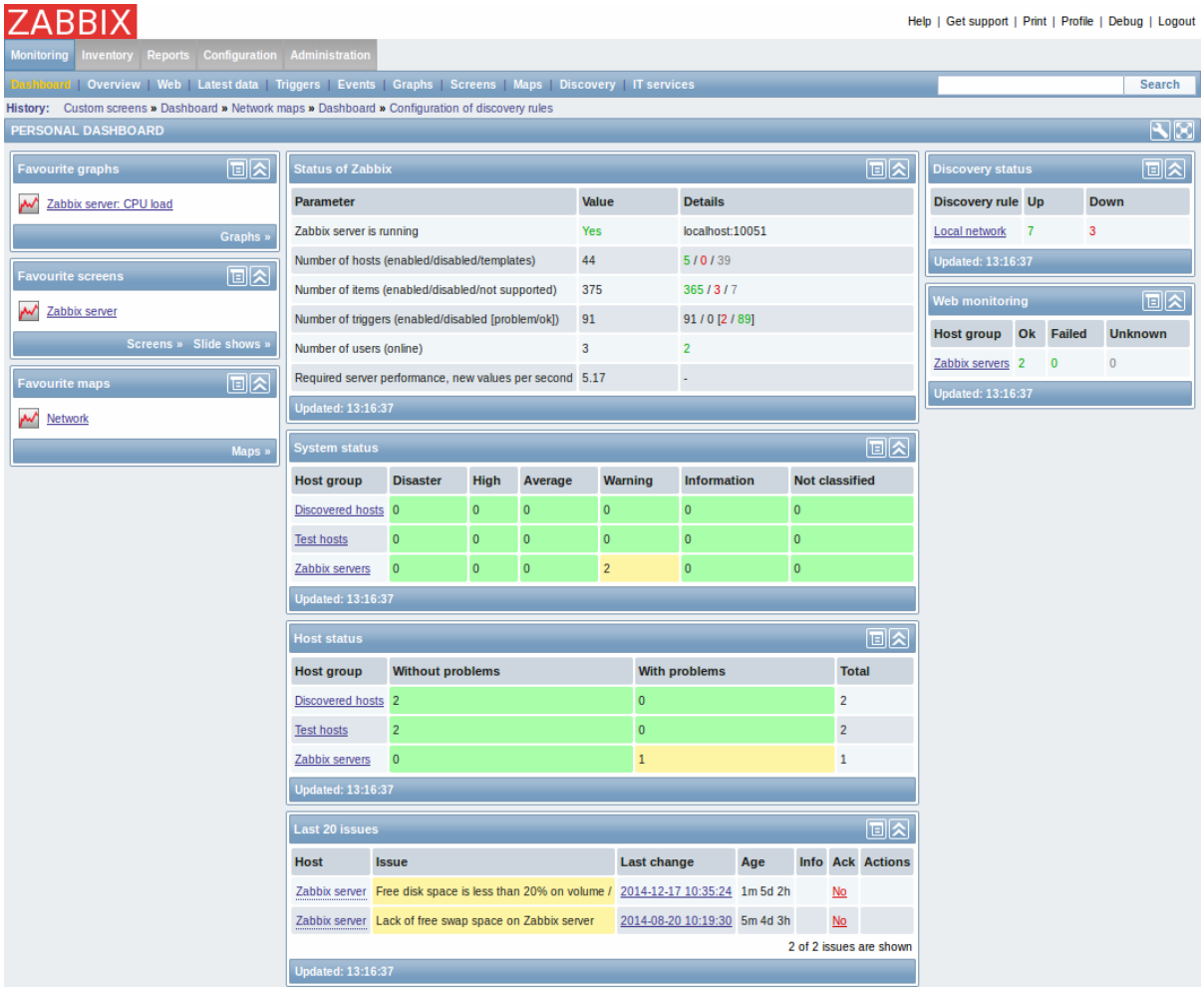
Overview

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.

1 Dashboard

Overview

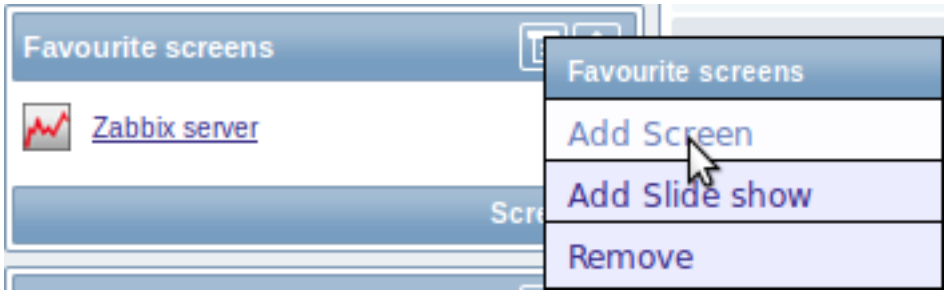
The *Monitoring* → *Dashboard* section, similar to the dashboard on your car, displays a summary of all the important information.



Favourites

There are some widgets for favourites where you can create quick shortcuts to the most needed graphs, custom graphs, screens, slide shows and maps.

Just click on the Menu button in the widget, select to add, for example, some screen and then select from the configured screens. The selected screens will be displayed as shortcuts in the favourites widget.



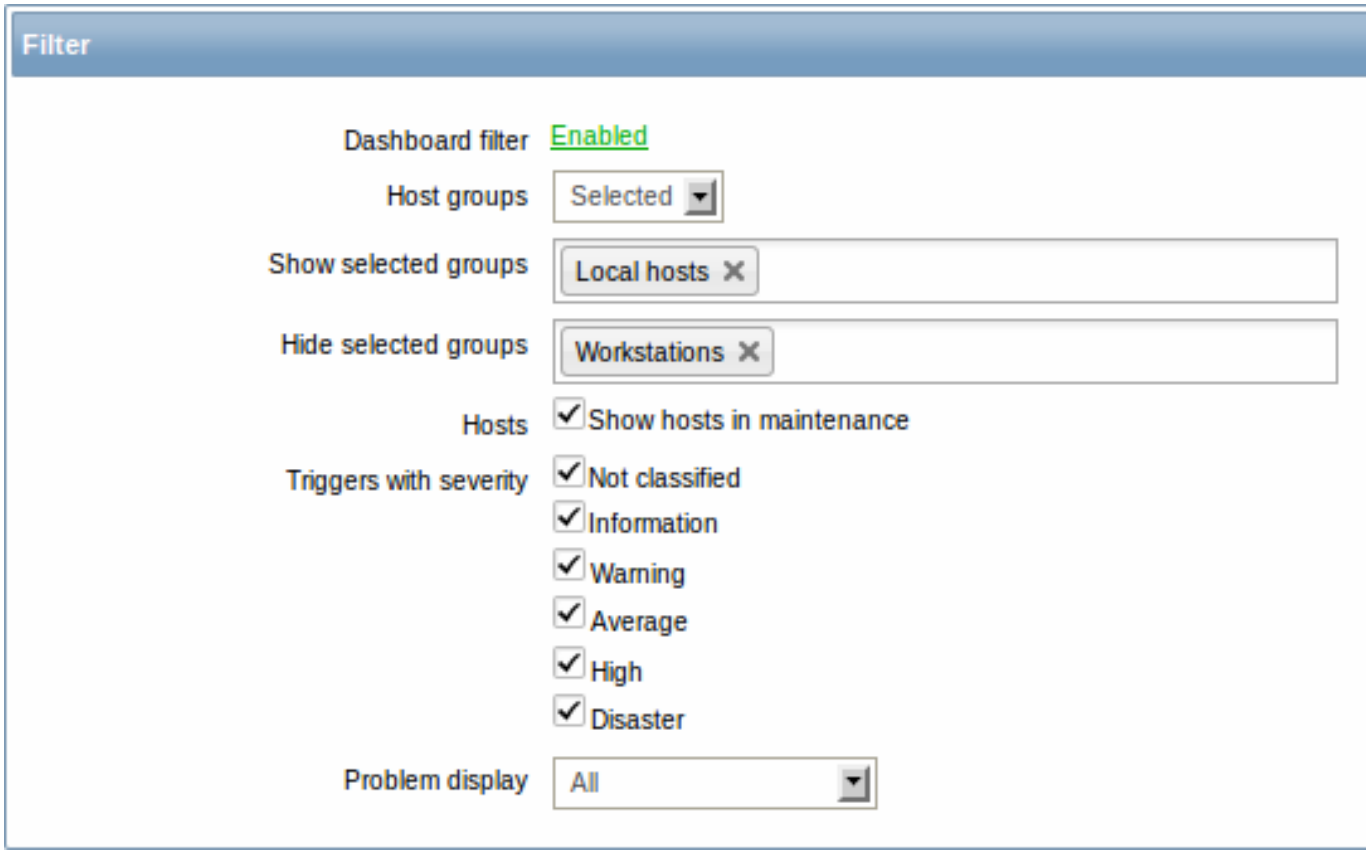
Status widgets

A number of status widgets - *Status of Zabbix, System status, Host status, Last 20 issues, Web monitoring, Discovery status* each display a summary of the respective data.

As you may have noticed from the screenshot, the widgets can be arranged in up to three columns. Additionally, all widgets can be freely moved around. Just grab a widget by its title bar, drag and drop wherever you would like it.

Dashboard filter

Clicking on in the *Personal dashboard* title bar allows you to access the dashboard filter.



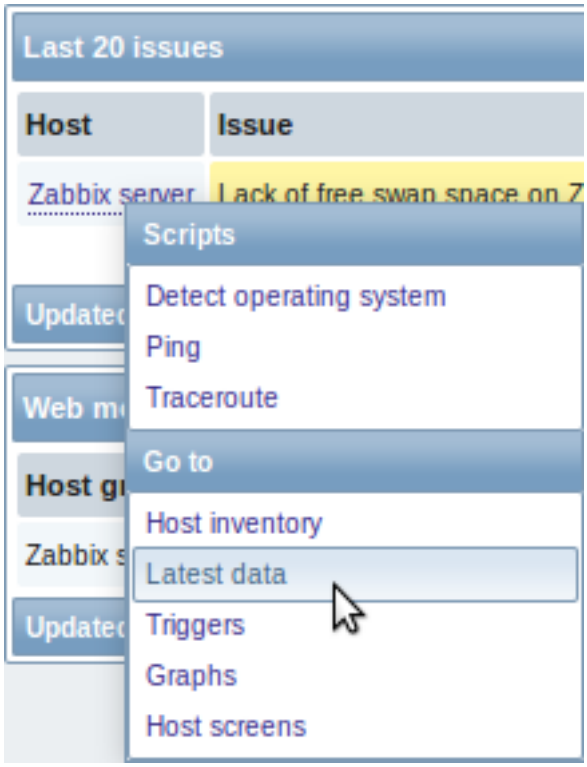
By enabling the filter you can limit what hosts and triggers displayed in the dashboard and define how the problem count is displayed.

Parameter	Description
Dashboard filter	Click the link to enable/disable the dashboard filter.
Host groups	Select to display host data from: **All** - all host groups **Selected** - selected host groups.
Show selected groups	This field is available if *Selected* is chosen in the *Host groups* field. Enter host groups to display. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Host data from these host groups will be displayed in the Dashboard. If no host groups are entered, all host groups will be displayed.
Hide selected groups	This field is available if *Selected* is chosen in the *Host groups* field. Enter host groups to hide. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Host data from these host groups will not be displayed in the Dashboard. 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 *hide* Group B at the same time, only data from host 001 will be displayed in the Dashboard.
Hosts	Mark the *Show hosts in maintenance* option to display data from hosts in maintenance in the Dashboard.
Triggers with severity	Mark the trigger severities to be displayed in the Dashboard.
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.

If dashboard filtering is applied, it is indicated by an orange filter icon: .

Host menu

Clicking on a host in the *Last 20 issues* widget brings up the host menu. It includes links to custom scripts, latest data, triggers, inventory, graphs and screens for the host.



The host menu is accessible by clicking on a host in several other frontend sections:

- Monitoring → Overview (on *Hosts: left*)
- Monitoring → Latest data
- Monitoring → Triggers
- Monitoring → Events
- Monitoring → Events → Event details
- Monitoring → Screens (in *Host issues* and *Host group issues* widgets)
- Monitoring → Maps
- Reports → Triggers top 100

2 Overview

Overview

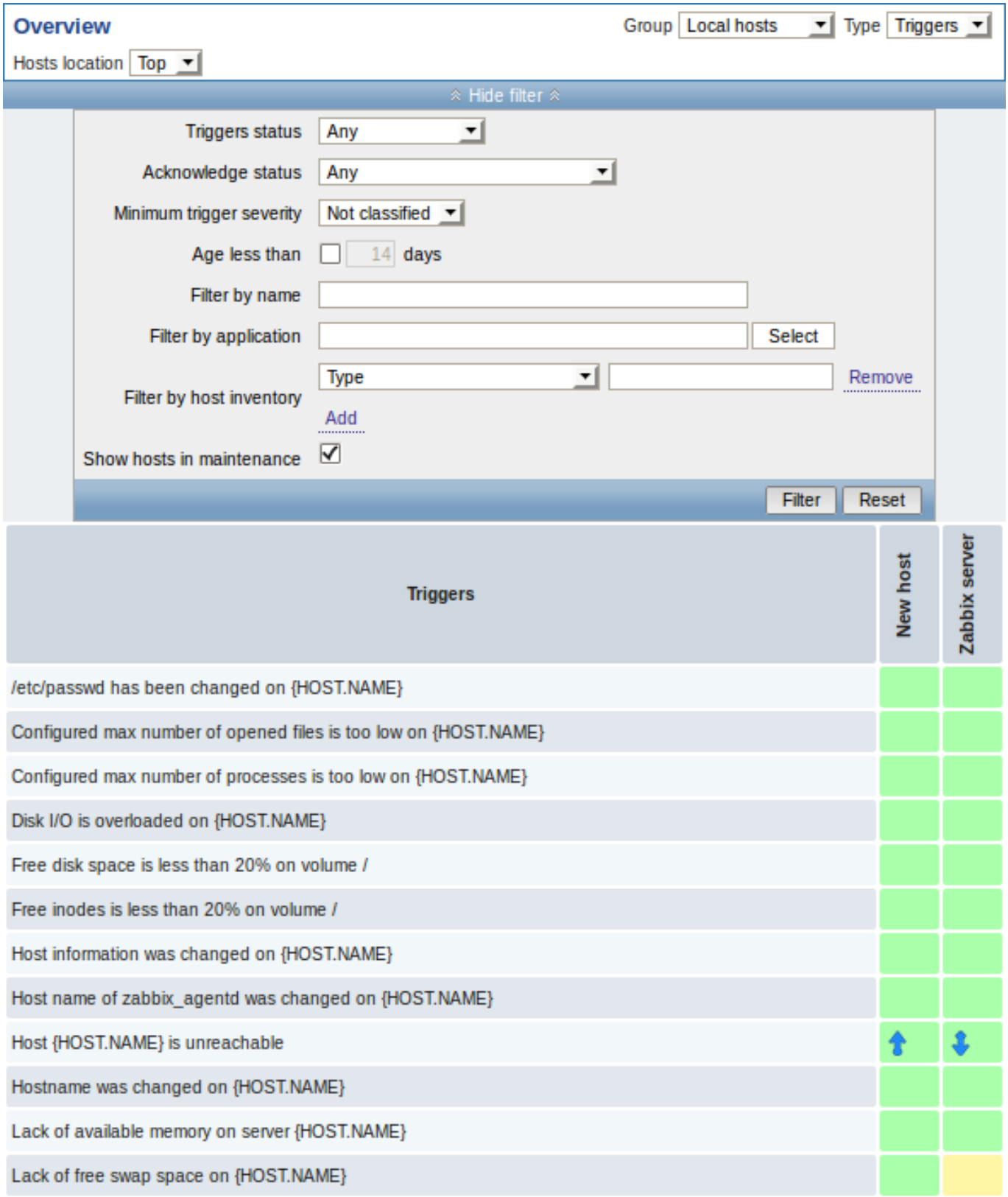
The *Monitoring → Overview* section offers an overview of trigger states or a comparison of data for various hosts at once.

The following display options are available:

- select horizontal or vertical display of information in the *Hosts location* dropdown
- select all hosts or specific host groups in the *Group* dropdown
- choose what type of information to display (triggers or data) in the *Type* dropdown

Overview of triggers

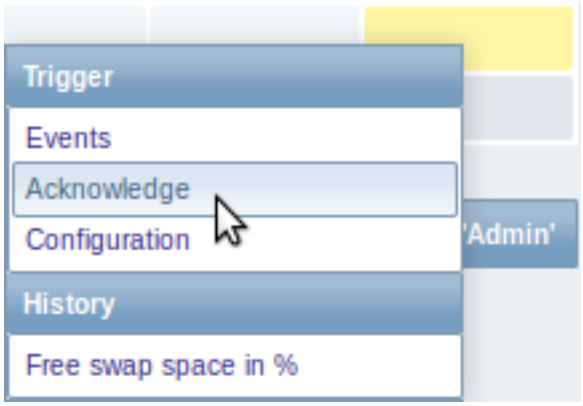
In the next screenshot Triggers are selected in the *Type* dropdown. As a result, trigger states of two local hosts are displayed, as coloured blocks (the colour depending on the state of the trigger):



Note that recent trigger changes (within the last 30 minutes) will be displayed as blinking blocks.

Blue up and down arrows indicate triggers that have dependencies. On mouseover, dependency details are revealed.

Clicking on a trigger block provides links to trigger events, configuration, the acknowledgement screen or a simple graph/latest values list.



Overview of data

In the next screenshot Data is selected in the *Type* dropdown. As a result, performance item data of two local hosts are displayed.

Overview Group Local hosts Type Data

Hosts location Top

Hide filter

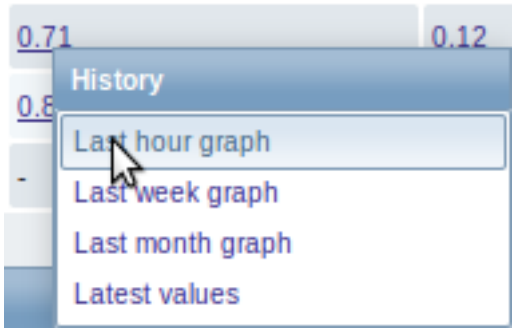
Filter by application Performance Select

Filter Reset

Items	New host	Zabbix server
Context switches per second	2.66 Ksps	109 sps
CPU idle time	0 %	76.11 %
CPU interrupt time	0 %	0 %
CPU iowait time	0 %	1.31 %
CPU nice time	0.0083 %	0 %
CPU softirq time	0.04 %	0.98 %
CPU steal time	0 %	0 %
CPU system time	87.04 %	9.31 %
CPU user time	12.84 %	12.47 %
Interrupts per second	3.42 Kips	67 ips
Processor load (1 min average per core)	3.51	0.72
Processor load (5 min average per core)	2.06	0.48
Processor load (15 min average per core)	1.04	0.26
Processor load - message test (15 min average per core)	-	-

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 of data in large pages. 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.



3 Web

Overview

In the *Monitoring* → *Web* section current information about web scenarios is displayed.

Web checks Group all Host all

Displaying **1** to **2** of **2** found

Host	Name	Number of steps	Last check	Status
New host	Availability of google	2	08 Aug 2013 09:40:19	OK
Zabbix server	Availability of zabbix	4	08 Aug 2013 09:40:30	OK

Note: The name of a disabled host is displayed in red (in both the host dropdown and the list). Data of disabled hosts is accessible starting with Zabbix 2.2.0.

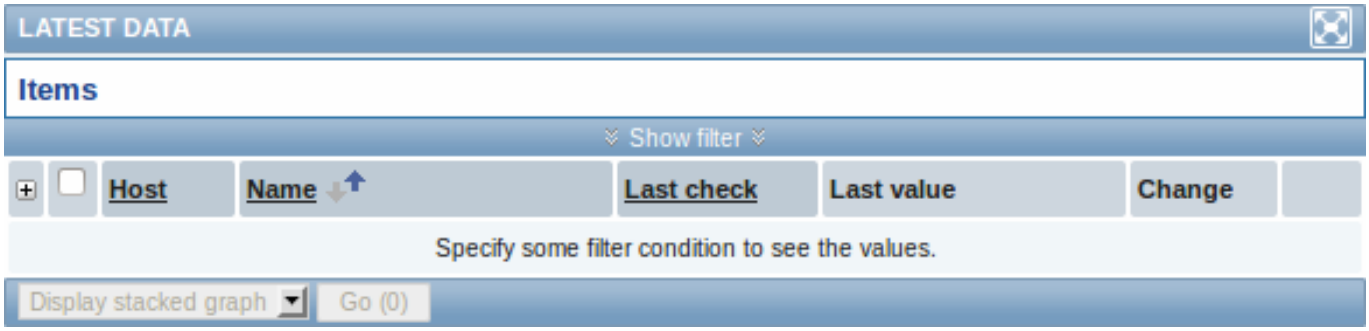
The scenario name is link to more detailed statistics about it:

4 Latest data

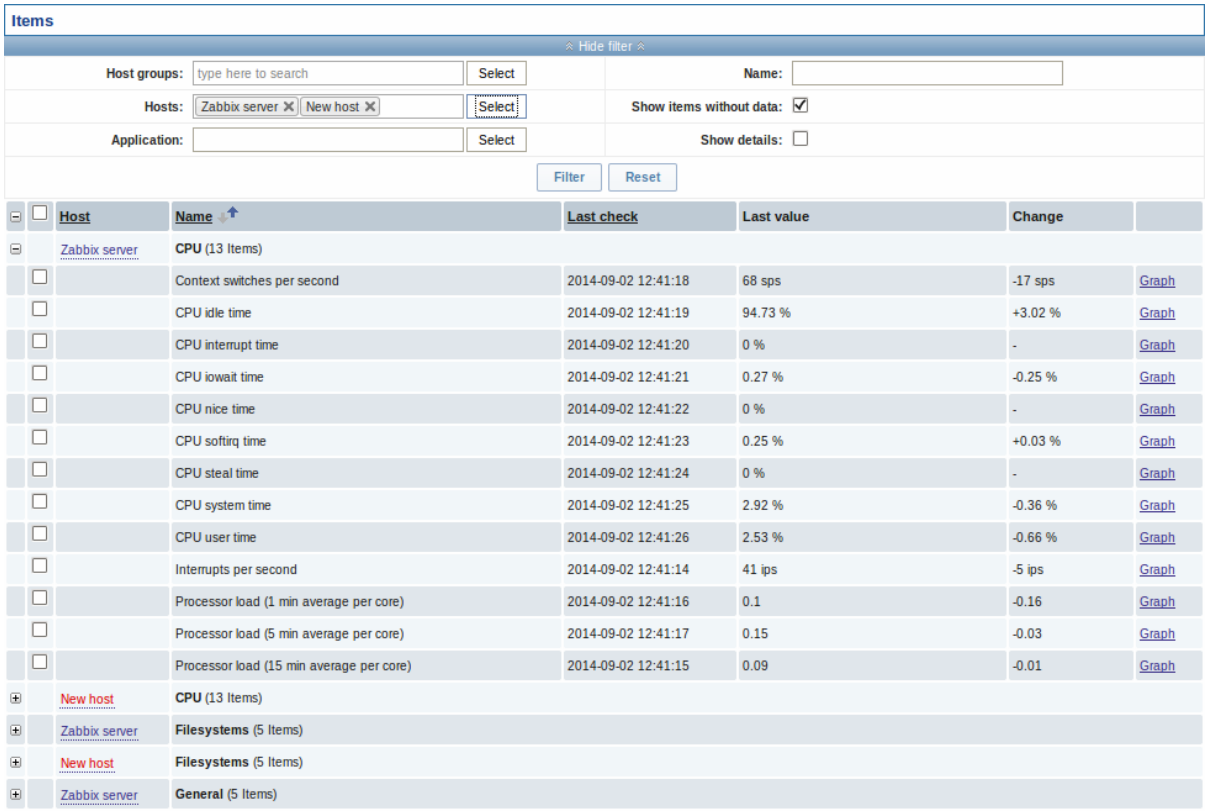
Overview

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.

When you open this page for the first time, nothing is displayed.



To access data, you need to make selections in the filter. For that, you click on *Show filter* and select host group, host, application or item name.



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: The name of a disabled host is displayed in red. Data of disabled hosts, including graphs and item value lists, is accessible in *Latest data* since Zabbix 2.2.0.

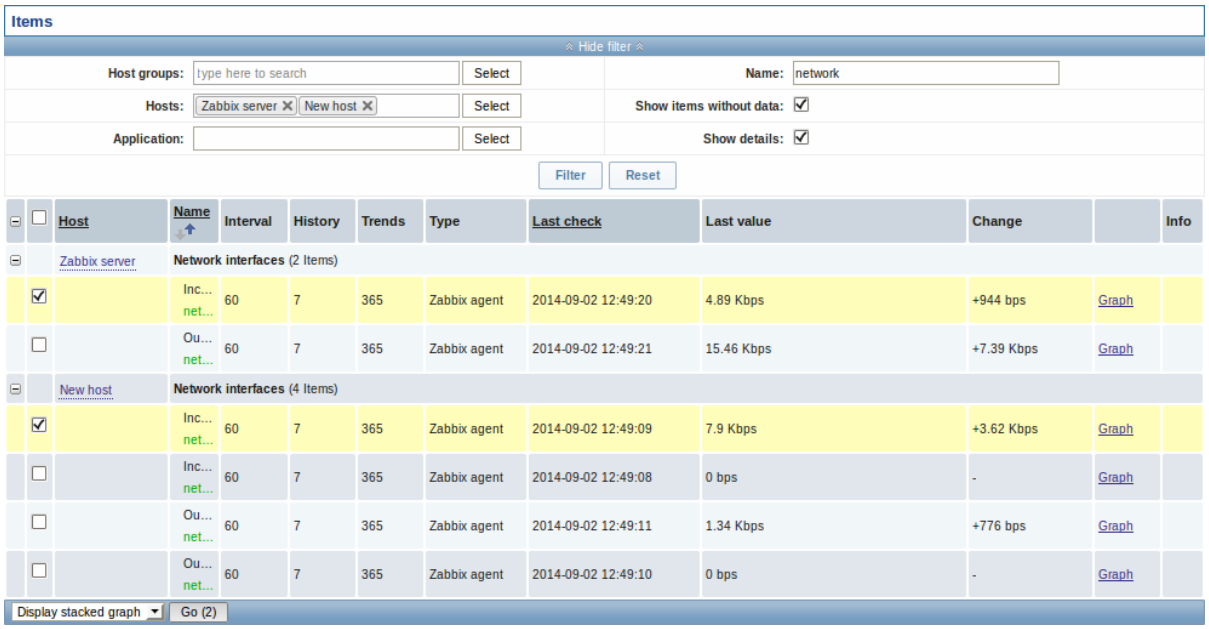
Items are displayed with their name, last check time, last value, change amount and a link to a simple graph/history of item values.

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*.

Using filter

You can use the filter to display only the items you are interested in. The *Show filter* link is located above the table in the middle. 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.

Moreover, *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.

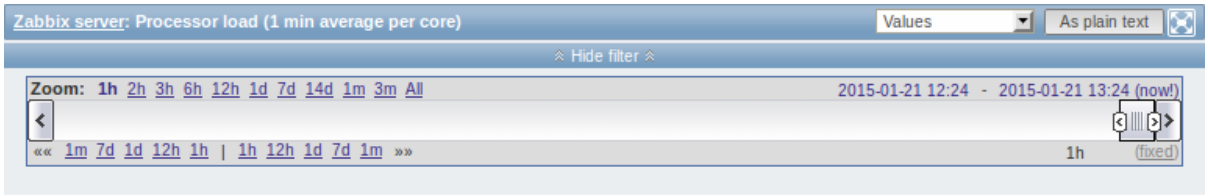
Graphs for comparing items

You may use the checkbox in the second column to select several items and then compare their data in a simple graph or stacked graph. To do that, select items of interest, then select graph type in the dropdown below the table and click on *Go*.

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.



Timestamp	Value
2015-01-21 13:24:16	0.03
2015-01-21 13:23:16	0.09
2015-01-21 13:22:16	0.03
2015-01-21 13:21:16	0.07
2015-01-21 13:20:16	0.1
2015-01-21 13:19:16	0.04
2015-01-21 13:18:16	0.1
2015-01-21 13:17:16	0.27
2015-01-21 13:16:16	0.41
2015-01-21 13:15:16	0.72
2015-01-21 13:14:16	1.97
2015-01-21 13:13:16	0.31
2015-01-21 13:12:16	0.05

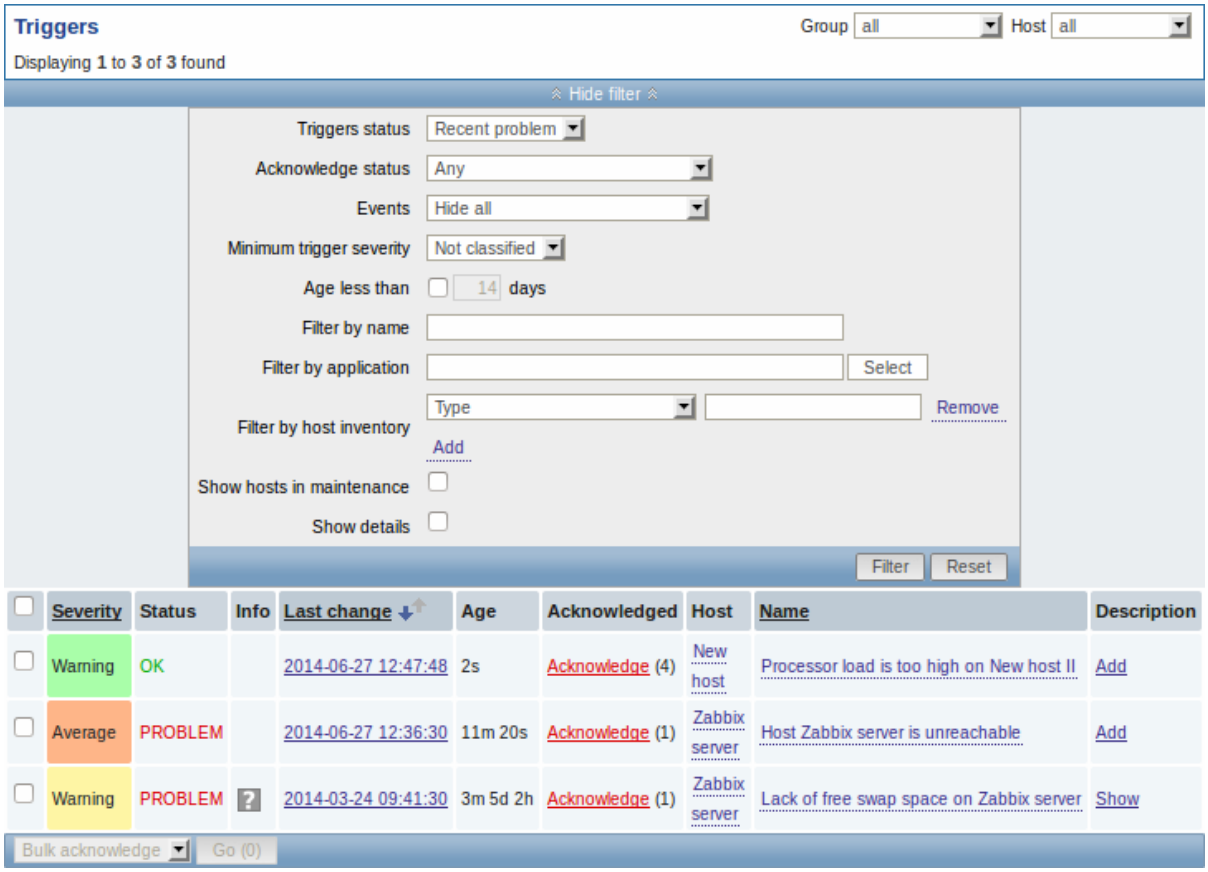
The values displayed in this list are ”raw”, that is, no postprocessing is applied.

Note:
The total amount of values displayed is defined by the value of *Search/Filter elements limit* parameter, set in Administration → General.

5 Triggers

Overview

The *Monitoring* → *Triggers* section displays the status of triggers.



Column	Description
Severity	The trigger severity is displayed. The color of the severity is used as cell background for problem triggers. For OK triggers, green background is used.
Status	The trigger status is displayed - OK or PROBLEM. By default, it will be blinking for 30 minutes for triggers that have recently changed their state. Additionally, triggers that have recently changed their state to OK, will be displayed for 30 minutes even if the filter is set to show only problems. Text color and blinking options can be configured in *Administration* → *General* → *Trigger displaying options*.
Info	A grey icon with a question mark indicates that there is some relevant information to be displayed. If you move your mouse pointer over it, the message will be displayed.
Last change	The date and time of last trigger status change is displayed.
Age	The age of last trigger status change is displayed.
Acknowledged	The acknowledgment status of the trigger is displayed: **Acknowledged** - green text indicating that the trigger is acknowledged. A trigger is considered to be acknowledged if all problem events for it are acknowledged (or if there have been only OK events). **Acknowledge** - a red link indicating unacknowledged problem events (and their number in parenthesis). If you click on the link you will be taken to a bulk acknowledgement screen where all events for this trigger can be acknowledged at once. *Note*: If you wish to acknowledge a single event only, go to *Monitoring* → *Events*. **No events** - if there have been no problem events for the trigger. Displaying this string is supported since Zabbix 2.0.4; prior to that these triggers were displayed as *Acknowledged*.
Host	The host of the trigger is displayed. It is also a link to the defined custom scripts, latest host data, host inventory overview and host screens.

Column	Description
Name	The name of the trigger is displayed. It is also a link to the trigger event list and the trigger configuration page, as well as to a simple graph of item data. The link list may also contain a custom trigger URL, if one is defined in trigger configuration.
Description	A link to trigger description.

Using filter

You can use the filter to display only the triggers you are interested in. The filter link is the blue bar located above the table.

By default only triggers in a 'Recent problem' status are displayed - including both problem triggers and triggers that only very recently changed to *OK*. You may also select to display triggers in 'Problem' status (only problem triggers) or 'Any'.

Note that if you select 'Any' the next time you return to the page 'Recent problem' will be selected by default, whereas if you select 'Problem' the next time you return 'Problem' will remain selected.

6 Events

Overview

The *Monitoring* → *Events* section displays latest events.

Time	Host	Description	Status	Severity	Duration	Ack	Actions
2015-01-21 11:06:02	New host	Processor load is too high on New host	OK	Warning	1h 33m 36s	No	-
2015-01-21 10:58:02	New host	Processor load is too high on New host	PROBLEM	Warning	8m	No	Ok
2015-01-21 10:52:02	New host	Processor load is too high on New host	OK	Warning	6m	No	-
2015-01-21 10:36:35	Zabbix server	Zabbix discoverer processes more than 75% busy	OK	Average	2h 3m 3s	No	-
2015-01-21 10:29:02	New host	Processor load is too high on New host	PROBLEM	Warning	23m	No	Ok
2015-01-21 10:27:35	Zabbix server	Zabbix discoverer processes more than 75% busy	PROBLEM	Average	9m	No	Ok
2015-01-21 10:20:02	New host	Processor load is too high on New host	OK	Warning	9m	No	-
2015-01-21 10:11:02	New host	Processor load is too high on New host	PROBLEM	Warning	9m	No	2 1
2015-01-21 09:59:52	HP Procurve switch	Operational status was changed on HP Procurve switch interface 20	OK	Information	2h 39m 46s	No	-
2015-01-21 09:58:02	New host	Processor load is too high on New host	OK	Warning	13m	No	-
2015-01-21 09:57:52	HP Procurve switch	Operational status was changed on HP Procurve switch interface 20	PROBLEM	Information	2m	No	Ok
2015-01-21 09:50:02	New host	Processor load is too high on New host	PROBLEM	Warning	8m	No	Ok

In the last dropdown to the right you can select trigger or discovery based events.

Using filter

You can use the filter to display events of a specific trigger. The filter link is located above the table.

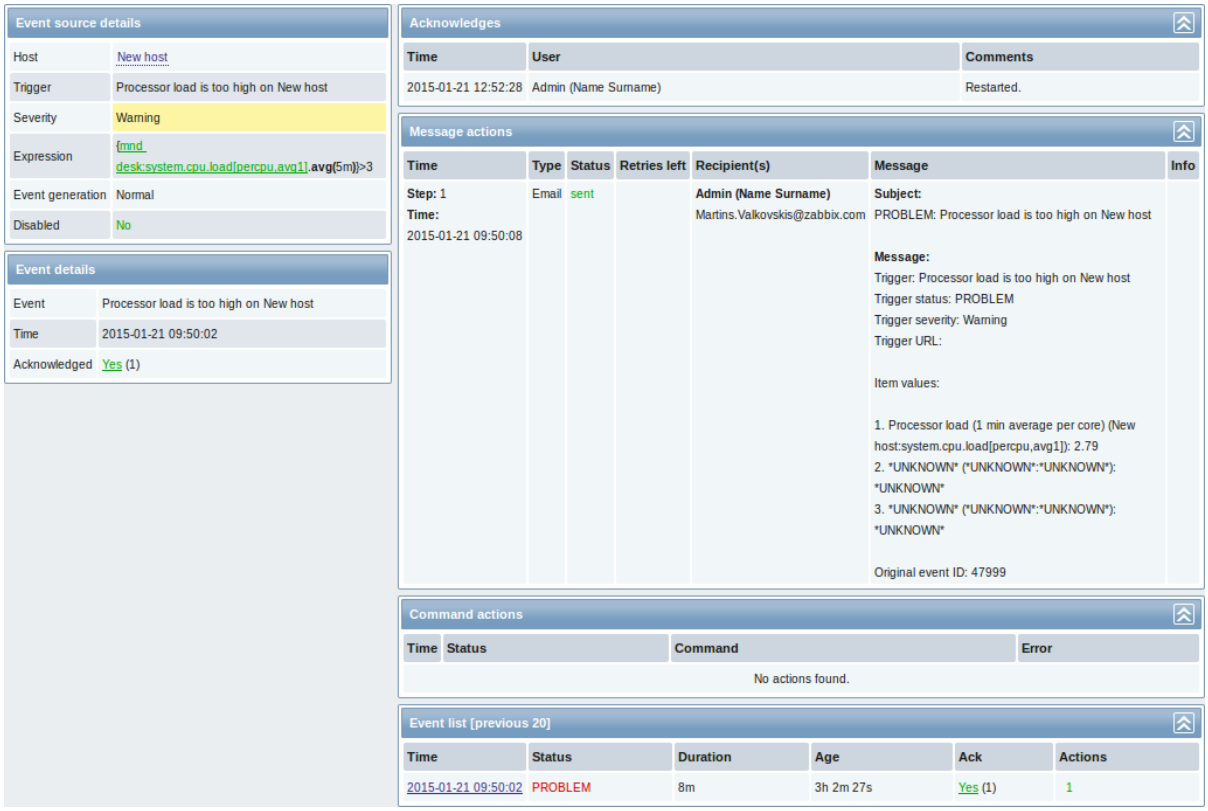
Access to events of a specific trigger is also possible by specifying page parameters in the URL.

Export

Currently displayed events can be exported to a CSV file. Look for the *Export to CSV* button to the right on the title bar.

Event details

Clicking on the timestamp in the first column of trigger events will take you to event details.

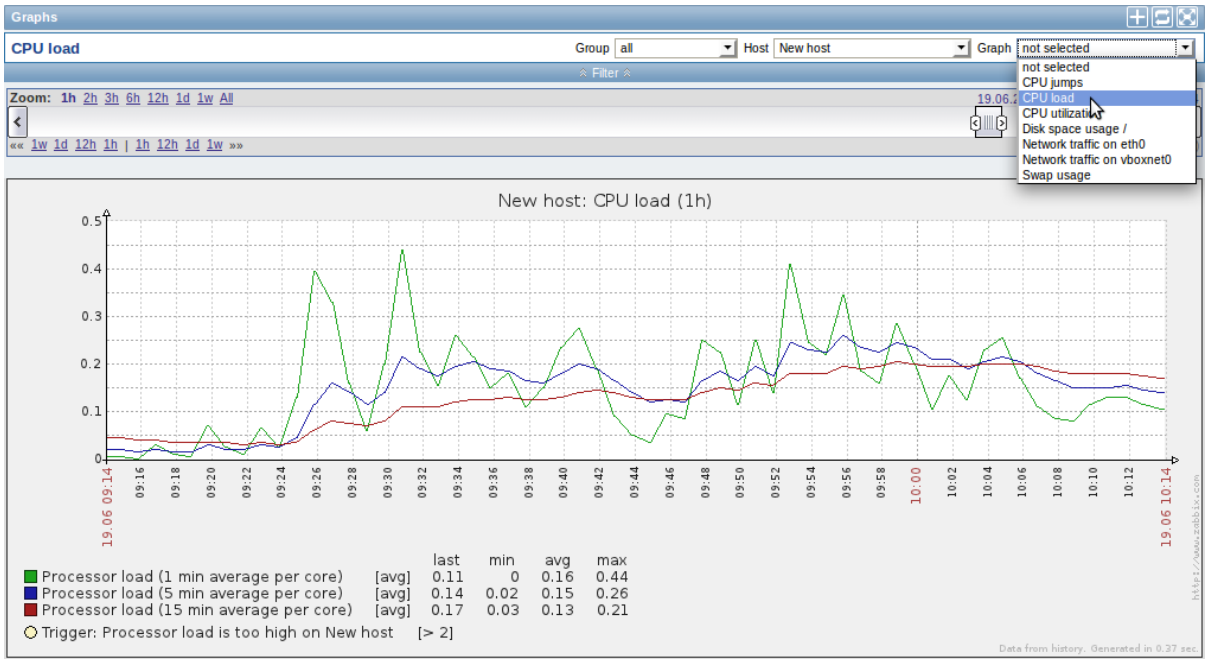


Details about the event, its source, acknowledgments, actions taken (messages, remote commands) and previous similar events are displayed.

7 Graphs

Overview

In the *Monitoring* → *Graphs* section any custom graph that has been configured can be displayed.



To display a graph, select the host group, host and then the graph from the dropdowns to the right.

Note: In the host dropdown, a disabled host is highlighted in red. Graphs for disabled hosts are accessible starting with Zabbix 2.2.0.

Time period selector

The filter section above the graph contains a time period selector. It allows you to select the desired time period easily.

The slider within the selector can be dragged back and forth, as well as resized, effectively changing the time period displayed. Links on the left hand side allow to choose some often-used predefined periods (above the slider area) and move them back and forth in time (below the slider area). The dates on the right hand side actually work as links, popping up a calendar and allowing to set a specific start/end time.

The **fixed/dynamic** link in the lower right hand corner has the following effects:

- controls whether the time period is kept constant when you change the start/end time in the calendar popup.
- when *fixed*, time moving controls (« 6m 1m 7d 1d 12h 1h | 1h 12h 1d 7d 1m 6m ») will move the slider, while not changing its size, whereas when *dynamic*, the control used will enlarge the slider in the respective direction.
- when *fixed*, pressing the larger **<** and **>** buttons will move the slider, while not changing its size, whereas when *dynamic*, **<** and **>** will enlarge the slider in the respective direction. The slider will move by the amount of its size, so, for example, if it is one month, it will move by a month; whereas the slider will enlarge by 1 day.

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.

Controls

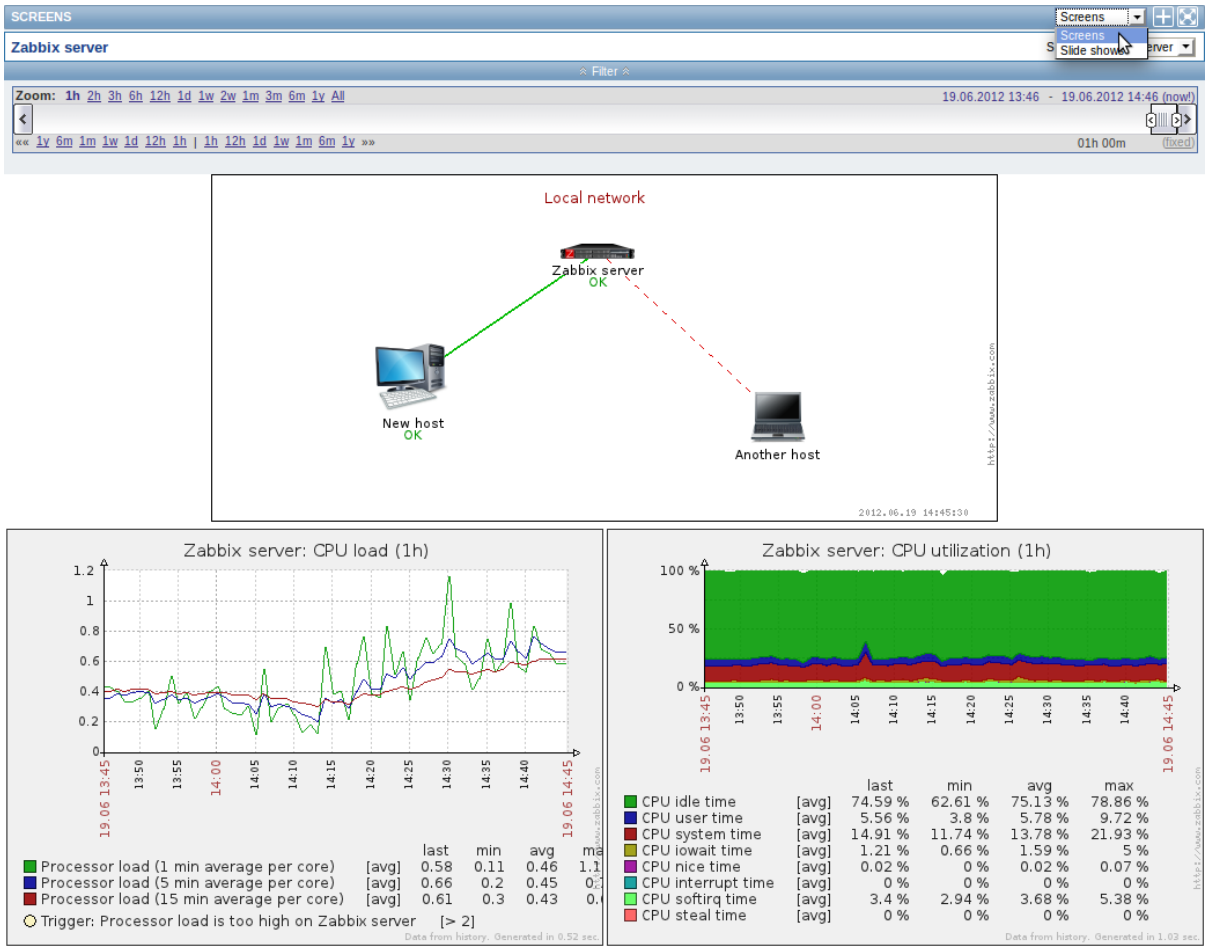
Three control buttons are available in the title bar:

- - add graph to the favourites widget in the Dashboard
- - reset graph display to the original setting of displaying the last hour data
- - use the full browser window to display the graph

8 Screens

Overview

In the *Monitoring* → *Screens* section any configured screen or slide show can be displayed.



Use the dropdown in the title bar to switch between screens and slide shows.

Time period selector

The filter section above the screen/slide show contains a time period selector. It allows you to select the desired time period easily, affecting the data displayed in graphs etc.

Controls

Three control buttons are available in the title bar:

- - add screen/slide show to the favourites widget in the Dashboard
- - use the full browser window to display the screen/slide show
- - slow down or speed up a slide show

Referencing a screen

Screens can be referenced by both `elementid` and `screenname` GET parameters. For example,

`http://zabbix/zabbix/screens.php?screenname=Zabbix%20server`

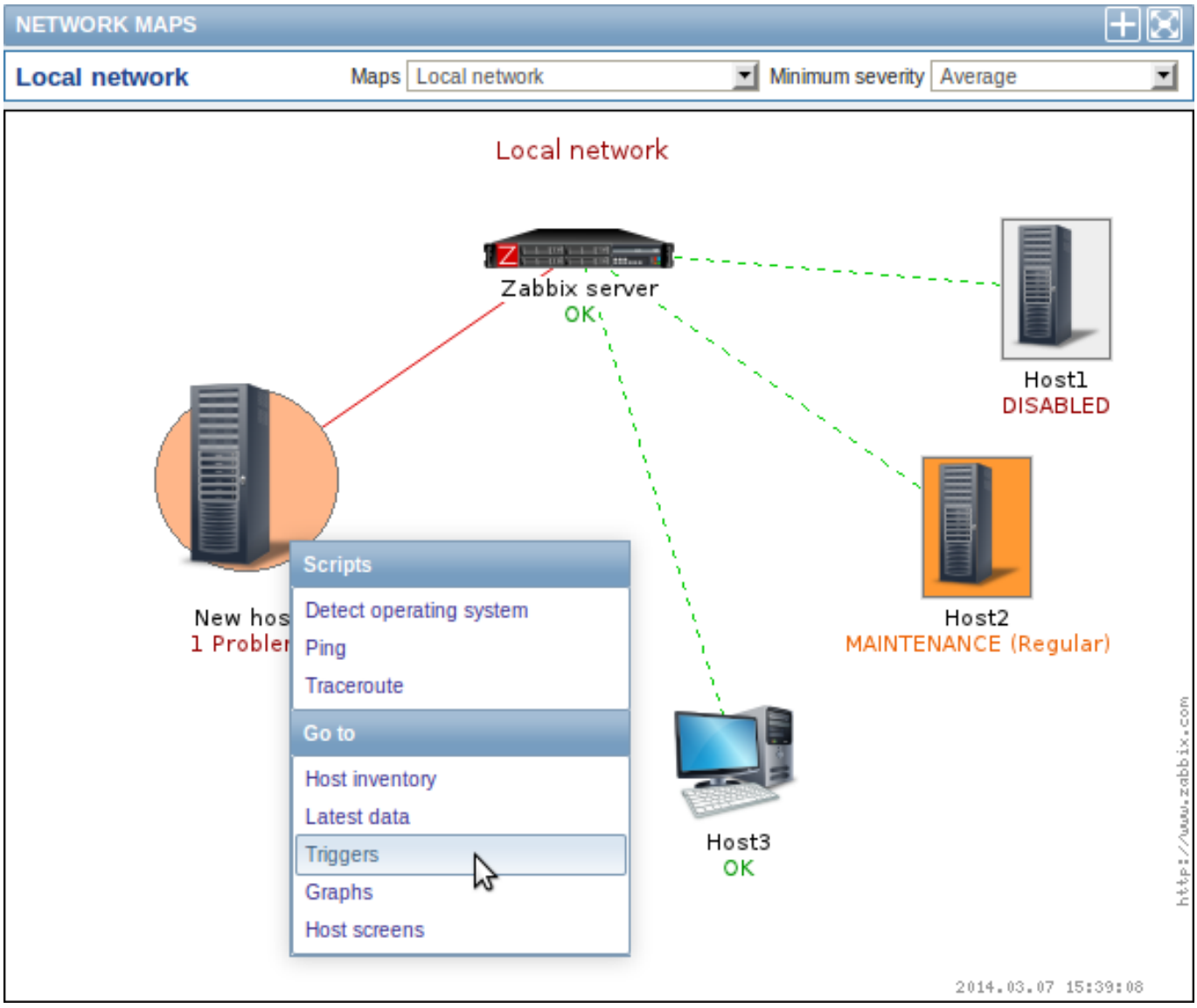
will open the screen with that name (Zabbix server).

If both `elementid` (screen ID) and `screenname` (screen name) are specified, `screenname` has higher priority.

9 Maps

Overview

In the *Monitoring* → *Maps* section any configured network map can be viewed.



You can use the dropdowns in the map title bar to:

- switch between different configured maps
- 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.

Icon highlighting

If a map element is in problem status, it is highlighted with a round circle. The fill colour of the circle corresponds to the severity colour of the problem trigger. 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 and a disabled (not-monitored) host is highlighted with a grey, 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.

Controls

Two control buttons are available in the title bar:

- - add map to the favourites widget in the Dashboard
- - use the full browser window to display the map

Referencing a network map

Network maps can be referenced by both `sysmapid` and `mapname` GET parameters. For example,

`http://zabbix/zabbix/maps.php?mapname=Local%20network`

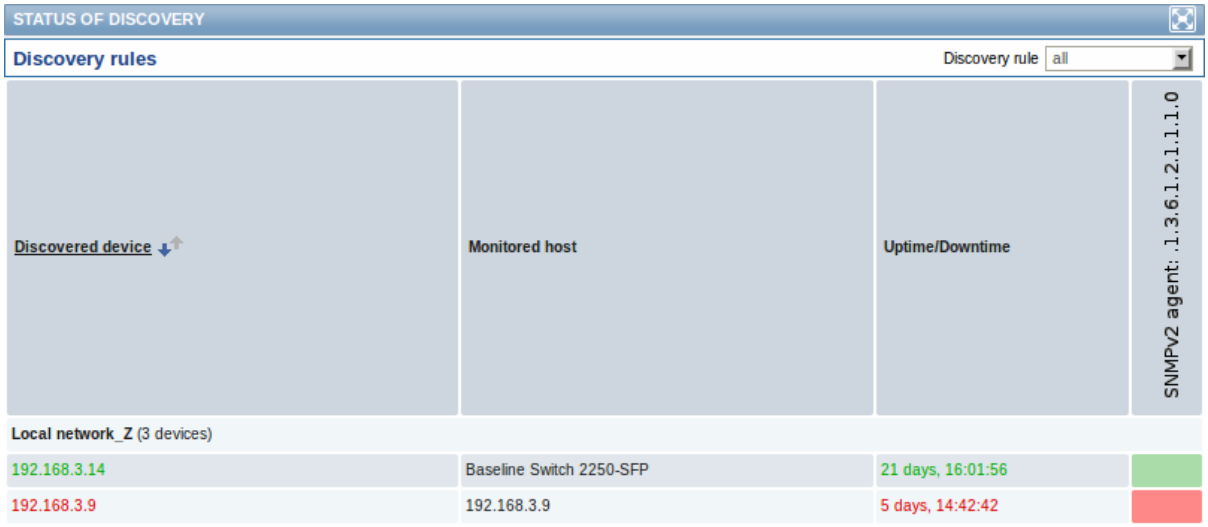
will open the map with that name (Local network).

If both `sysmapid` (map ID) and `mapname` (map name) are specified, `mapname` has higher priority.

10 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.

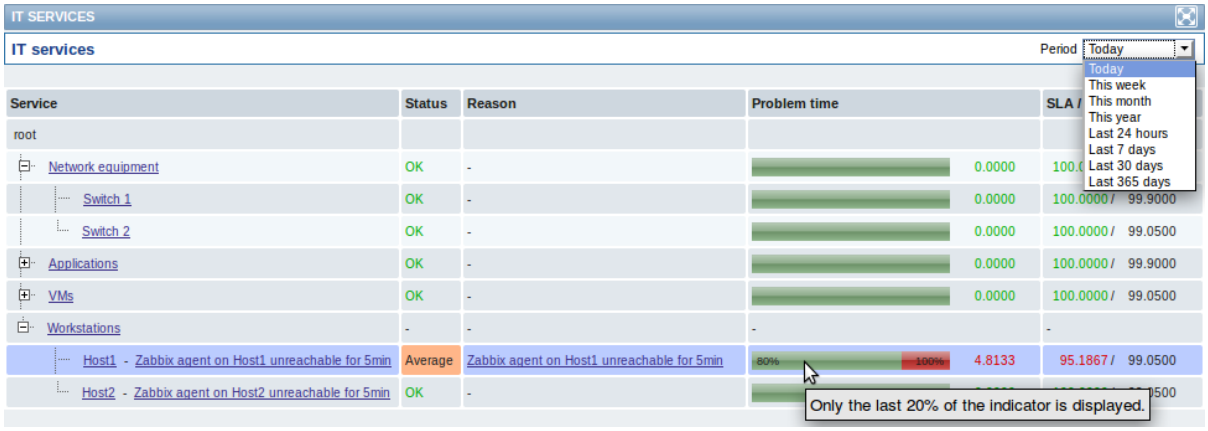
After that follow the columns showing the state of individual services for each discovered device. A tooltip for each cell will show individual service uptime or downtime.

Attention:
Only those services that have been found on at least one device will have a column showing their state.

11 IT services

Overview

In the *Monitoring* → *IT services* section the status of IT services is displayed.

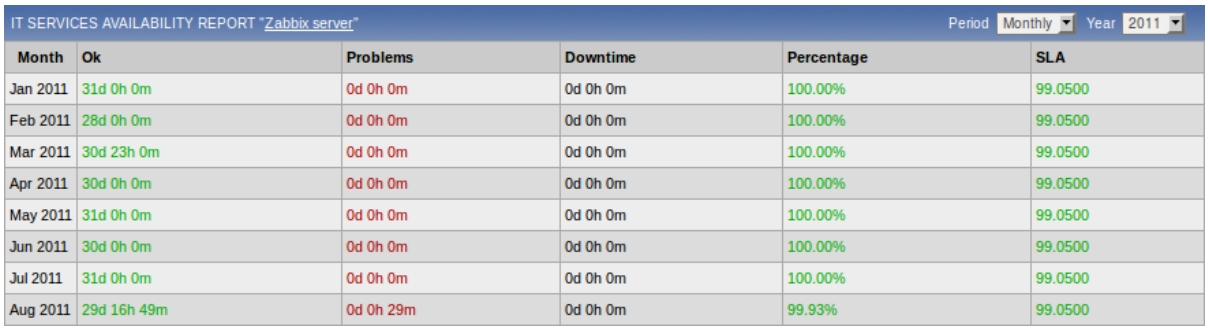


A list of the existing IT services is displayed along with data of their status and SLA. From the dropdown in the upper right corner you can select a desired period for display.

Displayed data:

Parameter	Description
Service	Service name.
Status	Status of service: **OK** - no problems **(trigger colour 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 *IT Services Availability Report*.



Month	Ok	Problems	Downtime	Percentage	SLA
Jan 2011	31d 0h 0m	0d 0h 0m	0d 0h 0m	100.00%	99.0500
Feb 2011	28d 0h 0m	0d 0h 0m	0d 0h 0m	100.00%	99.0500
Mar 2011	30d 23h 0m	0d 0h 0m	0d 0h 0m	100.00%	99.0500
Apr 2011	30d 0h 0m	0d 0h 0m	0d 0h 0m	100.00%	99.0500
May 2011	31d 0h 0m	0d 0h 0m	0d 0h 0m	100.00%	99.0500
Jun 2011	30d 0h 0m	0d 0h 0m	0d 0h 0m	100.00%	99.0500
Jul 2011	31d 0h 0m	0d 0h 0m	0d 0h 0m	100.00%	99.0500
Aug 2011	29d 16h 49m	0d 0h 29m	0d 0h 0m	99.93%	99.0500

Here you can assess IT service availability data over a longer period of time on daily/weekly/monthly/yearly basis.

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

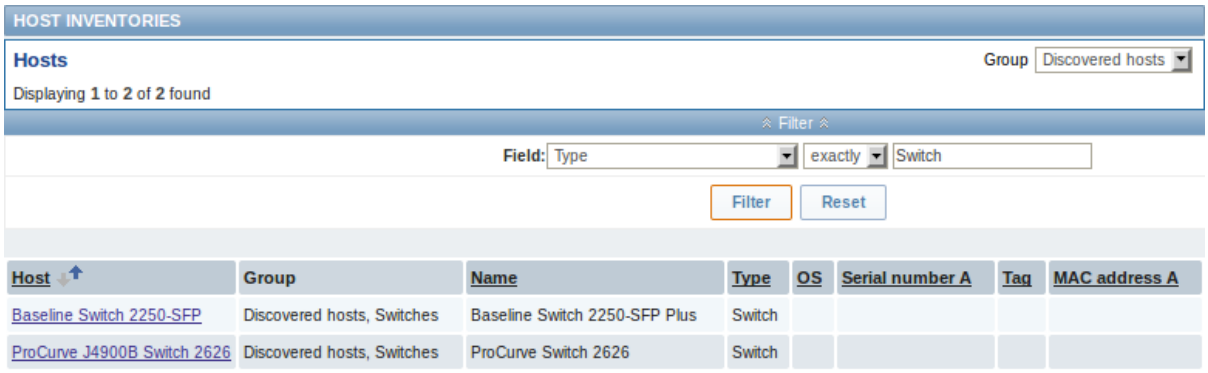
Overview

The *Inventory* → *Overview* section provides ways of having an overview of host inventory data.

For an overview to be displayed, choose a host group (or all groups) 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.

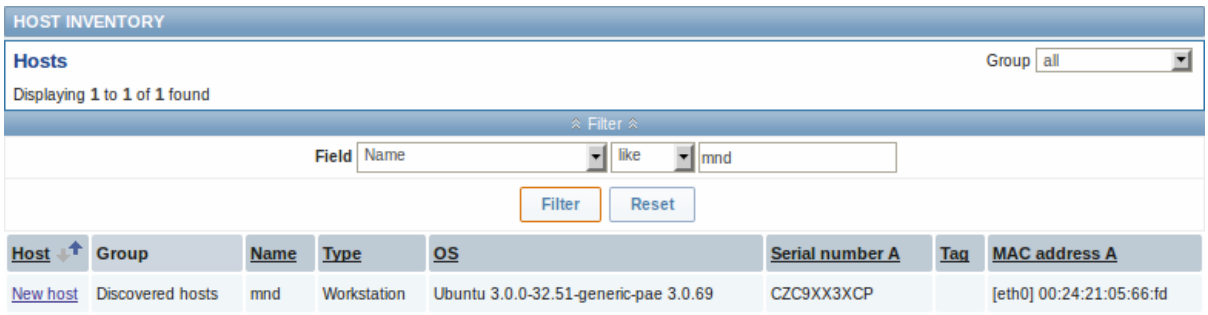


2 Hosts

Overview

In the *Inventory* → *Hosts* section inventory data of hosts are displayed.

Select a group from the dropdown in the upper right corner to display the inventory data of hosts in that group. You can also filter the hosts by any inventory field to display only the hosts you are interested in.

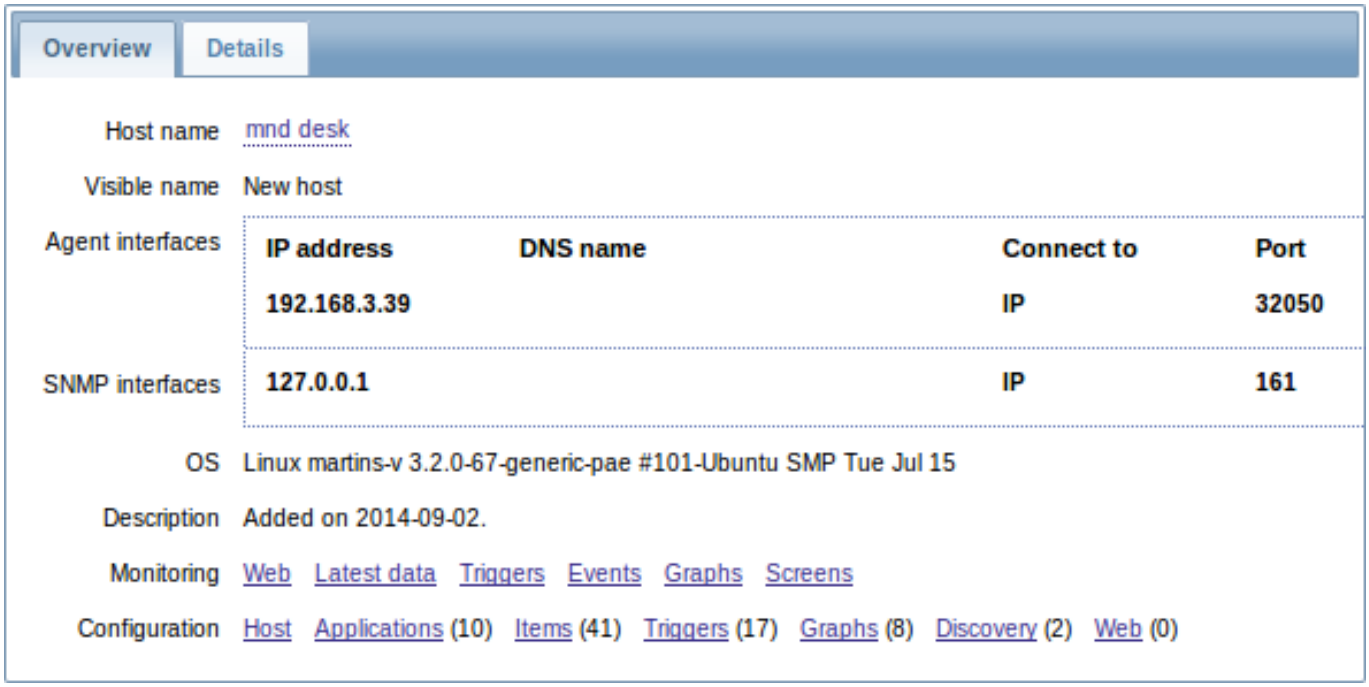


To display all host inventories, select ”all” in the group dropdown, clear the comparison field in the filter and press ”Filter”.

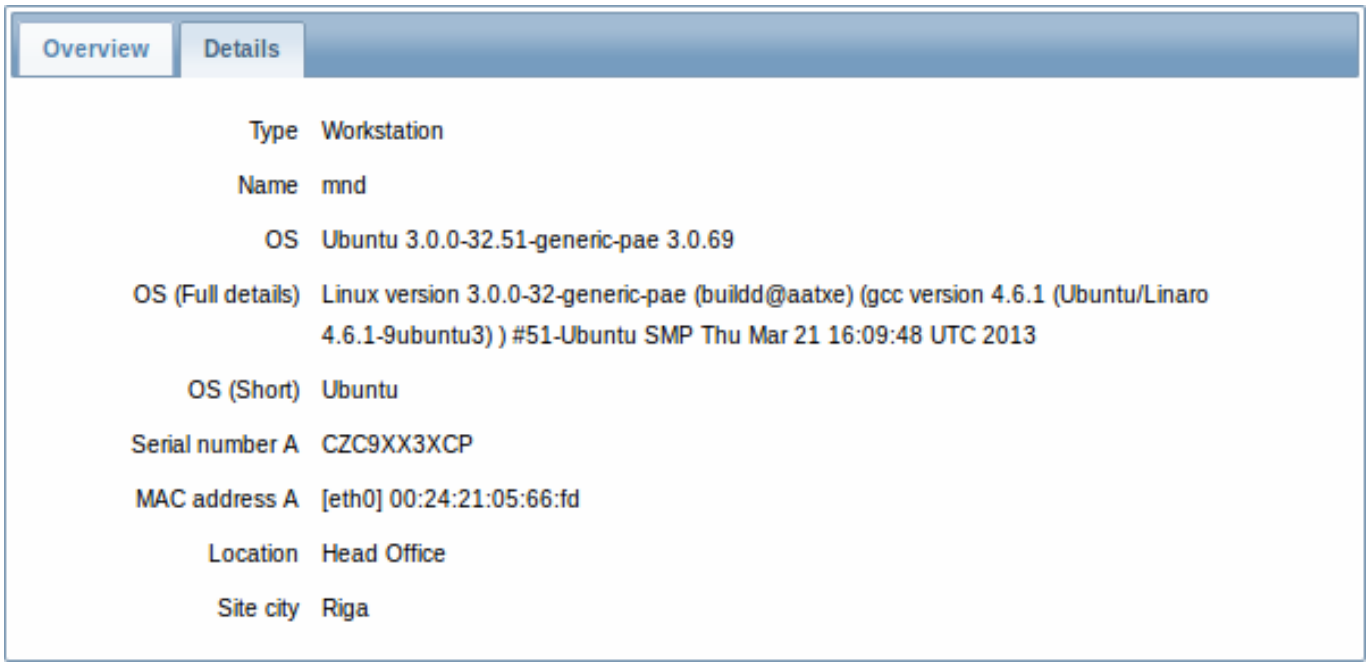
While only some key inventory fields are displayed in the table, you can also view all available inventory information for that host. To do that, click on the host name in the first column.

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 the status of Zabbix, triggers and gathered data.

1 Status of Zabbix

Overview

In *Reports* → *Status of Zabbix* a summary of key system data is displayed.

STATUS OF ZABBIX

Parameter	Value	Details
Zabbix server is running	Yes	localhost:10051
Number of hosts (monitored/not monitored/templates)	36	11 / 0 / 25
Number of items (monitored/disabled/not supported)	864	850 / 9 / 5
Number of triggers (enabled/disabled)[problem/ok]	152	127 / 25 [2 / 125]
Number of users (online)	3	2
Required server performance, new values per second	10.7	-

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 make sure the web frontend knows that the server is running 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. Templates are counted as a type of host too.	Number of monitored hosts/not monitored hosts/templates.
Number of items	Total number of items is displayed. Only items assigned to enabled hosts are counted.	Number of monitored/disabled/unsupported items.
Number of triggers	Total number of triggers is displayed. Only triggers assigned to enabled hosts and depending on enabled items are counted.	Number of enabled/disabled triggers. [Triggers in problem/ok state.]
Number of users	Total number of users configured is displayed.	Number of users online.

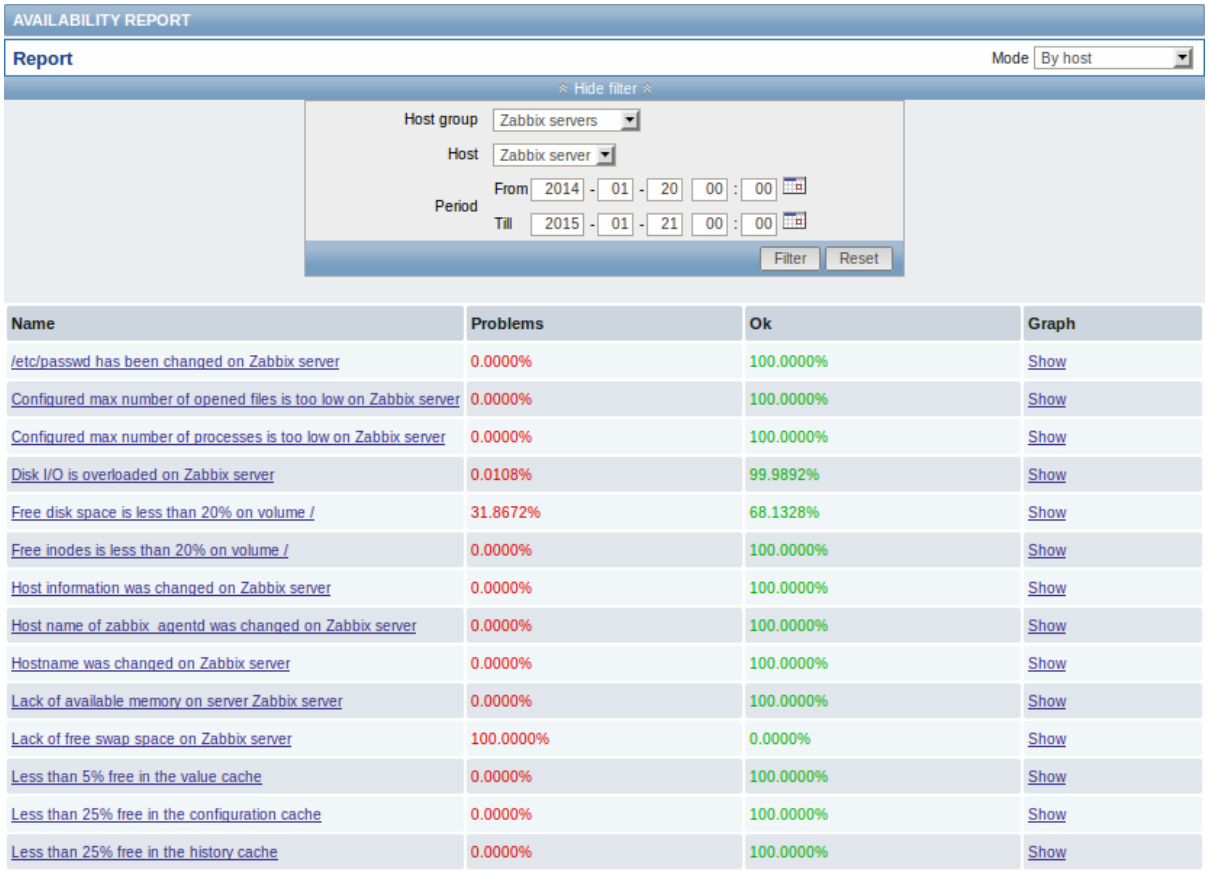
Parameter	Value	Details
Required server performance, new values per second	The expected number of new values processed by Zabbix server per second is displayed.	*Required server performance* is an estimate and can be useful as a guideline. For precise numbers of values processed, use the `zabbix[wcache,values,all]` 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 interval values are not. The calculation is not adjusted during a "nodata" maintenance period. Trapper items are not counted.

2 Availability report

Overview

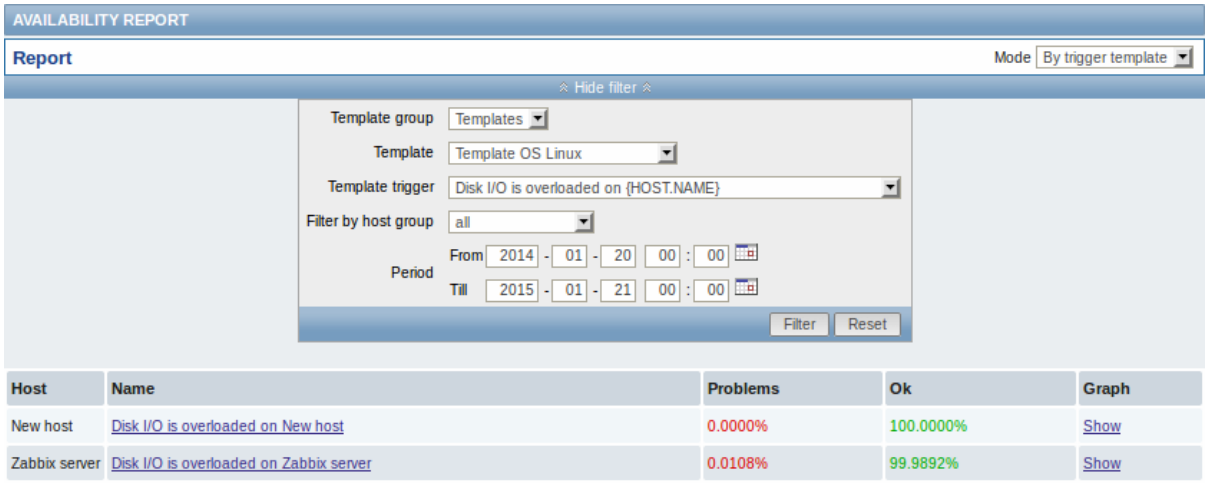
In *Reports → Availability report* you can see what proportion of time each trigger has been in problem/ok state. The percentage of time for each state is displayed.

Thus it is easy to determine the availability situation of various elements on your system.



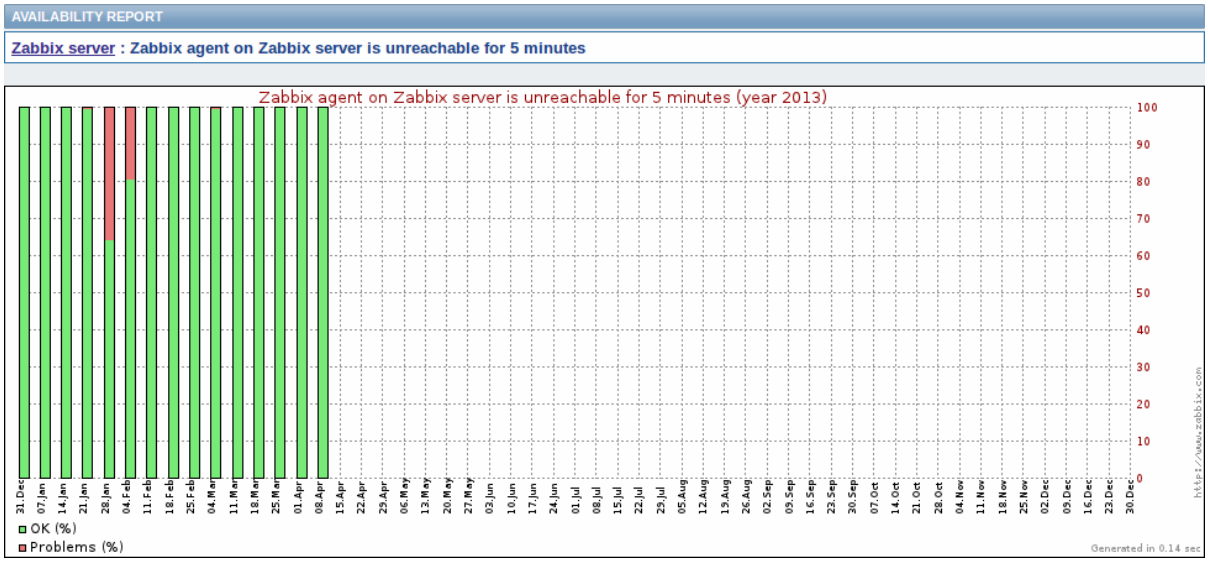
Name	Problems	Ok	Graph
/etc/passwd has been changed on Zabbix server	0.0000%	100.0000%	Show
Configured max number of opened files is too low on Zabbix server	0.0000%	100.0000%	Show
Configured max number of processes is too low on Zabbix server	0.0000%	100.0000%	Show
Disk I/O is overloaded on Zabbix server	0.0108%	99.9892%	Show
Free disk space is less than 20% on volume /	31.8672%	68.1328%	Show
Free inodes is less than 20% on volume /	0.0000%	100.0000%	Show
Host information was changed on Zabbix server	0.0000%	100.0000%	Show
Host name of zabbix_agentd was changed on Zabbix server	0.0000%	100.0000%	Show
Hostname was changed on Zabbix server	0.0000%	100.0000%	Show
Lack of available memory on server Zabbix server	0.0000%	100.0000%	Show
Lack of free swap space on Zabbix server	100.0000%	0.0000%	Show
Less than 5% free in the value cache	0.0000%	100.0000%	Show
Less than 25% free in the configuration cache	0.0000%	100.0000%	Show
Less than 25% free in the history cache	0.0000%	100.0000%	Show

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. Then in the filter you can narrow down the selection to the desired options and the time period.



The name of the trigger is a link to the latest events of that trigger.

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.



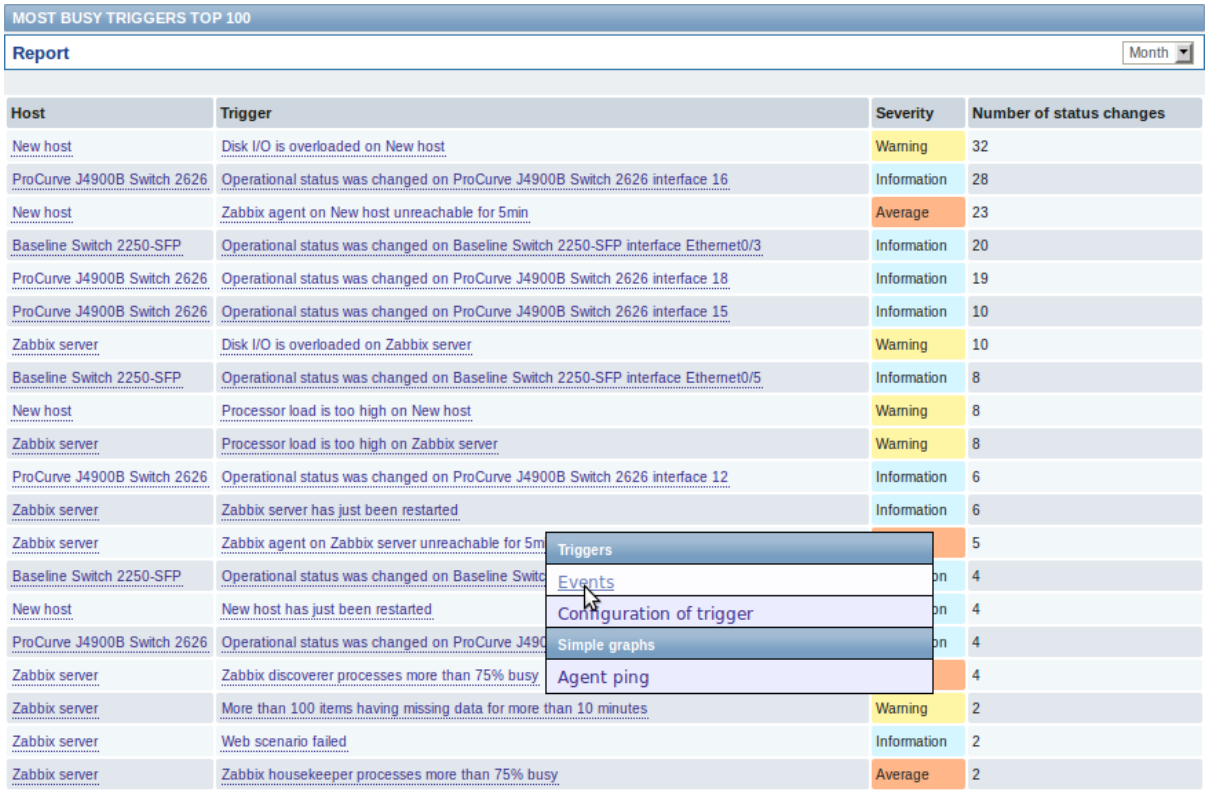
The green part of a bar stands for OK time and red for problem time.

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.

From the dropdown in the upper right corner you can choose the time period for evaluation - day, week, month, year.



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

4 Bar reports

Overview

In the *Reports* → *Bar reports* section you can create some customized bar reports on run-time. Reports can be viewed, but are not saved.

From the dropdown in the upper right corner you can choose one of the three types of available bar reports. Then use the *Filter* options to create the report.

Parameter	Description
Title	Name of the report.
X label	Label displayed below the X axis.
Y label	Label displayed alongside the Y axis.
Legend	With this checkbox marked, a legend will be displayed alongside the report.
Scale	Picking a scale will separate out the value bars for either every hour/day/week/month/year. So, for example, picking a *daily* scale will display one bar for the values of one day. This parameter is available for the first and third report type.
Period	Enter the start and end of the evaluation period. With the second report type, several custom periods, each displayed in different colour, can be entered.
Items	Click on *Add* to select the items whose data you wish to display.

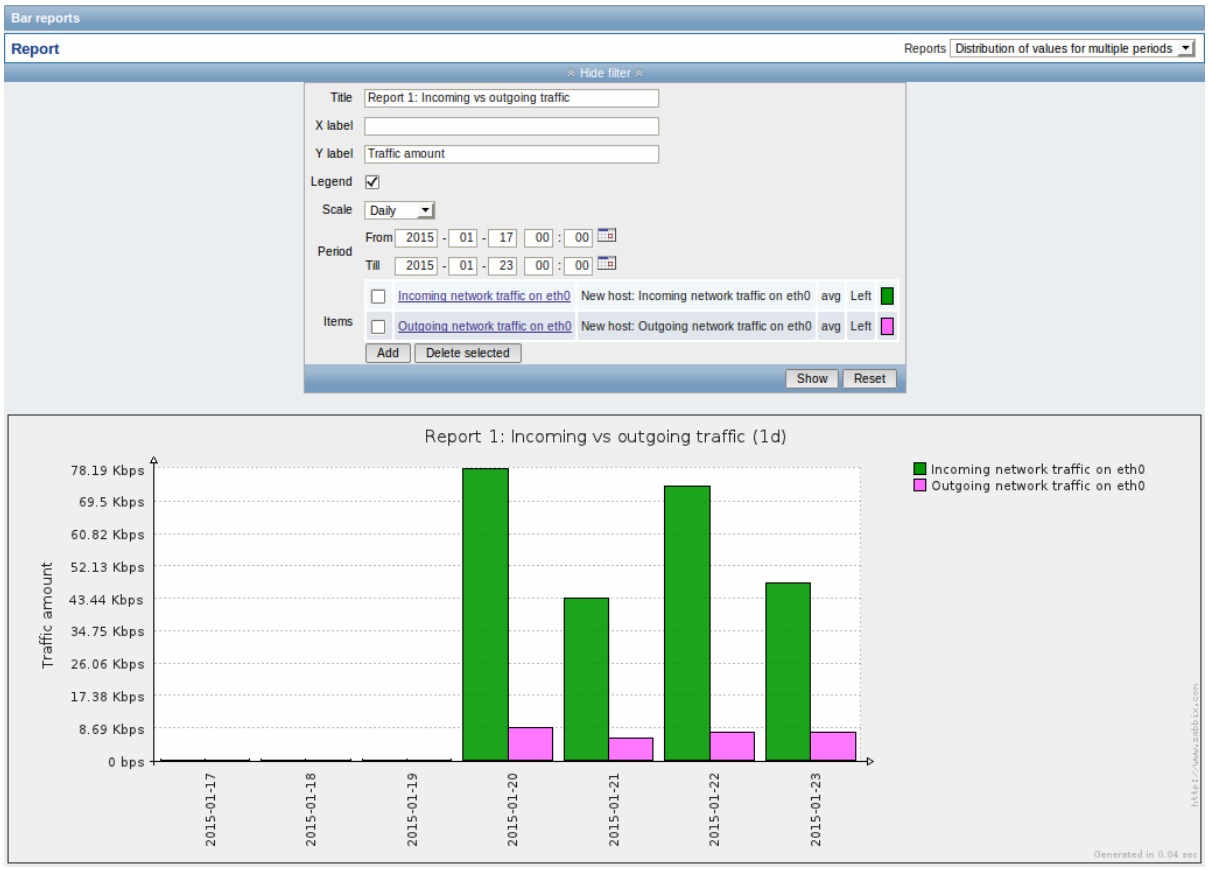
Specifically for the third report type:

Groups	From the *Other groups* pane select host groups. Item values for any host in the group having that item will be displayed.
Hosts	From the *Other hosts* pane select hosts. Item values for any selected host having that item will be displayed.

Average	Select whether to display averaged data for an hour/day/week/month/year.
Item	Select the item whose data you wish to display.
Palette	Pick a palette of colours for displaying side-by-side bars and colour intensity (middle/darken/brighter).

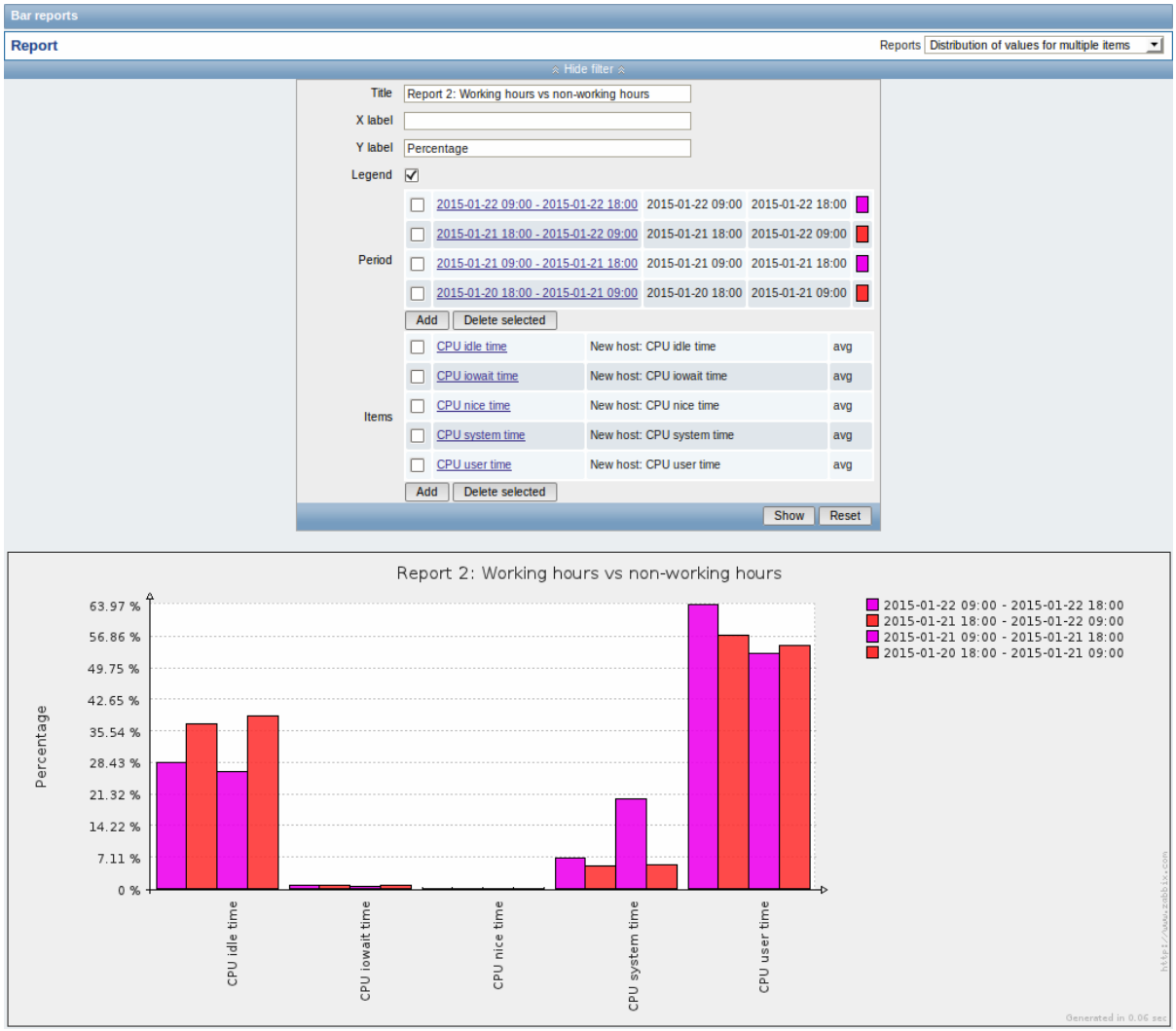
Item data comparison

The first bar report offers a possibility to simply compare item values side by side.



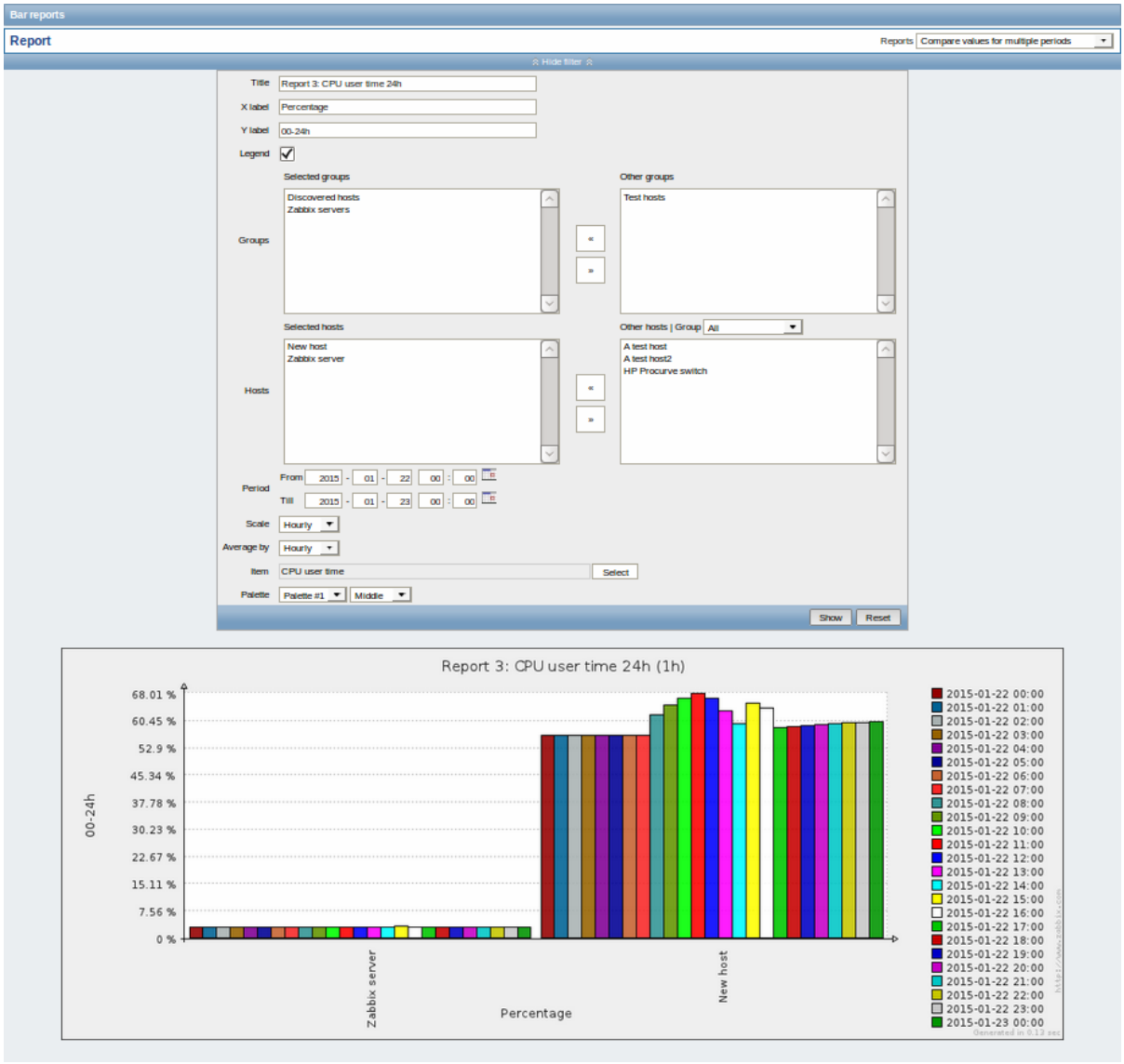
Period data comparison

The second bar report offers a possibility to compare the values of one or several items in custom periods.



One item data comparison

The third bar report offers a possibility to compare the values of one item for different hosts/predefined intervals (hourly/daily/weekly/monthly/y



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 visualisation 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.

Displayed data:

Column	Description
Name	Name of the host group. Clicking on the group name opens the host group configuration form.
#	Number of templates and hosts in the group (displayed in parentheses). Clicking on ”Templates” or ”Hosts” will, in the whole listing of templates or hosts, filter out those that belong to the group.
Members	Names of group members. Template names are displayed in grey, monitored host names in blue and non-monitored host names in red. Clicking on a name will open the template/host configuration form.

Mass editing options

A dropdown below the list offers some mass-editing options:

- *Enable selected* - change the status of all hosts in the group to ”Monitored”
- *Disable selected* - change the status of all hosts in the group to ”Not monitored”
- *Delete selected* - delete the host groups

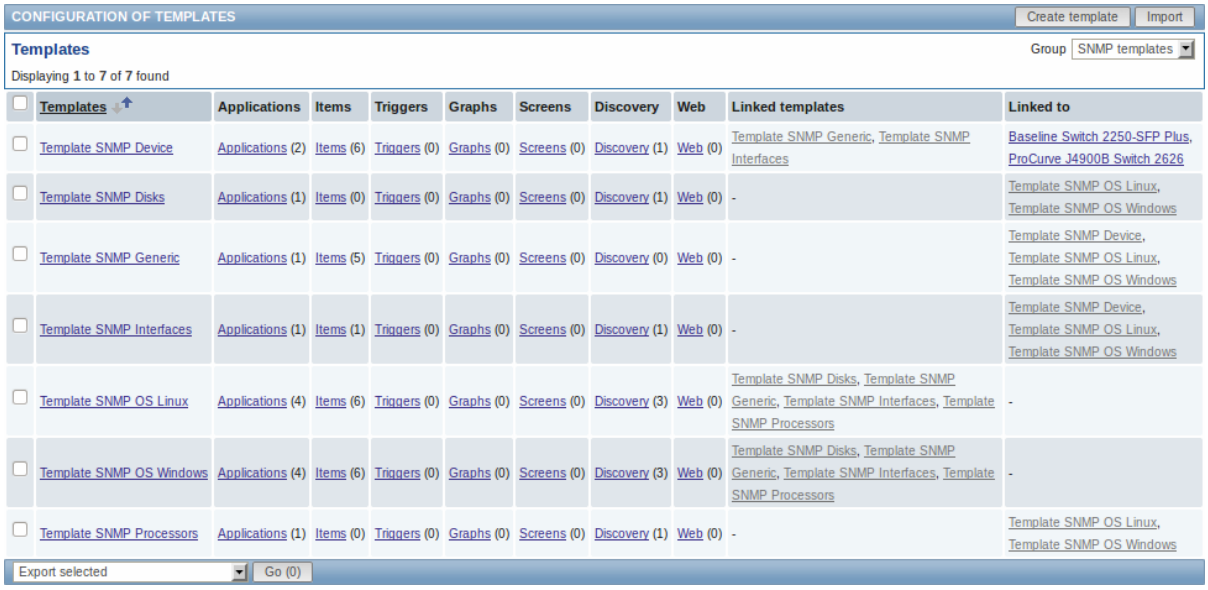
To use these options, mark the check-boxes before the respective host groups, then select the required option and click on ”Go”.

2 Templates

Overview

In the *Configuration* → *Templates* section users can configure and maintain templates.

A listing of existing templates with their details is displayed.



From the dropdown to the right in the title bar you can choose whether to display all templates or only those belonging to a group.

Displayed data:

Column	Description
Templates	Name of the template. Clicking on the template name opens the template configuration form.
Elements (Applications, Items, Triggers, Graphs, Screens, Discovery, Web)	Number of the respective elements in the template (displayed in parentheses). Clicking on the element name will, in the whole listing of that element, 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 elements of the linked templates.
Linked to	The hosts and templates that the template is linked to.

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.

Mass editing options

A dropdown below the list offers some mass-editing options:

- *Export selected* - export the template to an XML file
- *Delete selected* - delete the template while leaving its linked elements (items, triggers etc.) with the hosts
- *Delete selected with linked elements* - delete the template and its linked elements from the hosts

To use these options, mark the check-boxes before the respective templates, then select the required option and click on ”Go”.

3 Hosts

Overview

In the *Configuration* → *Hosts* section users can configure and maintain hosts.

A listing of existing hosts with their details is displayed.

From the dropdown to the right in the *Hosts* bar you can choose whether to display all hosts or only those belonging to one particular group.

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 parentheses.
Interface	The main interface of the host is displayed.
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 - *Monitored* or *Not monitored*. By clicking on the status you can change it.
Availability	Availability of the host is displayed. Four icons each represent a supported interface (Zabbix agent, SNMP, IPMI, JMX). If the interface is configured and available, it is displayed in green. If it is configured and unavailable, it is displayed in red, and, upon mouseover, will display details of why the interface cannot be reached.

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

A dropdown below the list offers some mass-editing options:

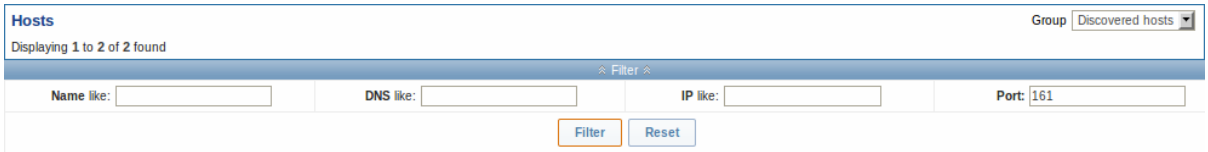
- *Export selected* - export the hosts to an XML file
- *Mass update* - update several properties for a number of hosts at once
- *Enable selected* - change host status to *Monitored*
- *Disable selected* - change host status to *Not monitored*
- *Delete selected* - delete the hosts

To use these options, mark the check-boxes before the respective hosts, then select the required option and click on ”Go”.

Filter

As the list may contain very many hosts, it may be needed to filter out the ones you really need.

The narrow blue bar just below the *Hosts* bar is actually a link to the filter. If you click on it, a filter becomes available where you can filter hosts by name, DNS, IP or port number.



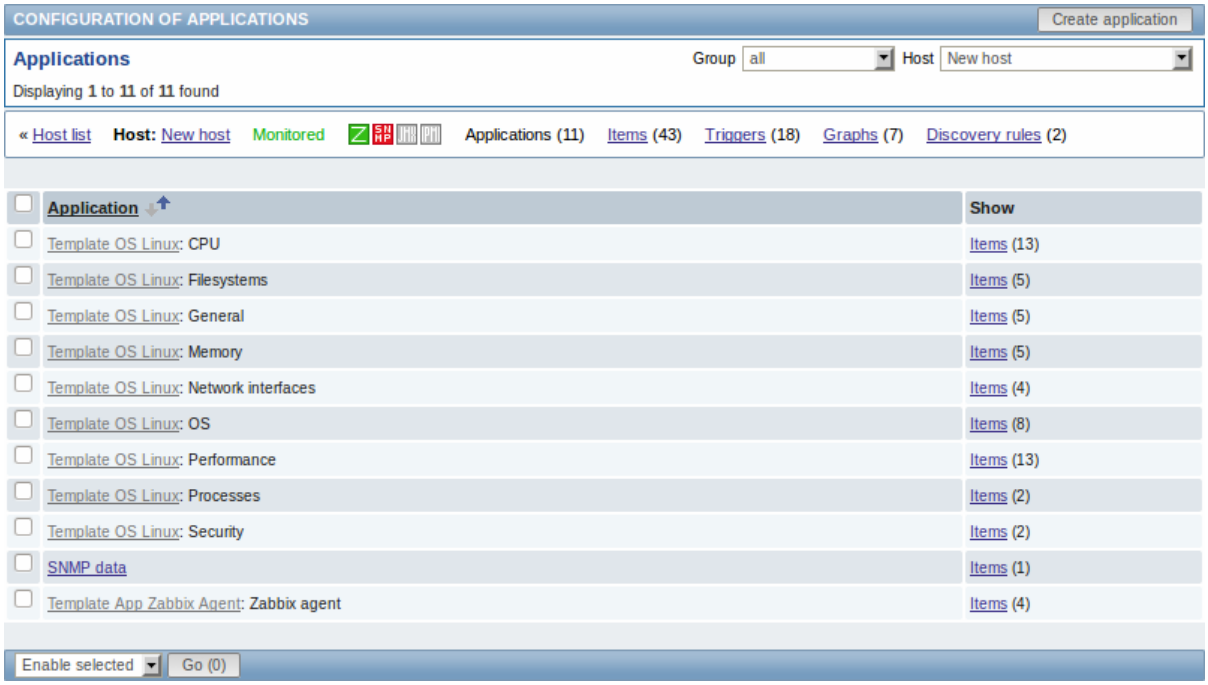
1 Applications

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
Name	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 grey link. Clicking on the template link will open the application list on the template level.
Show	Click on Items to view the items contained in the application. The number of items is displayed in parentheses.

To configure a new application, click on the *Create application* button in the top right-hand corner.

Mass editing options

A dropdown below the list offers some mass-editing options:

- *Enable selected* - change application status to *Enabled*
- *Disable selected* - change application status to *Disabled*
- *Delete selected* - delete the applications

To use these options, mark the check-boxes before the respective applications, then select the required option and click on ”Go”.

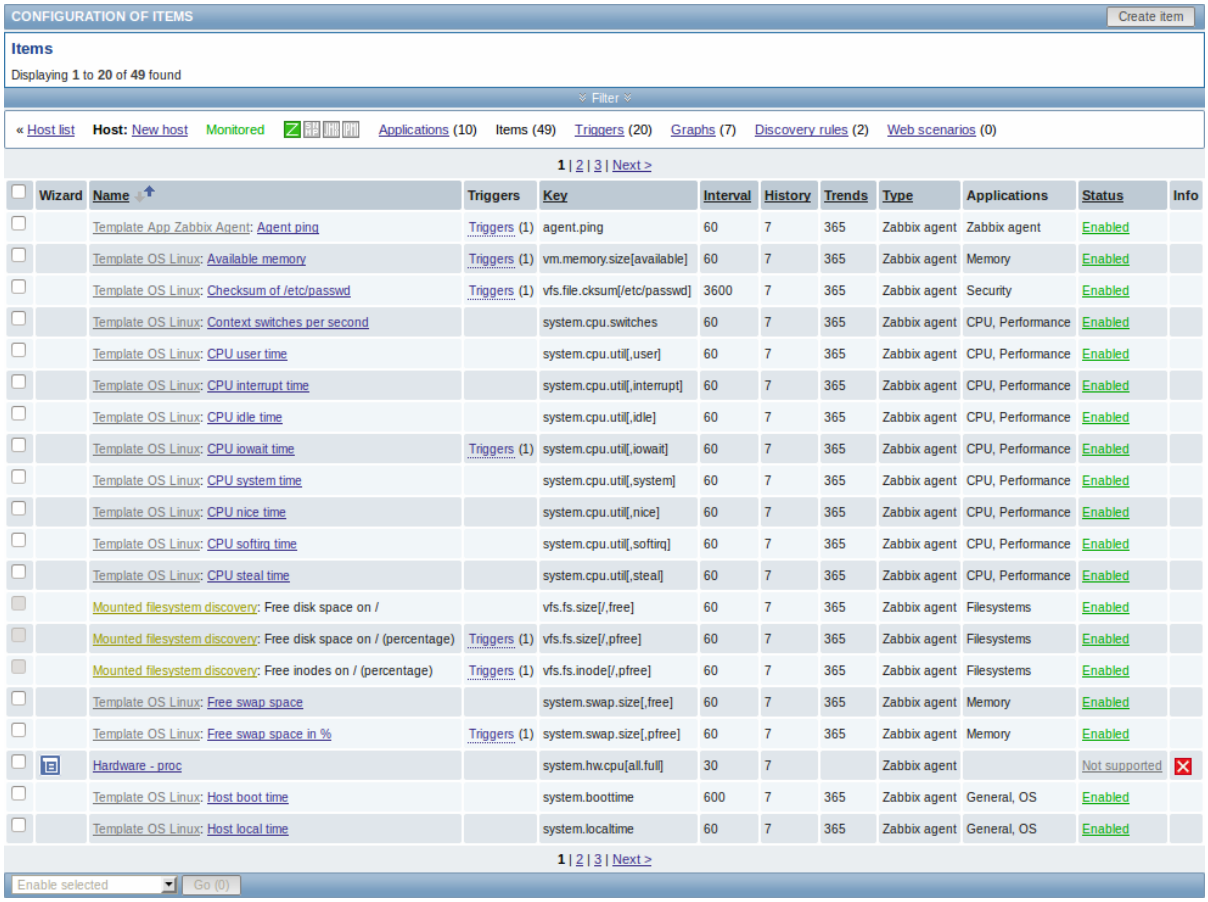
2 Items

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.
Name	Name of the item, mostly displayed as a blue link except for items created from item prototypes. 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 grey 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 khaki. 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 parentheses.
Key	Item key is displayed.
Interval	Frequency of the check is displayed.
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

A dropdown below the list offers some mass-editing options:

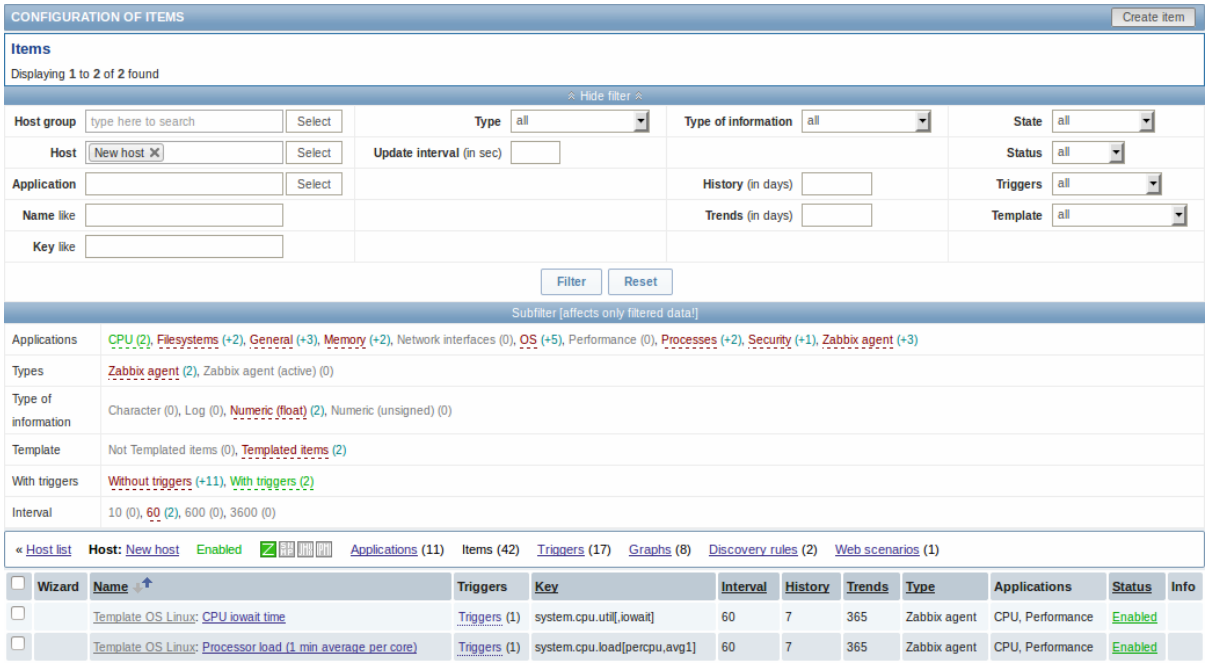
- *Enable selected* - change item status to *Enabled*
- *Disable selected* - change item status to *Disabled*
- *Mass update* - update several properties for a number of items at once
- *Copy selected to...* - copy the items to other hosts or templates
- *Clear history for selected* - delete history and trend data for items
- *Delete selected* - delete the items

To use these options, mark the check-boxes before the respective items, then select the required option and click on ”Go”.

Filter

As the list may contain very many items, it may be needed to filter out the ones you really need.

The narrow blue bar just below the *Items* bar is actually a link to the filter. If you click on it, a filter becomes available where you can filter items by several properties.



The **Subfilter** below the filter offers further filtering options (for the data already filtered). The links in red are groups of items with a common parameter value. If you click on the link it turns green and only the items with this parameter value remain in the list.

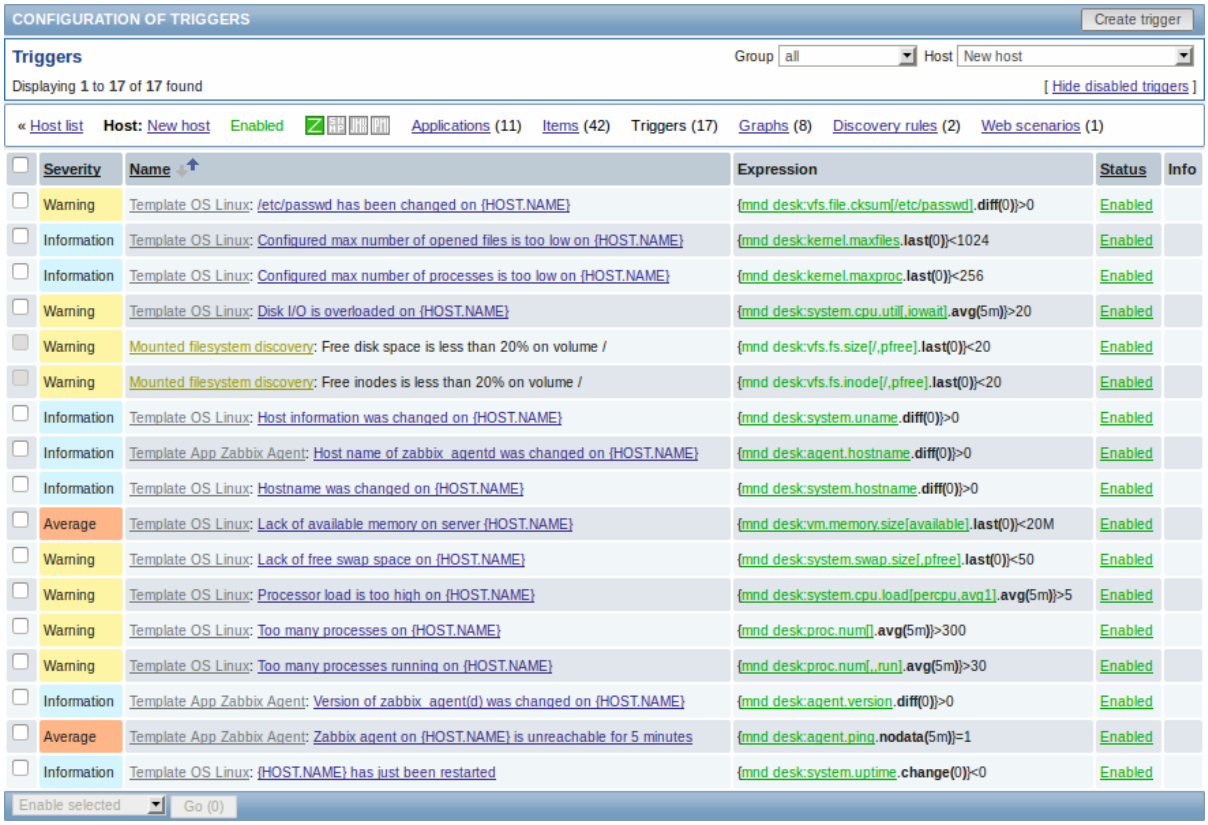
3 Triggers

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.

A list of existing triggers is displayed. By default, only the enabled triggers are displayed. To display disabled triggers as well, use the *Show disabled triggers* link to the right in the Triggers bar.



Displayed data:

Column	Description
Severity	Severity of the trigger is displayed by both name and cell background colour.
Name	Name of the trigger, mostly displayed as a blue link except for triggers created from trigger prototypes. 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 grey 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 khaki. Clicking on the discovery rule name will open the trigger prototype list.
Expression	Trigger expression is displayed. The host-item part of the expression is displayed as a link, leading to the item configuration form.
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).
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 trigger, click on the *Create trigger* button in the top right-hand corner.

Mass editing options

A dropdown below the list offers some mass-editing options:

- *Enable selected* - change trigger status to *Enabled*
- *Disable selected* - change trigger status to *Disabled*
- *Mass update* - update several properties for a number of triggers at once
- *Copy selected to...* - copy the triggers to other hosts or templates
- *Delete selected* - delete the triggers

To use these options, mark the check-boxes before the respective triggers, then select the required option and click on ”Go”.

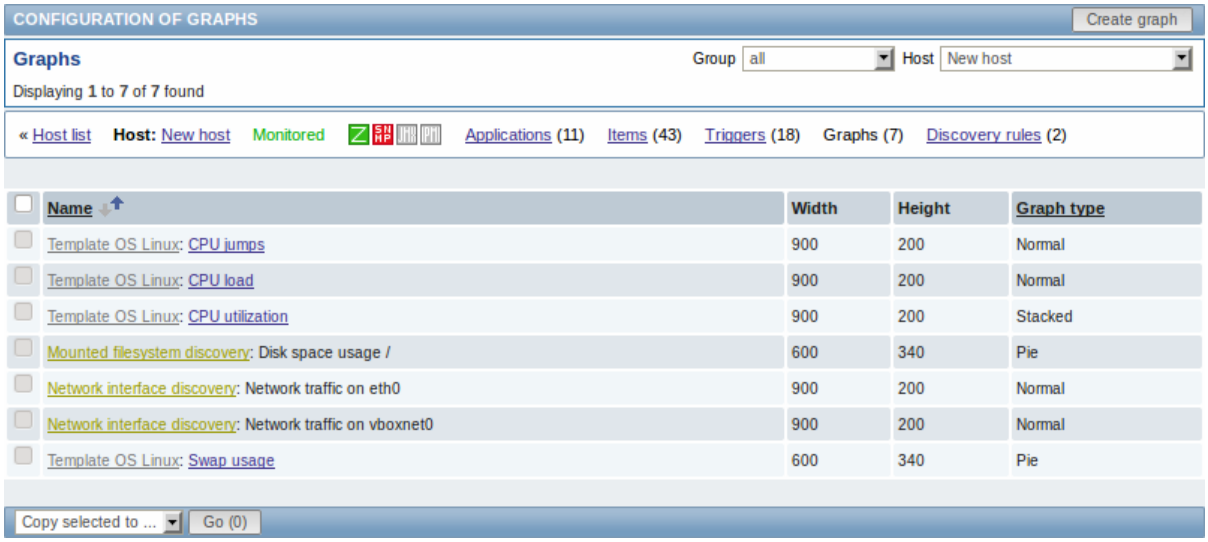
4 Graphs

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, mostly displayed as a blue link except for graphs created from graph prototypes. 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 grey 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 khaki. 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

A dropdown below the list offers some mass-editing options:

- *Copy selected to...* - copy the graphs to other hosts or templates
- *Delete selected* - delete the graphs

To use these options, mark the check-boxes before the respective graphs, then select the required option and click on ”Go”.

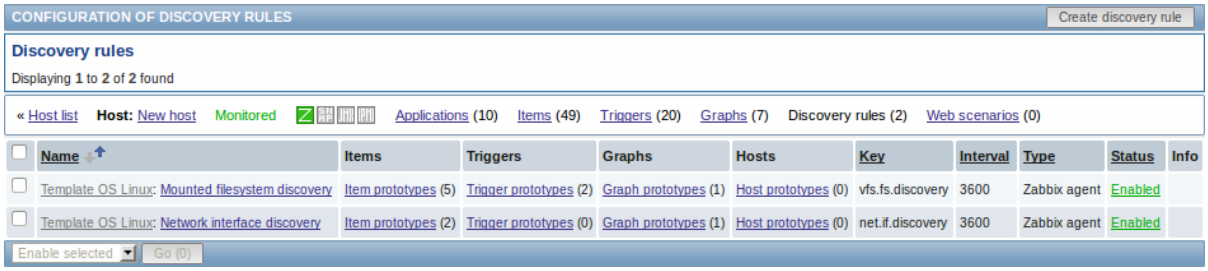
5 Discovery rules

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.



Displayed data:

Column	Description
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 grey 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 parentheses.
Triggers	A link to the list of trigger prototypes is displayed. The number of existing trigger prototypes is displayed in parentheses.
Graphs	A link to the list of graph prototypes displayed. The number of existing graph prototypes is displayed in parentheses.
Hosts	A link to the list of host prototypes displayed. The number of existing host prototypes is displayed in parentheses.
Key	The item key used for discovery is displayed.
Interval	The frequency of performing discovery is displayed.
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).
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

A dropdown below the list offers some mass-editing options:

- *Enable selected* - change the low-level discovery rule status to *Enabled*
- *Disable selected* - change the low-level discovery rule status to *Disabled*
- *Delete selected* - delete the low-level discovery rules

To use these options, mark the check-boxes before the respective discovery rules, then select the required option and click on ”Go”.

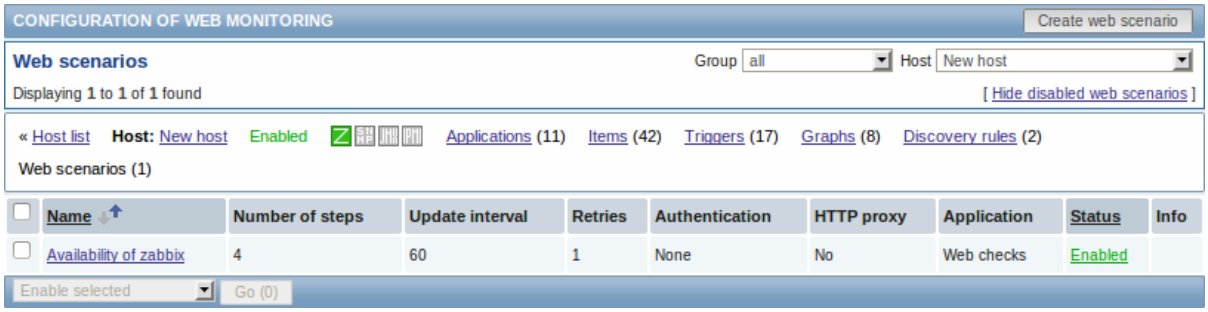
6 Web scenarios

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. From the dropdown to the right in the *Scenarios* bar you can choose whether to display all web scenarios or only those belonging to one particular group and host. Additionally you can choose to hide disabled scenarios (or show them again) by clicking on the respective link.



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.
Retries	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 scenario* button in the top right-hand corner.

Mass editing options

A dropdown below the list offers some mass-editing options:

- *Enable selected* - change the scenario status to *Enabled*
- *Disable selected* - change the scenario status to *Disabled*
- *Clear history for selected* - clear history and trend data for the scenarios
- *Delete selected* - delete the web scenarios

To use these options, mark the check-boxes before the respective web scenarios, then select the required option and click on "Go".

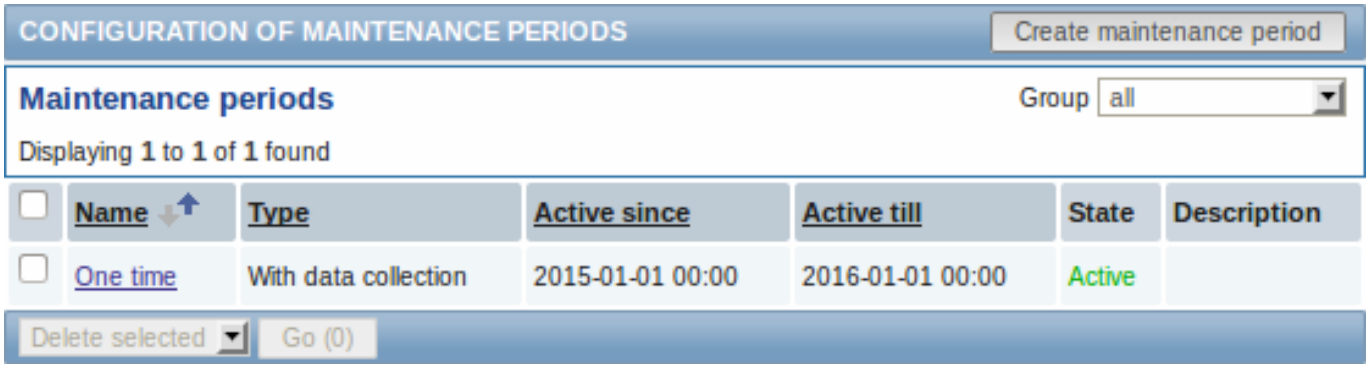
4 Maintenance

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.

From the dropdown to the right in the *Maintenance periods* bar you can choose whether to display all maintenance periods or only those belonging to one particular group.



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.
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.

Name, *Type*, *Active since* and *Active till* are sortable columns that can be sorted in ascending/descending order. To sort, click on the column name.

To configure a new maintenance period, click on the *Create maintenance period* button in the top right-hand corner.

Mass editing options

A dropdown below the list offers one mass-editing option:

- *Delete selected* - delete the maintenance periods

To use this option, mark the check-boxes before the respective maintenance periods and click on ”Go”.

5 Actions

Overview

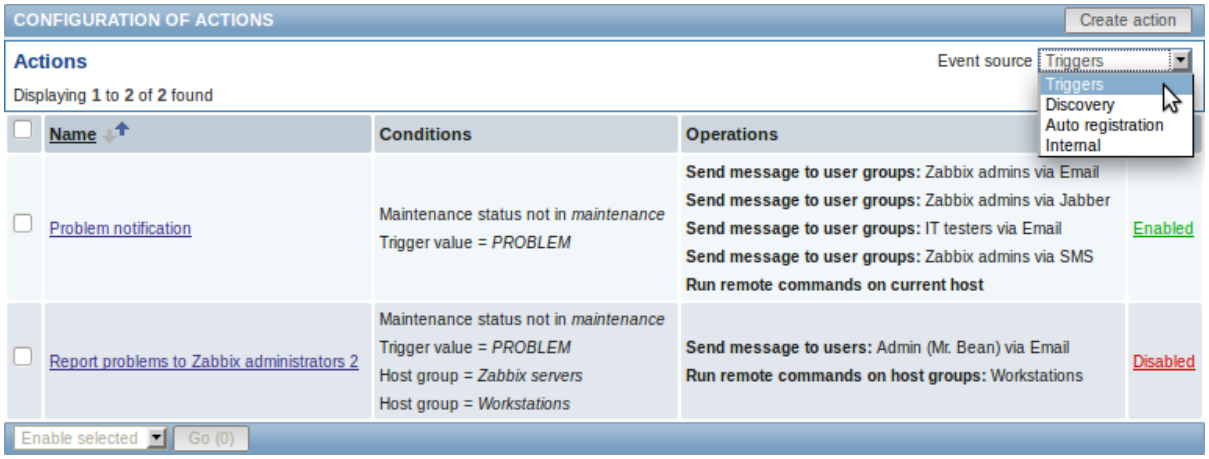
In the *Configuration* → *Actions* section users can configure and maintain actions.

A listing of existing actions with their details is displayed. The actions displayed are actions assigned to the selected event source (triggers, discovery, auto-registration).

To view actions assigned to a different event source, change the source from the dropdown to the right in the *Actions* bar.

For users without Super-admin rights actions are displayed according to permission settings. That means in some cases a user without Super-admin rights isn’t able to view the complete action list because of certain permission restrictions. An action is displayed to the user without Super-admin rights if the following conditions are fulfilled:

- The user has read-write access to all selected elements in the new condition (host, host group, trigger, etc.);
- The user is a member of all the user groups notifications are sent to. Otherwise this user will not be able to view this action.



Displayed data:

Column	Description
Name	Name of the action. Clicking on the action name opens the action configuration form.
Conditions	Action conditions are displayed.

Column	Description
Operations	Action operations are displayed. Since Zabbix 2.2, the operation list also displays the media type (e-mail, SMS, Jabber, etc) used for notification as well as the name and surname (in parentheses after the alias) of a notification recipient.
Status	Action status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it. If an action is disabled during an escalation in progress (like a message being sent), the message in progress will be sent and then one more message on the escalation will be sent. The follow-up message will have the following text at the beginning of the message body: *NOTE: Escalation cancelled: action '<Action name>' disabled*. This way the recipient is informed that the escalation is cancelled and no more steps will be executed.

To configure a new action, click on the *Create action* button in the top right-hand corner.

Mass editing options

A dropdown below the list offers some mass-editing options:

- *Enable selected* - change the action status to *Enabled*
- *Disable selected* - change the action status to *Disabled*
- *Delete selected* - delete the actions

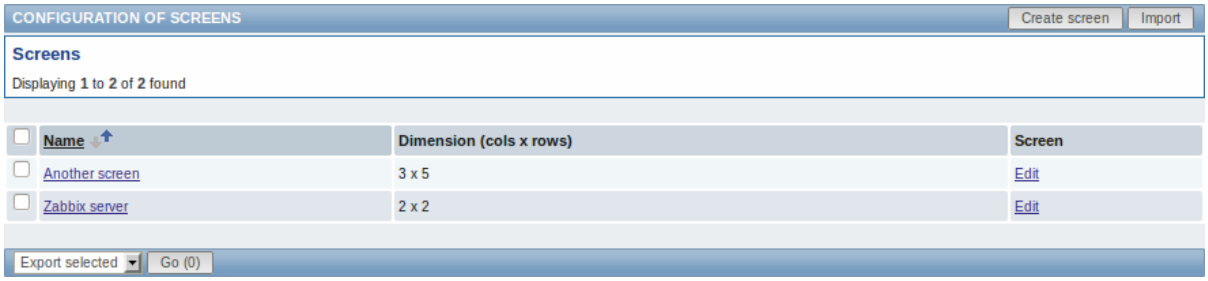
To use these options, mark the check-boxes before the respective actions, then select the required option and click on "Go".

6 Screens

Overview

In the *Configuration* → *Screens* section users can configure and maintain screens.

A listing of existing screens with their details is displayed.



Displayed data:

Column	Description
Name	Name of the screen. By clicking on the screen name you can open the grid of screen elements for editing.
Dimensions	The number of columns and rows of the screen.
Screen	Click on the *Edit* link to edit general screen properties (name and dimensions).

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.

Mass editing options

A dropdown below the list offers some mass-editing options:

- *Export selected* - export the screens to an XML file
- *Delete selected* - delete the screens

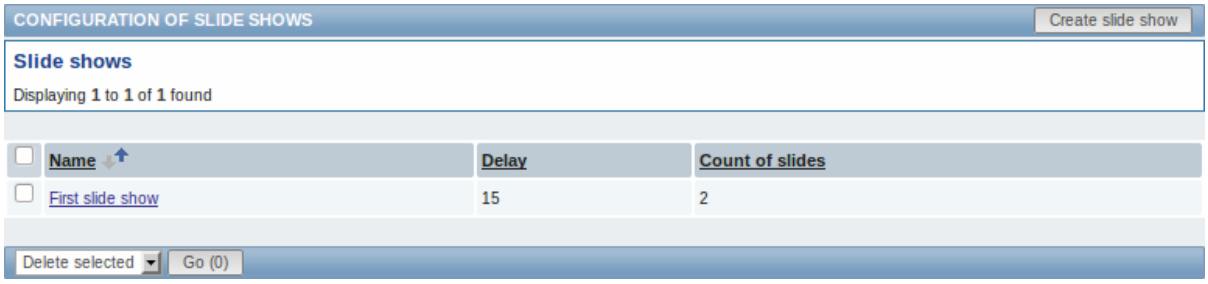
To use these options, mark the check-boxes before the respective screens, then select the required option and click on "Go".

7 Slide shows

Overview

In the *Configuration* → *Slide shows* section users can configure and maintain slide shows.

A listing of existing slide shows with their details is displayed.



Displayed data:

Column	Description
Name	Name of the slide show. Clicking on the slide show name opens the slide show configuration form.
Delay	The default duration of showing one slide is displayed.
Count of slides	The number of slides in the slide show is displayed.

To configure a new slide show, click on the *Create slide show* button in the top right-hand corner.

Mass editing options

A dropdown below the list offers one mass-editing option:

- *Delete selected* - delete the slide shows

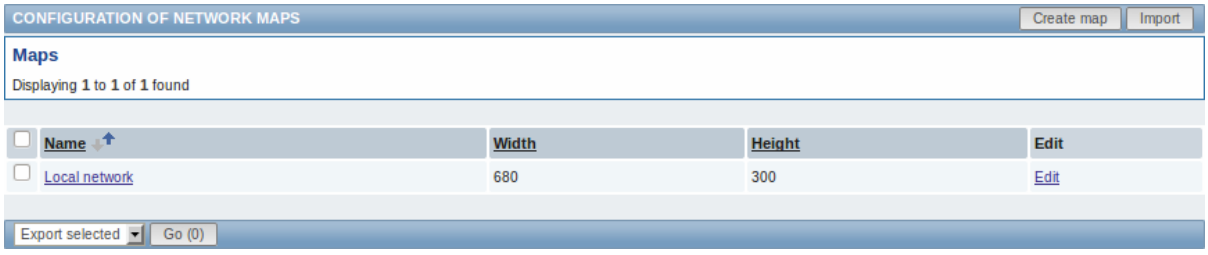
To use this option, mark the check-boxes before the respective slide shows and click on ”Go”.

8 Maps

Overview

In the *Configuration* → *Maps* section users can configure and maintain network maps.

A listing of existing maps with their details is displayed.



Displayed data:

Column	Description
Name	Name of the map. By clicking on the map name you can access the grid for adding map elements.
Width	Map width is displayed.
Height	Map height is displayed.
Edit	Click on the *Edit* link to edit general map properties.

To create 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.

Mass editing options

A dropdown below the list offers some mass-editing options:

- *Export selected* - export the maps to an XML file
- *Delete selected* - delete the maps

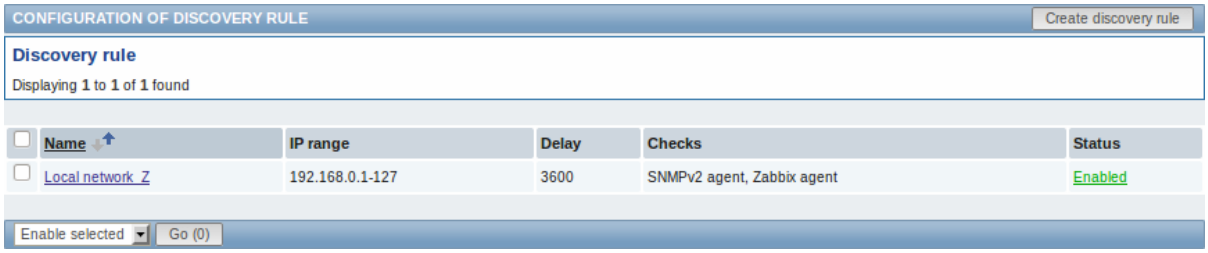
To use these options, mark the check-boxes before the respective maps, then select the required option and click on ”Go”.

9 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.
Delay	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

A dropdown below the list offers some mass-editing options:

- *Enable selected* - change the discovery rule status to *Enabled*
- *Disable selected* - change the discovery rule status to *Disabled*
- *Delete selected* - delete the discovery rules

To use these options, mark the check-boxes before the respective discovery rules, then select the required option and click on ”Go”.

10 IT services

Overview

In the *Configuration* → *IT 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 it and add services and then other services below the ones you have added.

For details on adding services, see the IT services section.

5 Administration

Overview

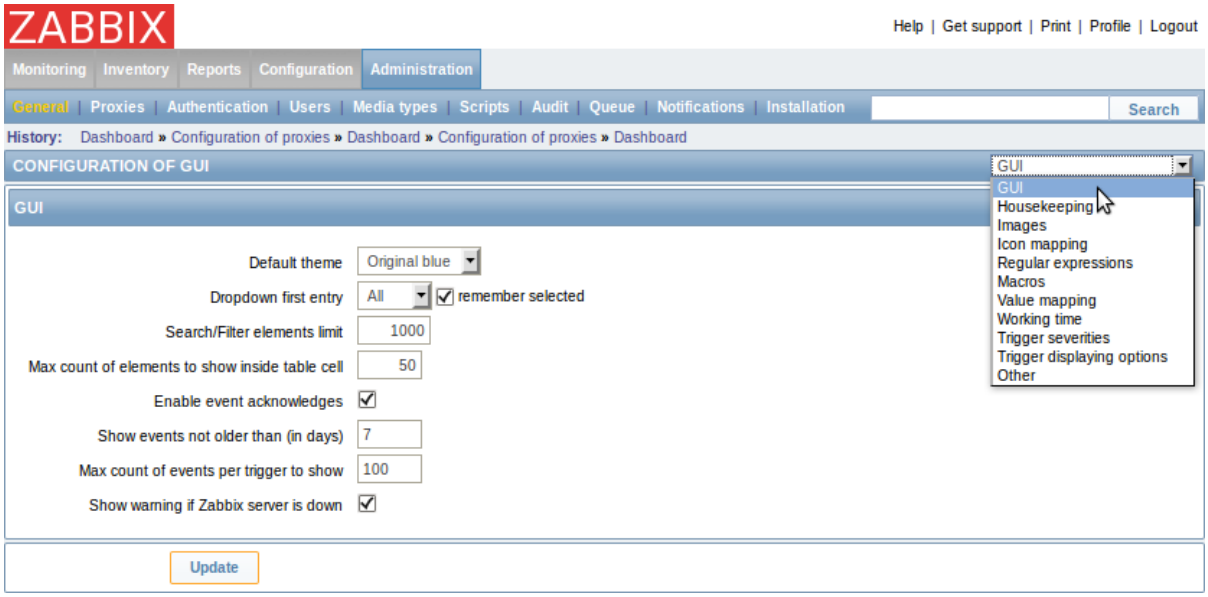
The Administration menu is for administrative functions of Zabbix. This menu is available to users of Super Administrators type only.

1 General

Overview

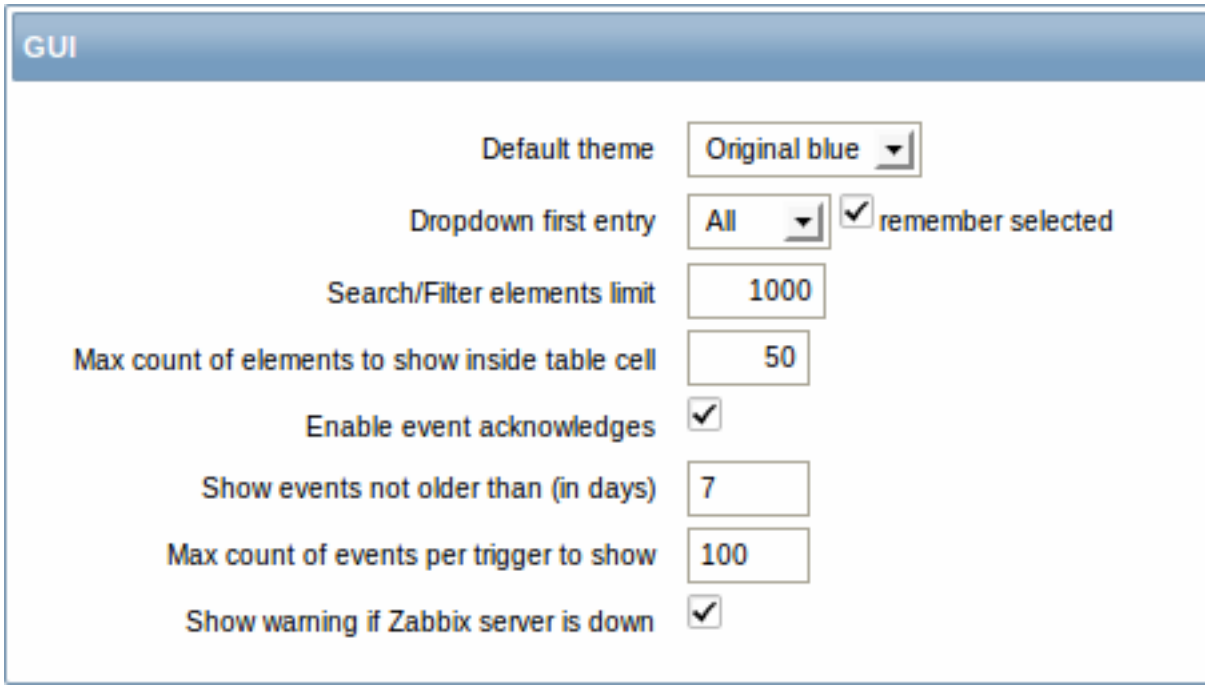
The *Administration* → *General* section contains a number of screens for setting frontend-related defaults and customizing Zabbix.

The dropdown to the right allows you to switch between different configuration screens.



1 GUI

This screen provides customization of several frontend-related defaults.

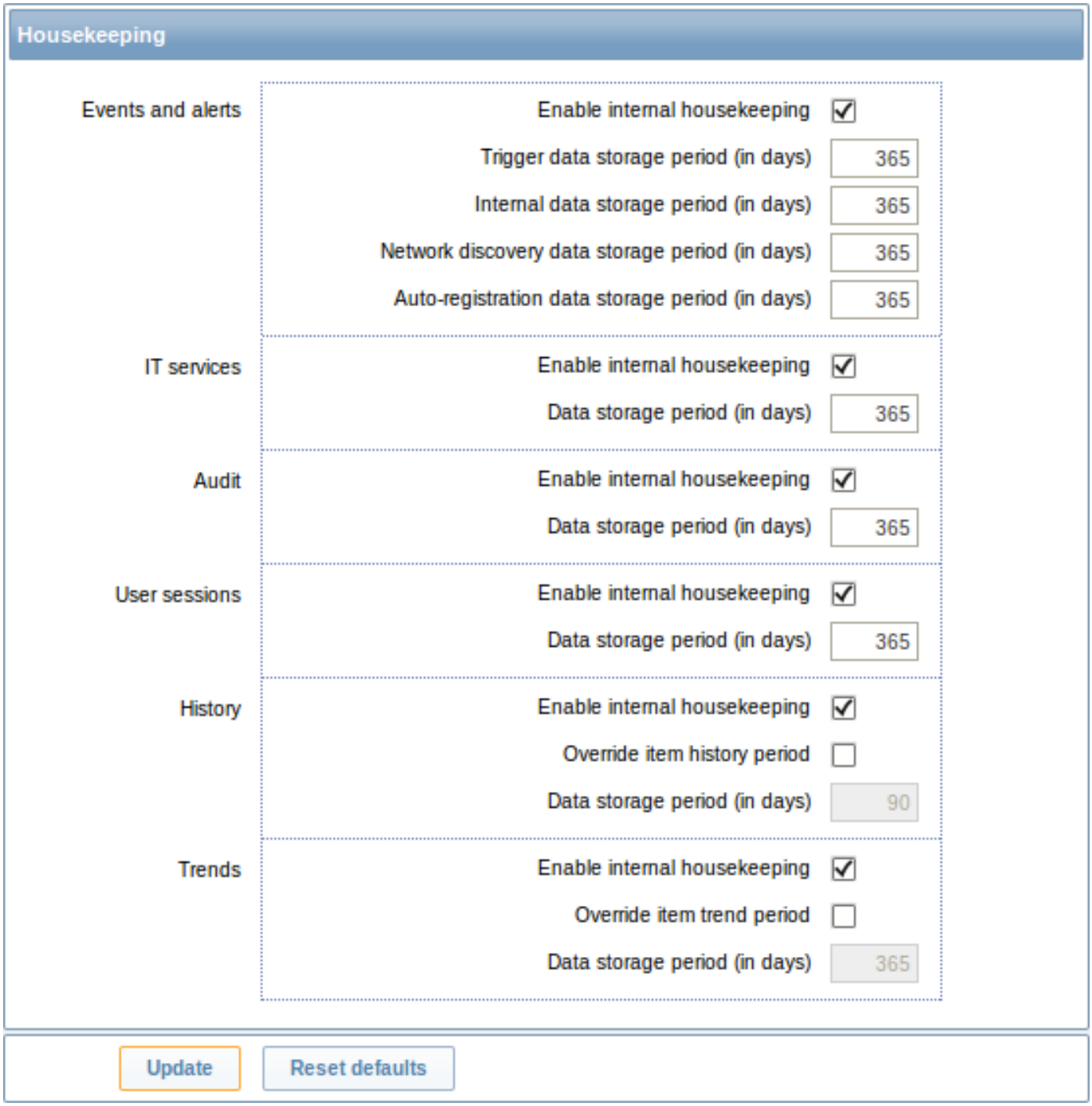


Configuration parameters:

Parameter	Description
Default theme	Default theme for users who have not set a specific one in their profiles.
Dropdown first entry	Whether first entry in element selection dropdowns should be *All* or *None*. With *remember selected* checked, the last selected element in the dropdown will be remembered (instead of the default) when navigating to another page.
Search/Filter elements limit	Maximum amount of elements (rows) that will be displayed in a web-interface list, like, for example, in *Monitoring* → *Events* or *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.
Enable event acknowledges	This parameter defines if event acknowledgments are activated in Zabbix interface.
Show events not older than (in days)	This parameter defines for how many days events are displayed in Status of Triggers screen. Default is 7 days.
Max count of events per trigger to show	Maximum number of event to show for each trigger in Status of Triggers screen. Default is 100.
Show warning if Zabbix server is down	This parameter enables a warning message to be displayed atop 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 Housekeeper

The housekeeper is a periodical process, executed by Zabbix server. The process removes outdated information and information deleted by user.



In this section housekeeping tasks can be enabled or disabled on a per-task basis separately for: events and alerts/IT services/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.

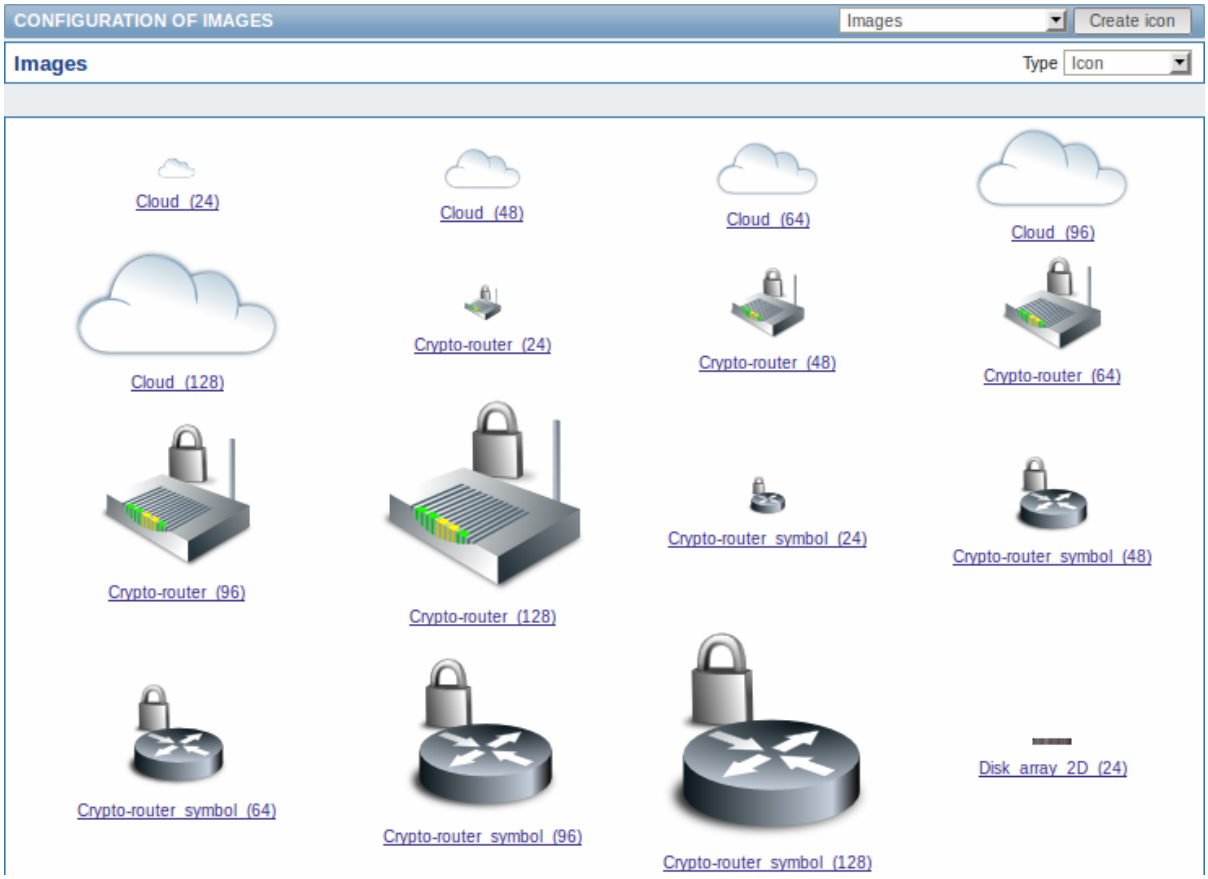
For history and trends an additional option is available: *Override item history period* and *Override item trends period*. This option allows to globally set for how many days item history/trends will be kept, in this case overriding the values set for individual items in *Keep history/Keep trends* fields in item configuration.

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.

Reset defaults button allows to revert any changes made.

3 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.

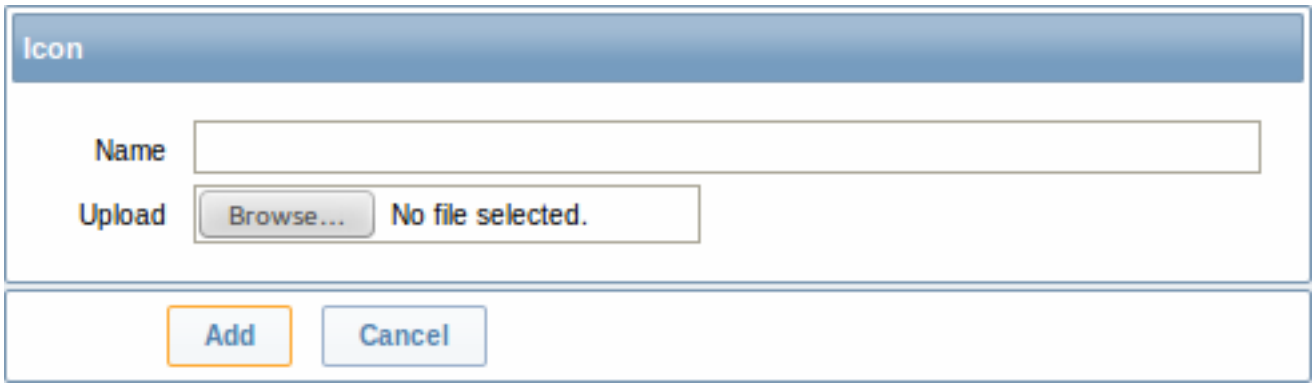


Image attributes:

Parameter	Description
Name	Unique name of an image.
Upload	Select the file (PNG, JPEG) from a local system to be uploaded to Zabbix.

Note:
Maximum size of the upload file is limited by value of ZBX_MAX_IMAGE_SIZE that is 1024x1024 bytes or 1 MB.

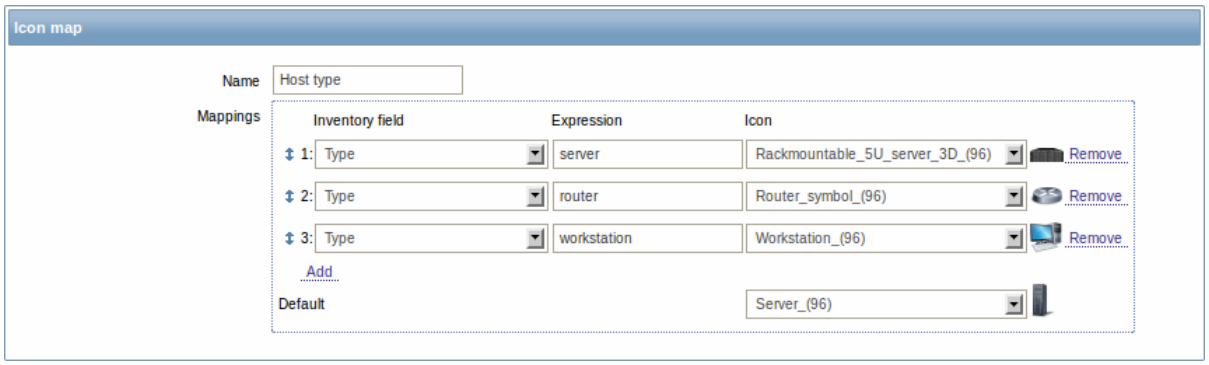
The upload of an image may fail if the image size is close to 1 MB and the `max_allowed_packet` MySQL configuration parameter is at a default of 1MB. In this case, increase the max_allowed_packet parameter.

4 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:

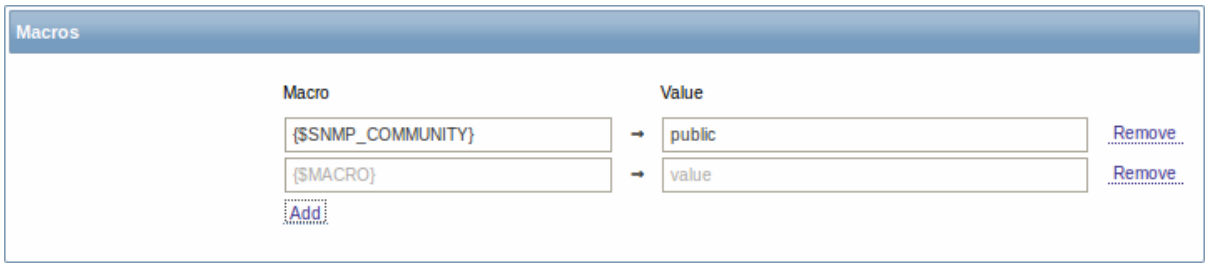
Parameter	Description
Name	Unique name of icon map.
Mappings	A list of mappings. The order of mappings determines which one will have priority. You can move mappings up and down the list with drag-and-drop.
Inventory field	Host inventory field that will be looked into to seek a match.
Expression	Regular expression describing the match.
Icon	Icon to use if a match for the expression is found.
Default	Default icon to use.

5 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.

6 Macros

This section allows to define system-wide macros.



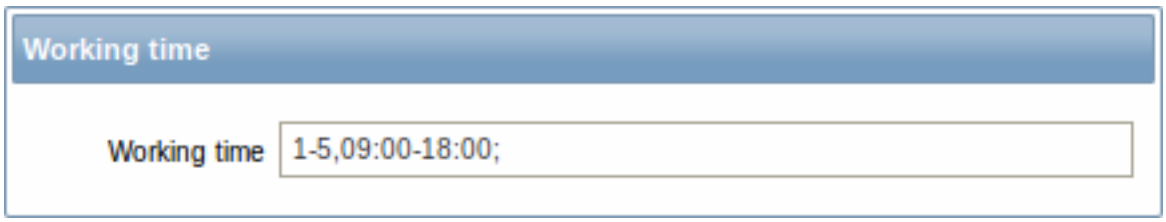
See User macros section for more details.

7 Value mapping

This section allows to create value maps that allow for human-readable representation of incoming data in Zabbix frontend. See Value mapping section for more details.

8 Working time

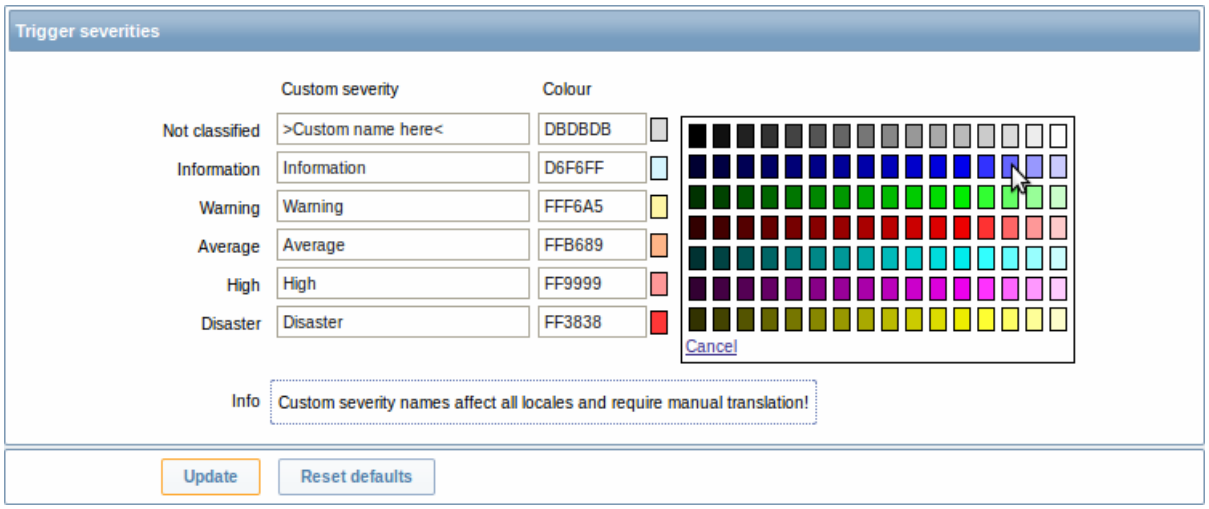
Working time is system-wide parameter, which defines working time. Working time is displayed as a white background in graphs, while non-working time is displayed in grey.



See Time period specification page for description of the time format.

9 Trigger severities

This section allows to customize trigger severity names and colors.

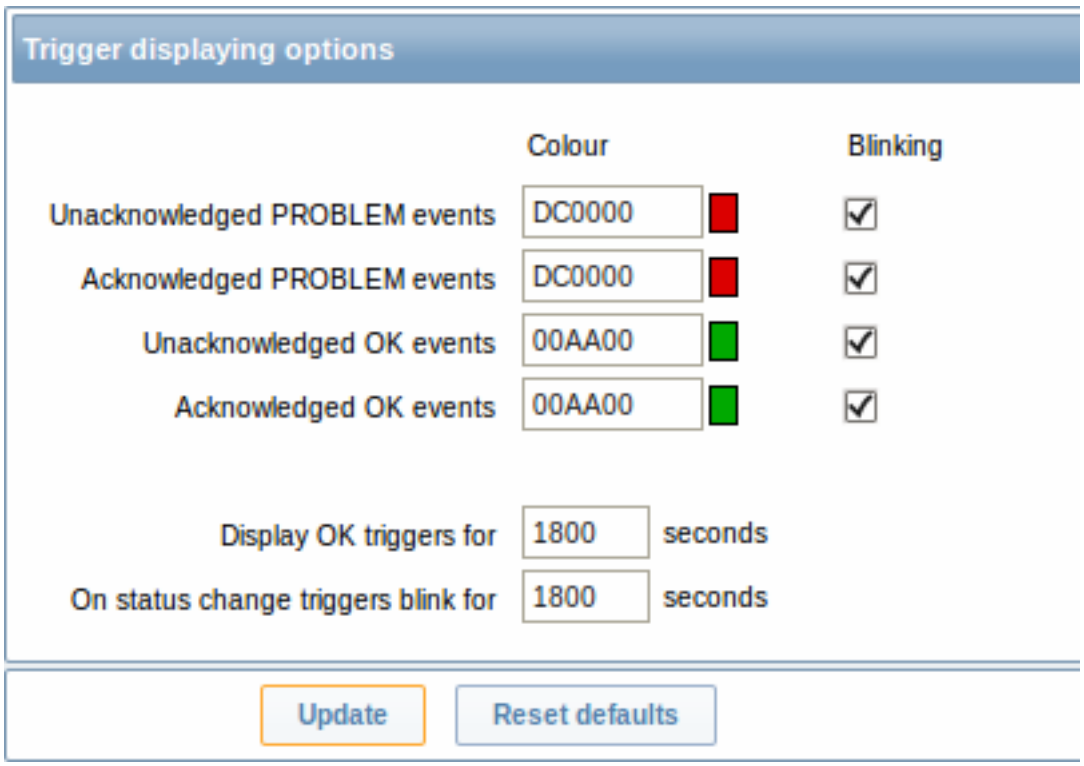


You can enter new names and color codes or click on the color to select another from the provided palette.

See Customising trigger severities page for more information.

10 Trigger displaying options

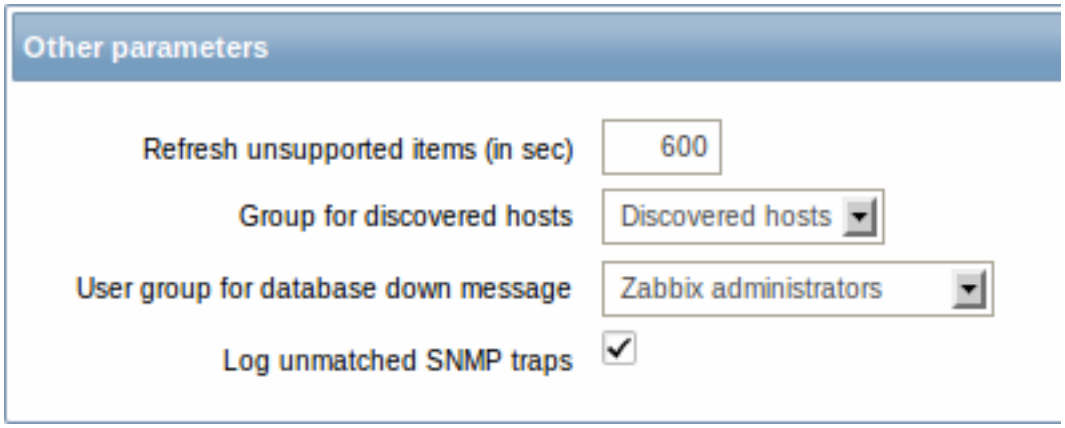
This section allows to customize how trigger status is displayed in the frontend.



The colors for acknowledged/unacknowledged events can be customized and blinking enabled or disabled. Also the time period for displaying OK triggers and for blinking upon trigger status change can be customized.

11 Other parameters

This section allows to configure several other frontend parameters.



Parameter	Description
Refresh unsupported items (in sec)	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 will activate unsupported item every N seconds set here. If set to 0, the automatic activation will be disabled. The configured value also applies to how often Zabbix proxies reactivate unsupported items.
Group for discovered hosts	Hosts discovered by network discovery and agent auto-registration will be automatically placed in the host group, selected here.
User group for database down message	User group for sending alarm message or 'None'. Availability of Zabbix server depends on availability of backend database. It cannot work without a database. **Database watchdog**, a special Zabbix server process, will alarm selected users in case of disaster. If the database is down, the watchdog will send notifications 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. *Note*: Until Zabbix version 1.8.2 database watchdog was supported for MySQL only. Since 1.8.2, it is supported for all databases.
Log unmatched SNMP traps	Log SNMP trap if no corresponding SNMP interfaces have been found.

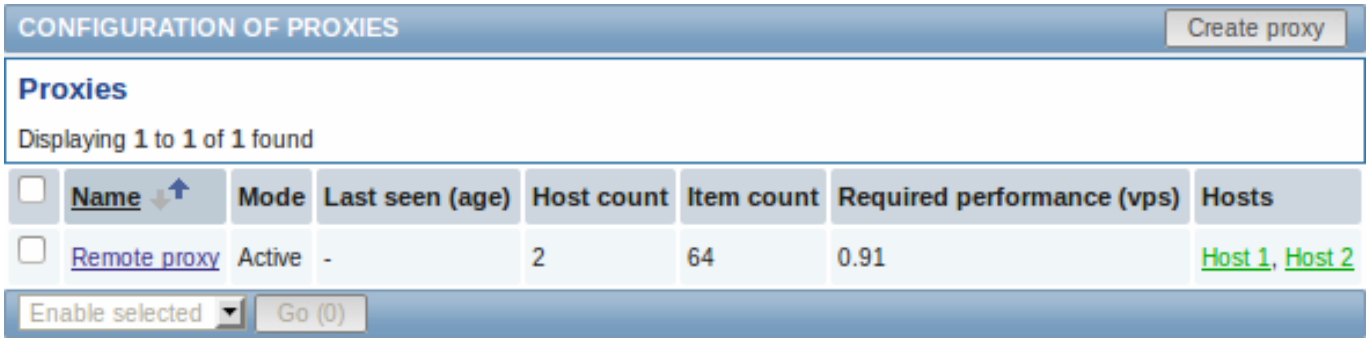
2 Proxies

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*.
Last seen (age)	The time when the proxy was last seen by the server is displayed.
Host count	The number of hosts monitored by the proxy is displayed.
Item count	The number of items monitored by 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

A dropdown below the list offers some mass-editing options:

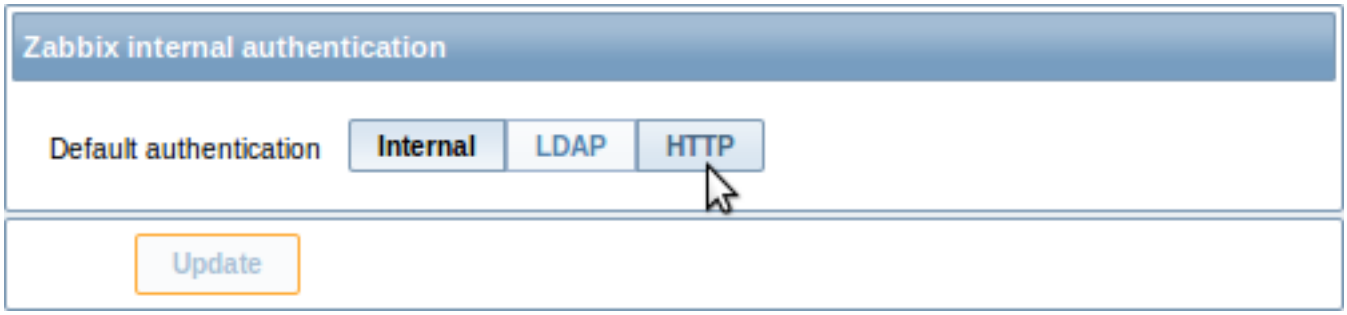
- *Enable selected* - change the status of hosts monitored by the proxy to *Monitored*
- *Disable selected* - change the status of hosts monitored by the proxy to *Not monitored*
- *Delete selected* - delete the proxies

To use these options, mark the check-boxes before the respective proxies, then select the required option and click on ”Go”.

3 Authentication

Overview

In *Administration* → *Authentication* the user authentication method to Zabbix can be changed. The available methods are internal, LDAP and HTTP authentication.



By default, internal Zabbix authentication is used. To change, click on the button with the method name and press *Update*.

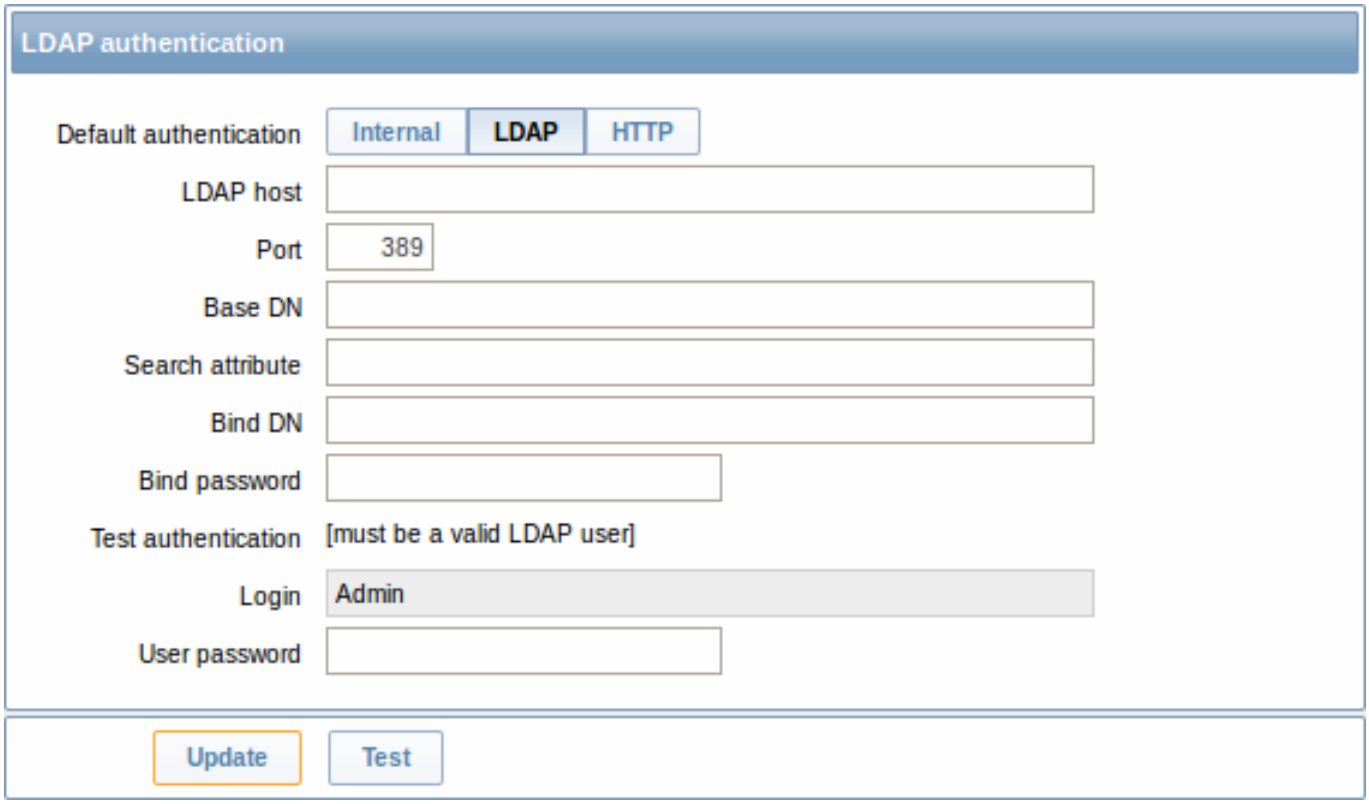
Internal

Internal Zabbix authentication is used.

LDAP

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.

Zabbix LDAP authentication works at least with Microsoft Active Directory and OpenLDAP.



Configuration parameters:

Parameter	Description
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) Required, anonymous binding is not supported.
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.

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.

Note:
Some user groups can still be authenticated by Zabbix. These groups must have frontend access set to Internal.

HTTP

Apache-based (HTTP) 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.

Attention:
Be careful! Make sure that Apache authentication is configured and works properly before switching it on.

Note:
In case of Apache authentication all users (even with frontend access set to Internal) will be authenticated by Apache, not by Zabbix!

4 Users

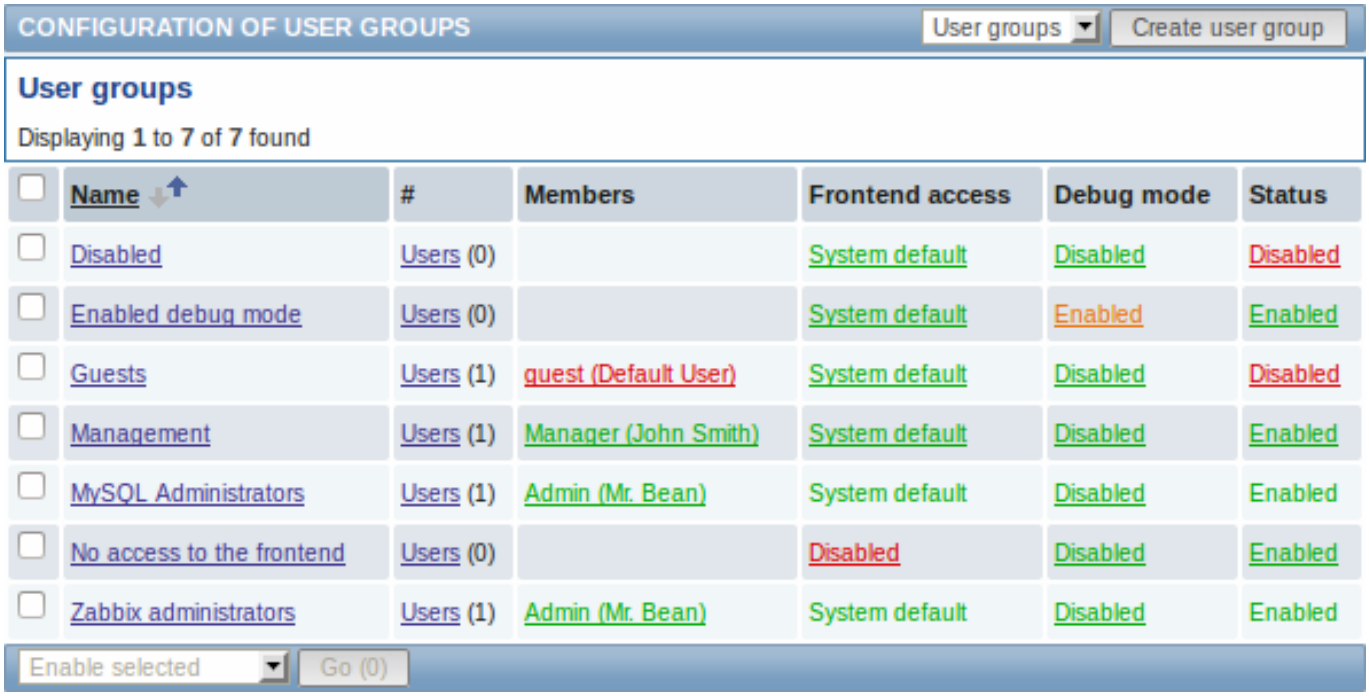
Overview

In the *Administration* → *Users* section both user groups and users of the system are maintained.

By default the user group screen is displayed. To switch to the user screen and back, use the dropdown in the top right-hand corner.

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.

Column	Description
#	The number of users in the group (displayed in parentheses). 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.
Frontend access	Frontend access level is displayed: **System default** - Zabbix, LDAP or HTTP authentication; depending on the chosen authentication method **Internal** - the user is authenticated by Zabbix regardless of system settings **Disabled** - frontend access for this user is disabled. By clicking on the current level you can change it.
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

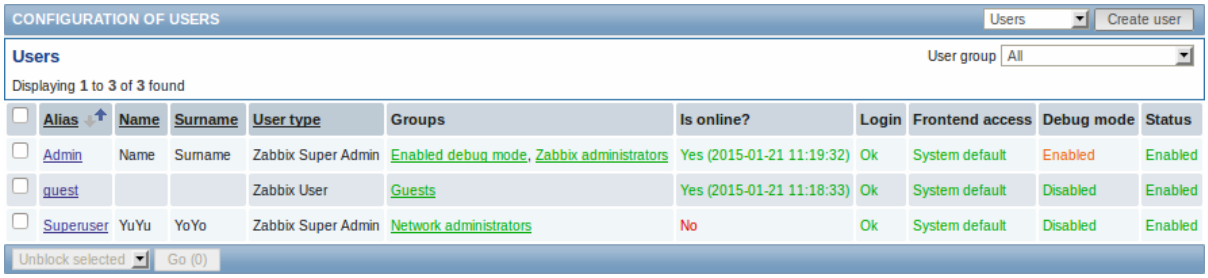
A dropdown below the list offers some mass-editing options:

- *Enable selected* - change the user group status to *Enabled*
- *Disable selected* - change the user group status to *Disabled*
- *Enable DEBUG* - enable debug mode for the user groups
- *Disable DEBUG* - disable debug mode for the user groups
- *Delete selected* - delete the user groups

To use these options, mark the check-boxes before the respective user groups, then select the required option and click on "Go".

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.
Surname	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* or *Disabled*, depending on the one set for the whole user group.

Column	Description
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

A dropdown below the list offers some mass-editing options:

- *Unblock selected* - re-enable system access to blocked users
- *Delete selected* - delete the users

To use these options, mark the check-boxes before the respective users, then select the required option and click on ”Go”.

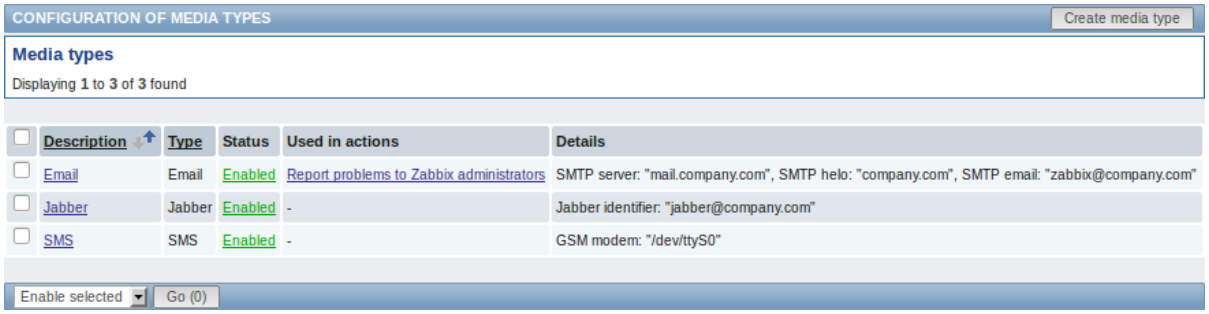
5 Media types

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
Description	Description of the media type. Clicking on the description 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.

To configure a new media type, click on the *Create media type* button in the top right-hand corner.

Mass editing options

A dropdown below the list offers some mass-editing options:

- *Enable selected* - change the media type status to *Enabled*
- *Disable selected* - change the media type status to *Disabled*
- *Delete selected* - delete the media types

To use these options, mark the check-boxes before the respective media types, then select the required option and click on ”Go”.

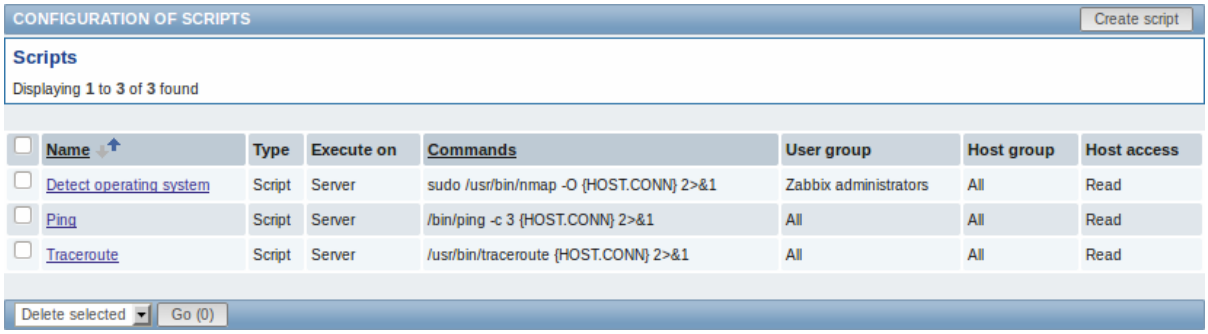
6 Scripts

Overview

In the *Administration* → *Scripts* section user-defined global scripts can be configured and maintained.

These scripts, depending on the set user permissions, then become available for execution by clicking on the host in various frontend locations (*Dashboard*, *Latest data*, *Status of triggers*, *Events*, *Maps*) and can also be run as an action operation. The scripts are executed on the Zabbix server or agent.

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.

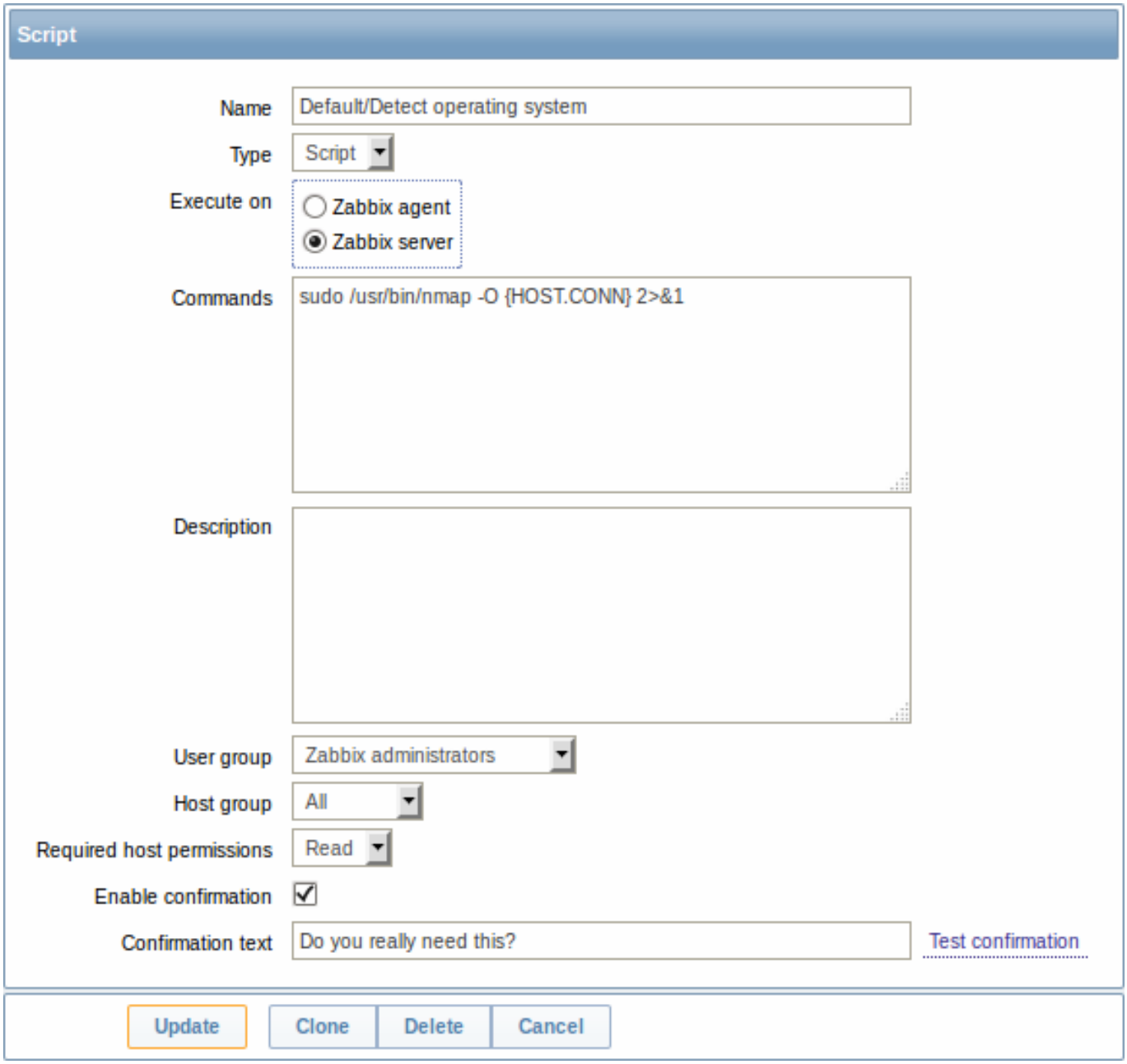
Mass editing options

A dropdown below the list offers one mass-editing option:

- *Delete selected* - delete the scripts

To use this option, mark the check-boxes before the respective scripts and click on ”Go”.

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	Select script type - Script or IPMI command. A special dropdown selection for scripts containing IPMI commands is available since Zabbix 2.0 version (previously a special syntax of *IPMI <command>* had to be used in the command field).

Parameter	Description
Execute on	Select the radio button whether to execute the script on Zabbix server or agent. The option to execute scripts on Zabbix agent is available since Zabbix 2.0 version (providing remote commands are enabled in the EnableRemoteCommands parameter in Zabbix agent configuration file).
Commands	Enter full path to the commands to be executed within the script. The following macros are supported in the commands: {HOST.CONN}, {HOST.IP}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}. If a macro may resolve to a value with spaces (for example, host name), don't forget to quote as needed. Since Zabbix 2.2, user macros are supported in script commands.
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. Since Zabbix 2.2, the confirmation text will expand host name macros - {HOST.HOST}, {HOST.NAME}, host connection macros - {HOST.IP}, {HOST.DNS}, {HOST.CONN} and user macros. *Note:* The macros will not be expanded when testing the confirmation message.

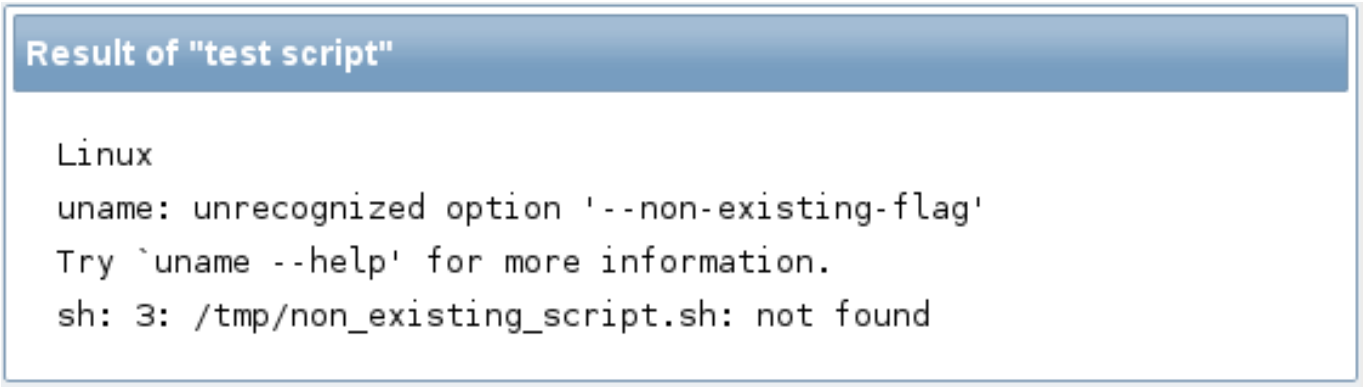
Script result

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 example of a script and the result window below:

```
uname
uname --non-existing-flag
/tmp/non_existing_script.sh
```



7 Audit

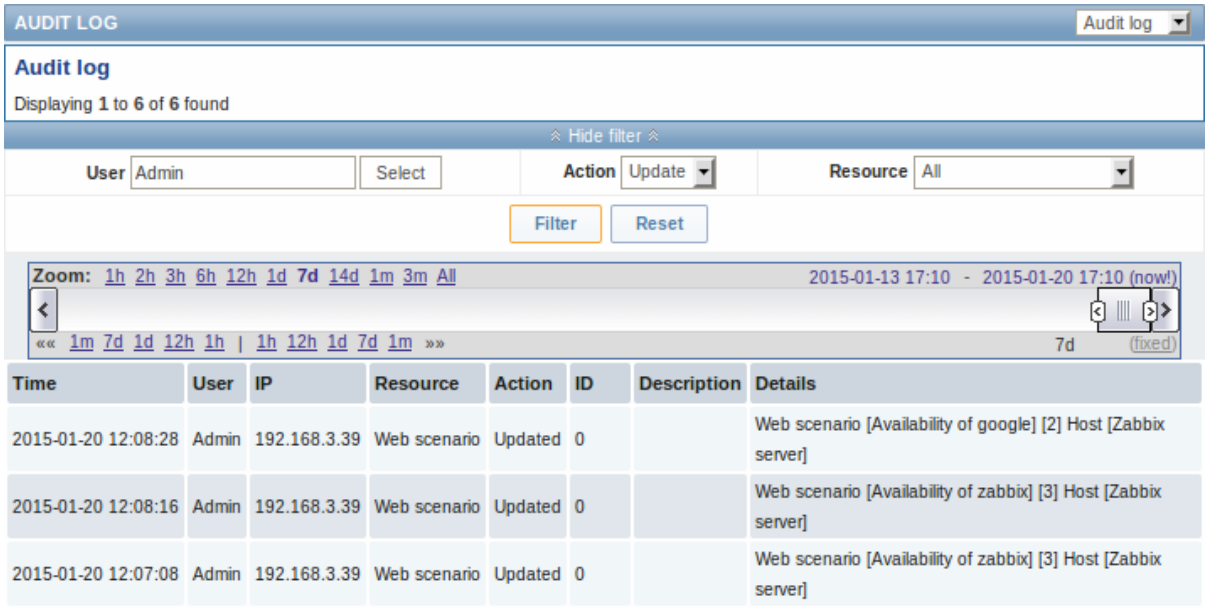
Overview

In the *Administration* → *Audit* section users can view records of changes made in the frontend and details of executed actions.

By default frontend audit records are displayed. To switch to action details and back, use the dropdown in the top right-hand corner.

Audit log

In this screen the audit log of various changes made in the frontend can be seen. You can use the filter, located below the *Audit log* bar, to narrow down the records by user, activity type, affected resource and the time period.



Time	User	IP	Resource	Action	ID	Description	Details
2015-01-20 12:08:28	Admin	192.168.3.39	Web scenario	Updated	0		Web scenario [Availability of google] [2] Host [Zabbix server]
2015-01-20 12:08:16	Admin	192.168.3.39	Web scenario	Updated	0		Web scenario [Availability of zabbix] [3] Host [Zabbix server]
2015-01-20 12:07:08	Admin	192.168.3.39	Web scenario	Updated	0		Web scenario [Availability of zabbix] [3] Host [Zabbix server]

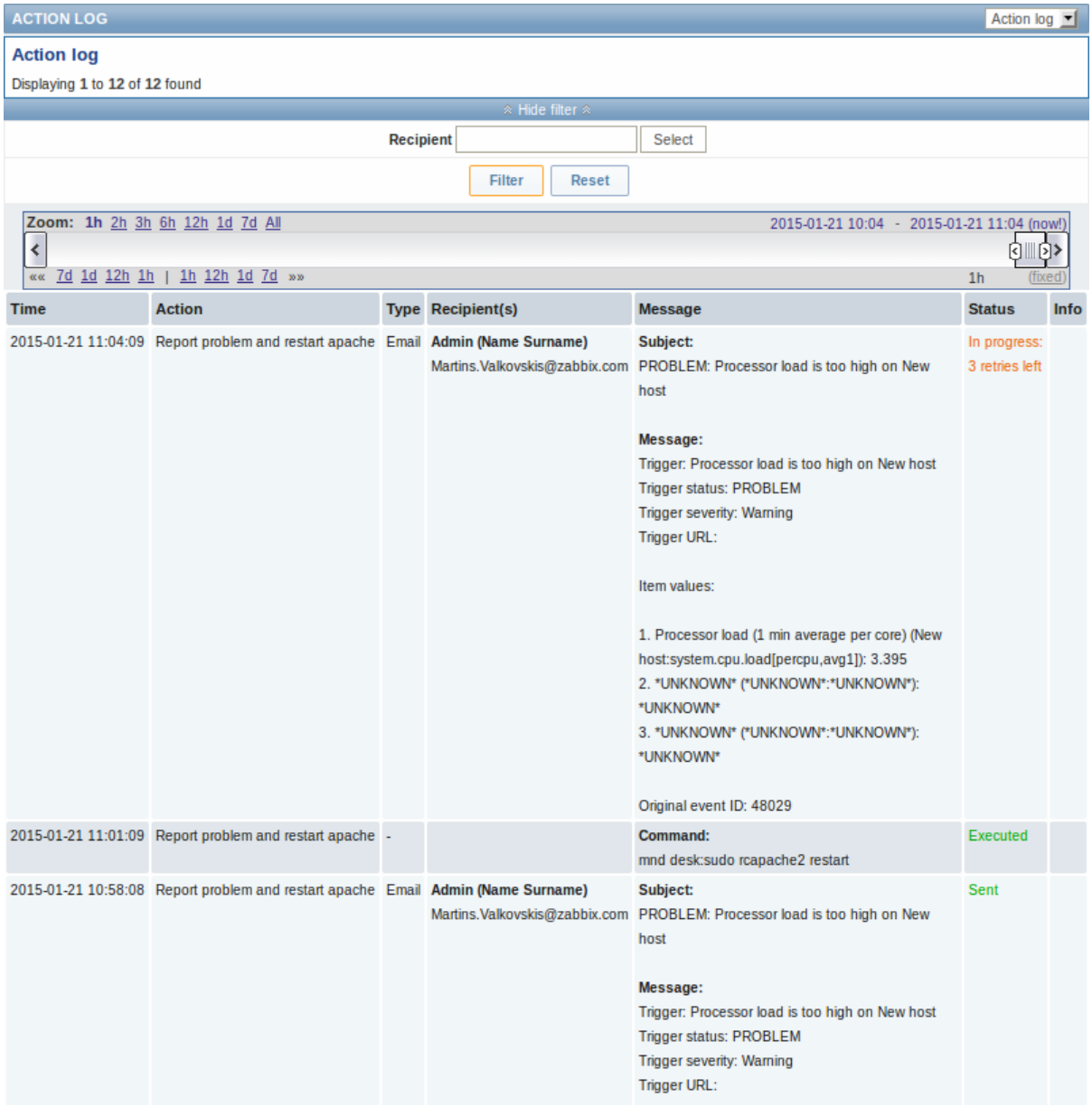
Displayed data:

Column	Description
Time	Timestamp of the audit record.
User	User of the activity.
IP	IP that was used in the activity.
Resource	Affected resource is displayed.
Action	Activity type is displayed - *Login, Logout, Added, Updated, Deleted, Enabled* or *Disabled*.
ID	ID of the affected resource is displayed.
Description	Description of the resource is displayed.
Details	Detailed information on the performed activity is displayed.

Action log

In this screen details of operations (notifications, remote commands) executed within an action are displayed.

You can use the filter, located below the *Action log* bar, to narrow down the records by recipient of e-mail and time period.



Displayed data:

Column	Description
Time	Timestamp of the operation.
Action	Name of the action causing operations is displayed. Action name is displayed since Zabbix **2.4.0**.
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. User alias, name and surname are displayed since Zabbix **2.4.0**.
Message	The content of the message/remote command is displayed.
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.

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, connection problems or problems with agents, some items may get delayed and the information is displayed in this section. For more details, see the Queue section.

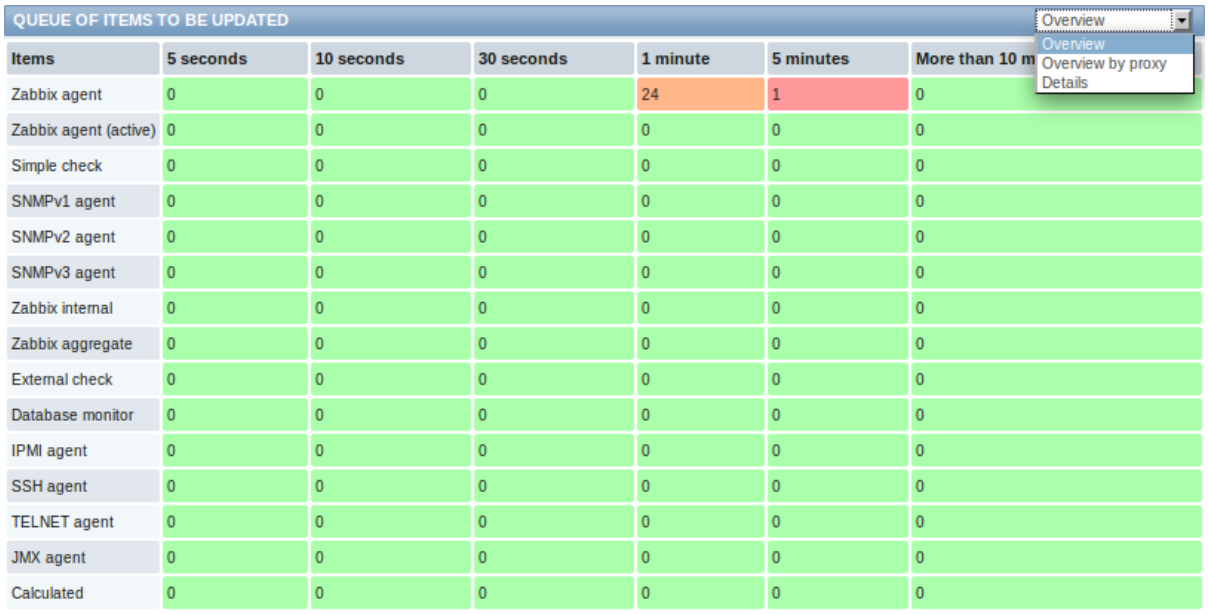
Note:
Queue is available only if Zabbix server is running.

From the dropdown in the upper right corner 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.

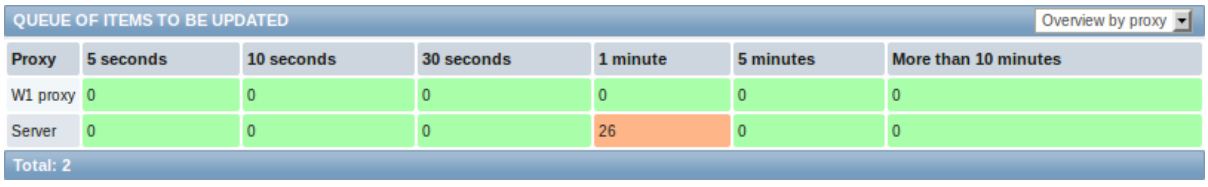


Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 m
Zabbix agent	0	0	0	24	1	0
Zabbix agent (active)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	0	0
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	0	0	0	0	0	0
Zabbix aggregate	0	0	0	0	0	0
External check	0	0	0	0	0	0
Database monitor	0	0	0	0	0	0
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

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.

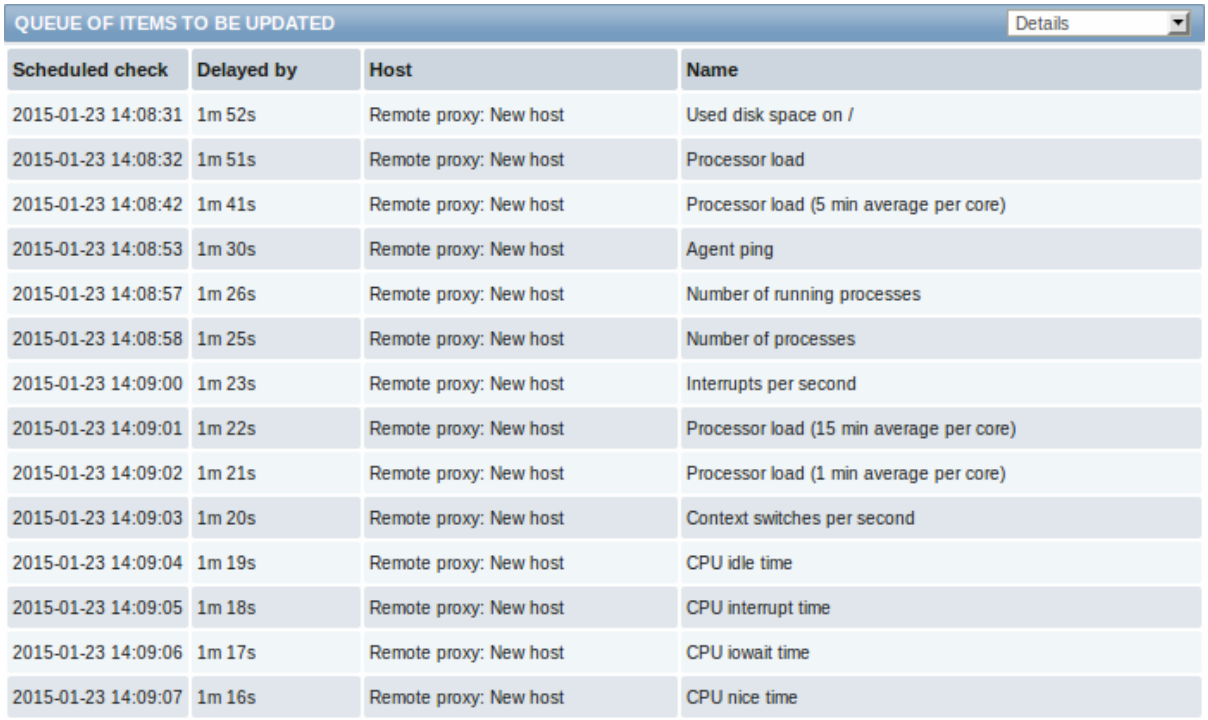


Proxy	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
W1 proxy	0	0	0	0	0	0
Server	0	0	0	26	0	0

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.



In the host column, hosts monitored by proxy are prefixed with the proxy name (since Zabbix 2.4.0).

Displayed data:

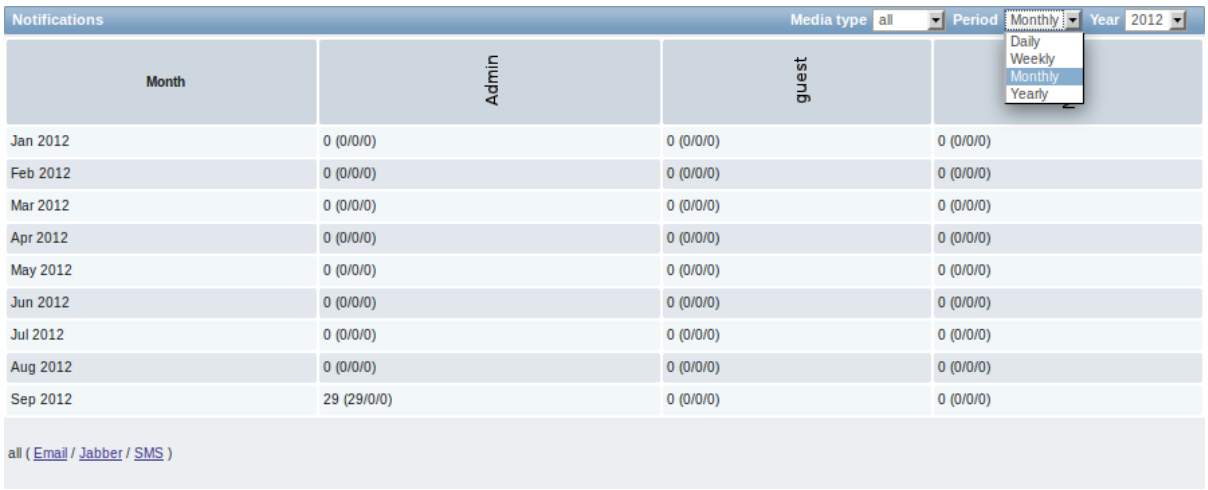
Column	Description
Next check	The time when the check was due is displayed.
Delayed by	The length of the delay is displayed.
Host	Host of the item is displayed.
Name	Name of the waiting item is displayed.

9 Notifications

Overview

In the *Administration* → *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.

10 Installation

Overview

In the *Administration* → *Installation* section Zabbix frontend can be reinstalled.

To continue with the installation, click on *Next*. To exit the installation, click on *Cancel*.

2 User profile

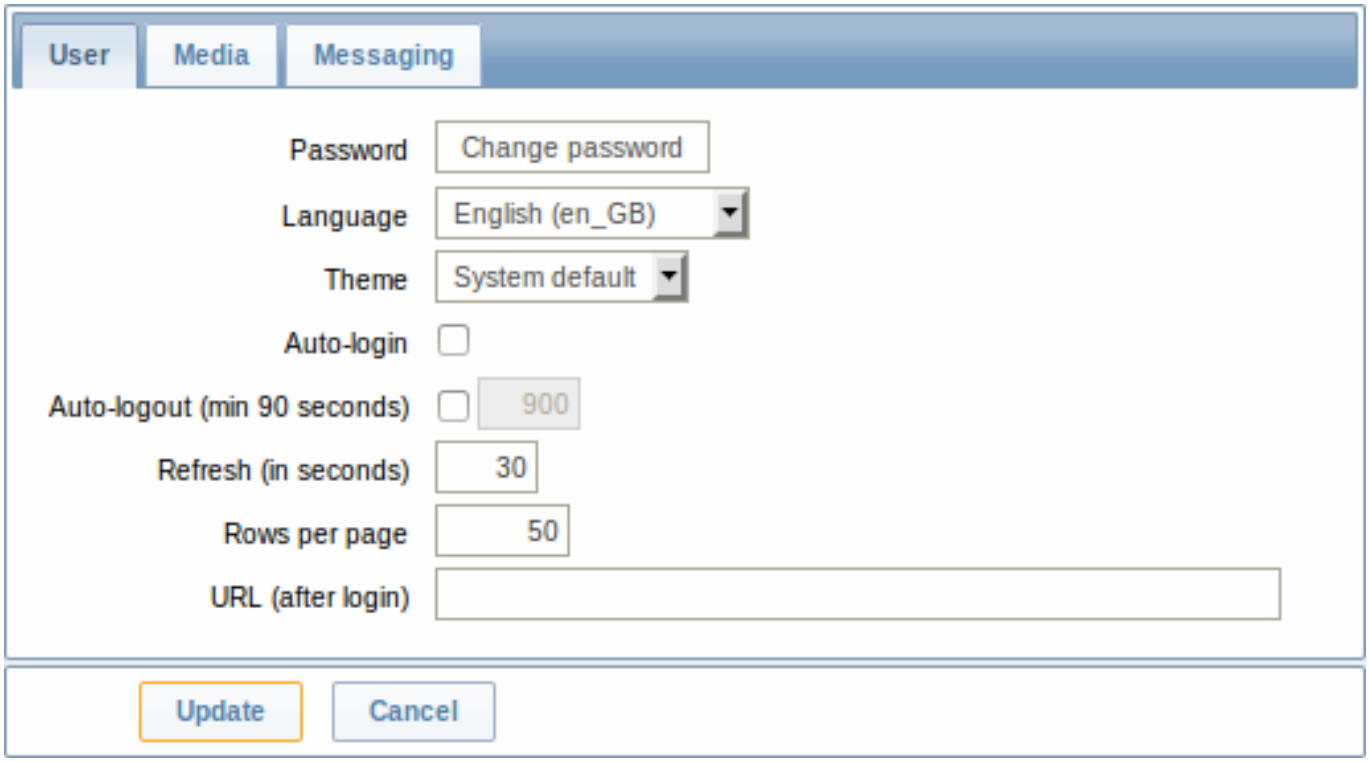
Overview

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 ***Profile*** in the upper right corner of Zabbix window.

Configuration

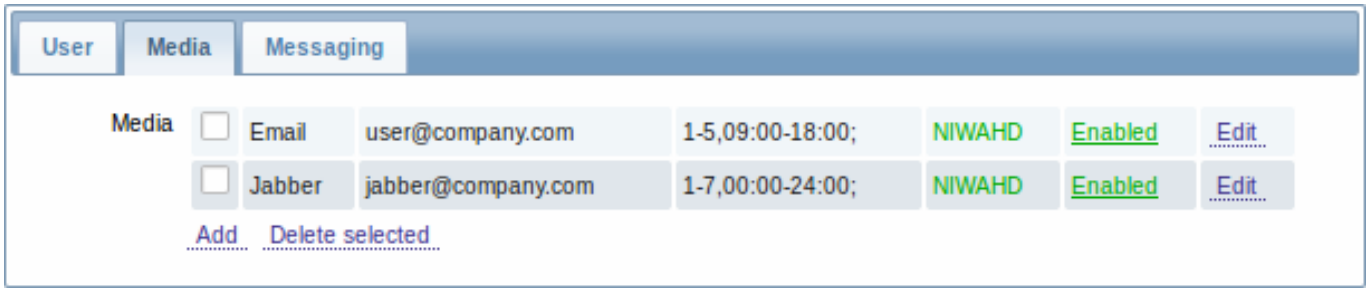
The **User** tab allows you to set various user preferences.



Parameter	Description
Password	Click on the link to display two fields for entering a new password.
Language	Select the interface language of your choice. The php gettext extension is required for the translations to work.
Theme	Select a color theme specifically for your profile.
Auto-login	With this checkbox marked you will be logged in automatically, without having to enter your username and password.
Auto-logout (min 90 seconds)	With this checkbox marked you will be logged out automatically, after the set amount of seconds (minimum 90 seconds). However, this option will not work if the "Show warning if Zabbix server is down" global configuration option is enabled and Zabbix frontend is kept opened or when, for example, Monitoring menu pages perform background information refreshes. Additionally, *Auto-logout* will be deactivated in the user profile when logging in with *Remember me for 30 days* checked.
Refresh (in seconds)	You can set how often the information in the pages will be refreshed on the Monitoring menu, except for Dashboard, which uses its own refresh parameters for every widget.
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*.

Note:
If some language is not available for selection in the user profile it means that a locale for it is not installed on the web server. See the link at the bottom of this page to find out how to install them.

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.

See also

1. How to install additional locales to be able to select unavailable languages in the user profile

1 Global notifications

Overview

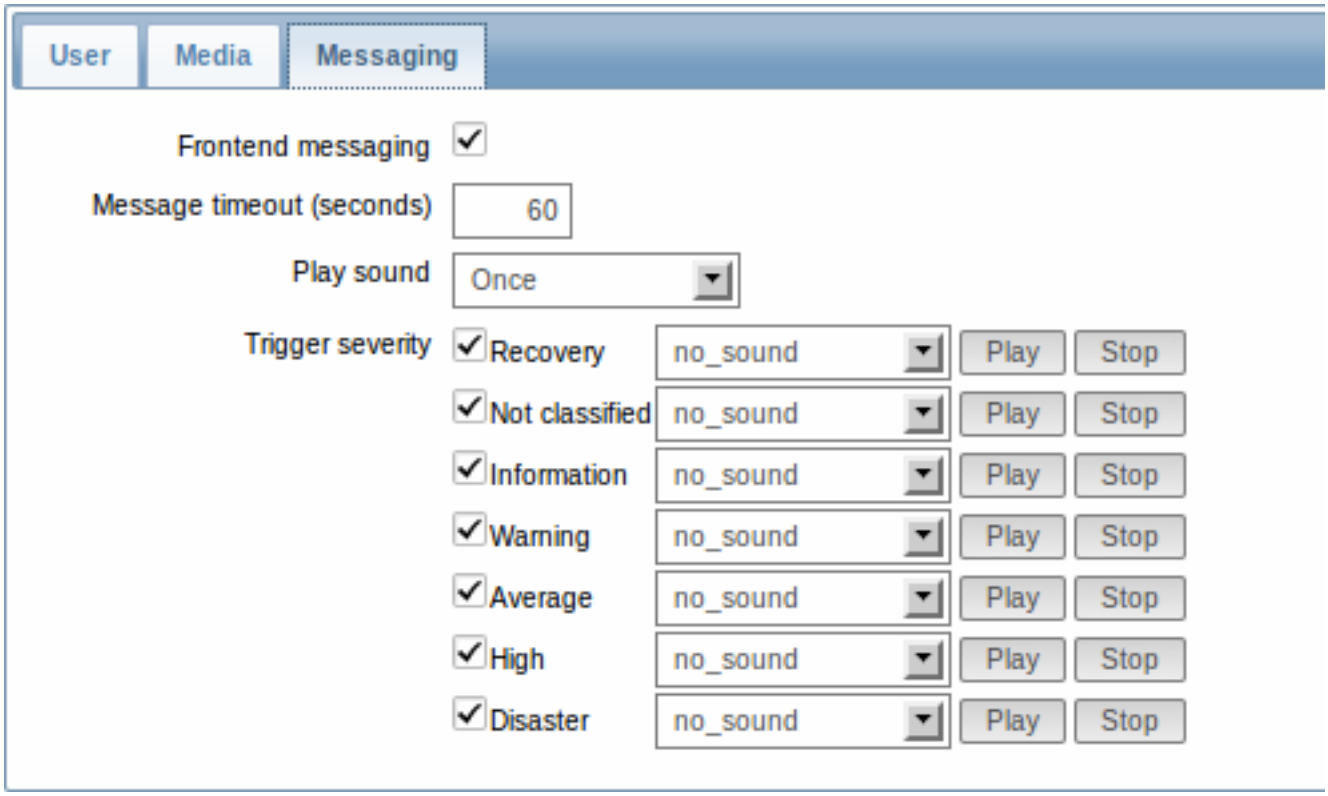
Global notifications are a way of displaying issues that are currently happening right on the screen you're at in Zabbix frontend.

Without global notifications, working in some other location than *Status of triggers* or *Dashboard* pages 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.

Configuration

Global notifications can be enabled per user in the *Messaging* tab of profile configuration.

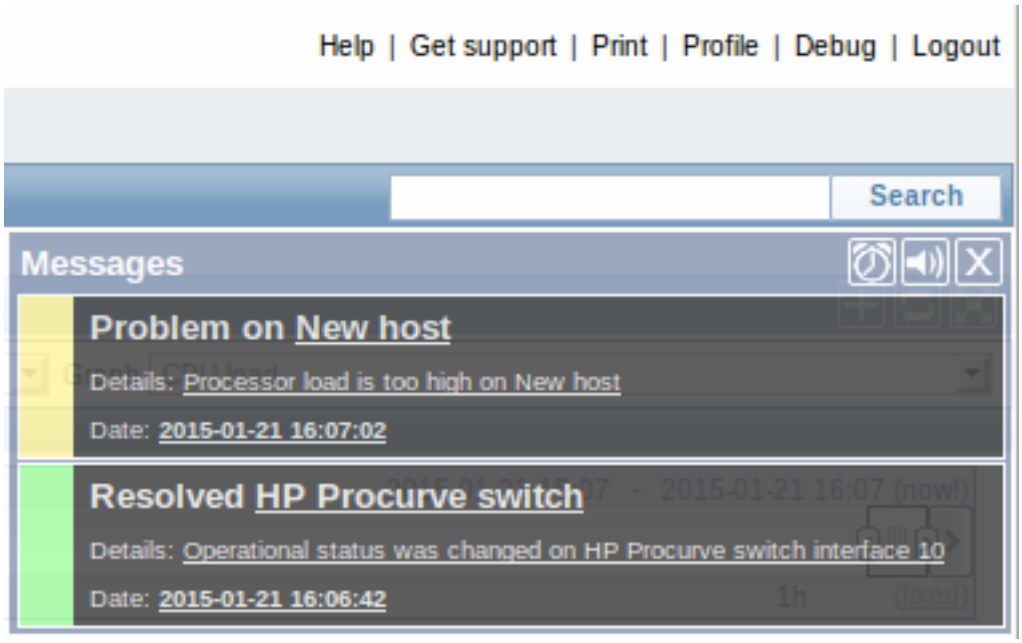


Parameter	Description
Frontend messaging	Mark the checkbox to enable global notifications.
Message timeout	You can set for how long the message will be displayed. By default, messages will stay on screen for 60 seconds.

Parameter	Description
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.

Global messages displayed

As the messages arrive, they are displayed in a floating section on the right hand side. This section can be repositioned vertically by dragging the section header.



For this section, several controls are available:

- **Snooze** button silences currently active alarm sound;
- **Mute/Unmute** button switches between playing and not playing the alarm sounds;
- **Clear** button removes all currently visible messages.

2 Sound in browsers

Overview

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.

The sounds of Zabbix frontend have been successfully tested in the following web browser versions and no additional configuration was required:

- Firefox 3.5.16 on Linux
- Opera 11.01 on Linux
- Google Chrome 9.0 on Windows
- Firefox 3.5.16 on Windows
- IE7 browser on Windows

- Opera v11.01 on Windows
- Chrome v9.0 on Windows
- Safari v5.0 on Windows, but this browser requires *Quick Time Player* to be installed

Additional requirements

Firefox v 3.5.16

For playing `wav` files in the Firefox browser you can use one of the following applications:

- Windows Media Player
- Quick Time plug-in.

Then, in *Tools* → *Options* → *Applications*, in "Wave sound (audio/wav)" set Windows Media Player to play these files.

Safari 5.0

Quick Time Player is required.

Microsoft Internet Explorer

To play sounds in MSIE7 and MSIE8:

- In *Tools* → *Internet Options* → *Advanced* enable *Play sounds in webpages*
- In *Tools* → *Manage Add-ons...* enable **Windows Media Player**
- In the Windows Media Player, in *Tools→Options→File Types* enable *Windows audio file (wav)*

In the Windows Media Player, in Tools→Options tab, "File Types" is only available if the user is a member of "Power Users" or "Administrators" group, i.e. a regular user does not have access to this tab and does not see it.

An additional thing - if IE does not have some *.wav file in the local cache directory (%userprofile%\Local Settings\Temporary Internet Files) the sound will not play the first time.

Known not to work

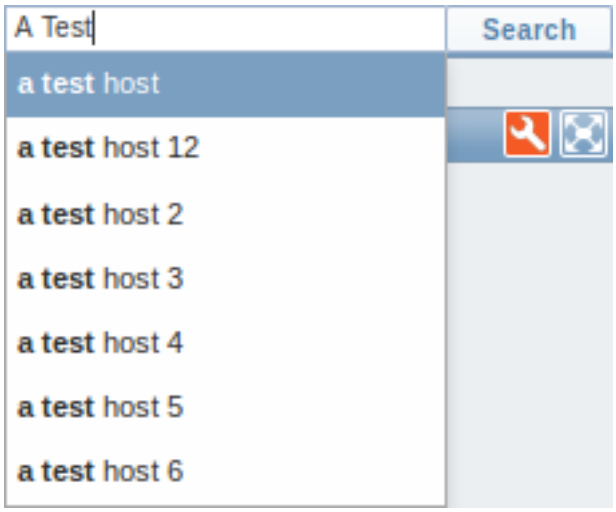
Browsers where the sound did not work:

- Opera 10.11 on Linux.

3 Global search

It is possible to search for various entities in the Zabbix frontend. Search input box is located in the upper right corner. Search can be started by pressing enter or clicking on the *Search* button.

If there is a host that starts with the entered string, a dropdown will appear, listing all such hosts:

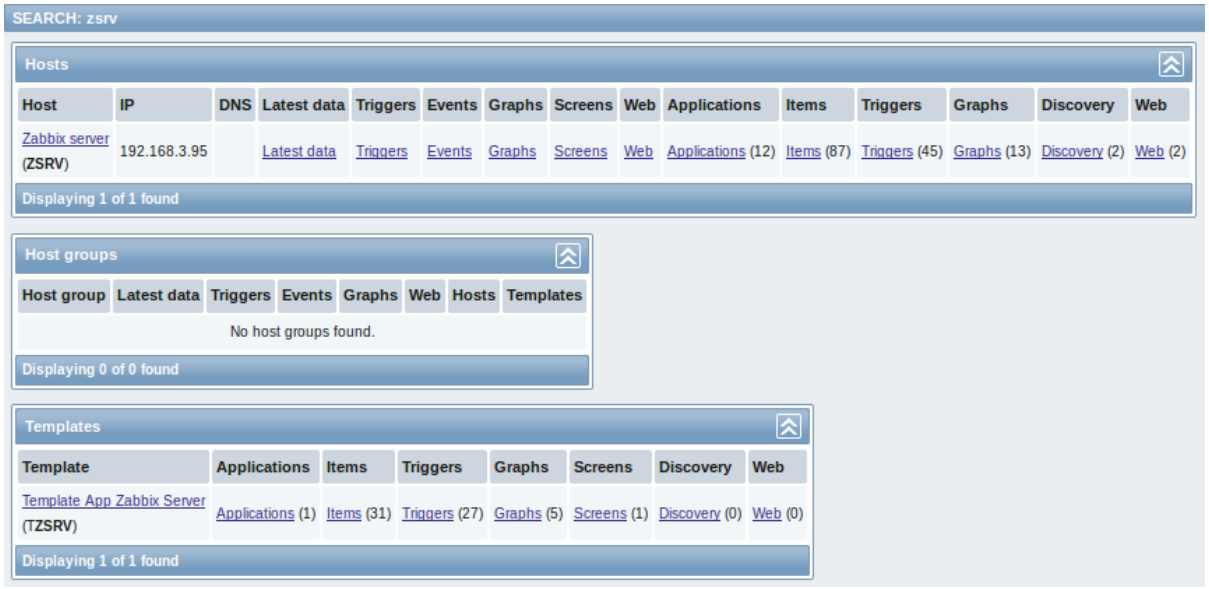


Entities searched It is possible to search for these entities and their properties:

- Hosts
  - Host name
  - Visible name
  - IP address
  - DNS name

- Templates
  - Name
- Host groups
  - Name

In the search results, it is possible to collapse each individual block. Below each block the amount of entities found and displayed is shown, for example, *Displaying 13 of 13 found*. The amount of displayed entries in each block is limited to 100.



For all configuration entities the amount of entities found is displayed in parenthesis. If no entities of that type are found, the entry is displayed without a link.

Enabled hosts are displayed in blue, disabled hosts in red. If the host name of a host/template matches the query string and is different from the visible name, it is displayed below the visible name in parentheses.

Links available For entities found the following links are available:

- Hosts
  - Monitoring
    - Latest data
    - Triggers
    - Events
    - Graphs (*since Zabbix 2.2*)
    - Host screens
    - Web scenarios (*since Zabbix 2.2*)
  - Configuration
    - Host properties
    - Applications
    - Items
    - Triggers
    - Graphs
    - Discovery rules (*since Zabbix 2.2*)
    - Web scenarios (*since Zabbix 2.2*)
- Host groups
  - Monitoring
    - Latest data
    - Triggers
    - Events
    - Graphs (*since Zabbix 2.2*)
    - Web scenarios (*since Zabbix 2.2*)
  - Configuration
    - Host group properties
    - Host group members (hosts and templates; separate links since Zabbix 2.0.2)
- Templates
  - Configuration
    - Template properties

* Applications
* Items
* Triggers
* Graphs
* Template screens
* Discovery rules (*since Zabbix 2.2*)
* Web scenarios (*since Zabbix 2.2*)

4 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 a defined range of IP addresses will be able to work with the frontend normally during maintenance mode.

Configuration

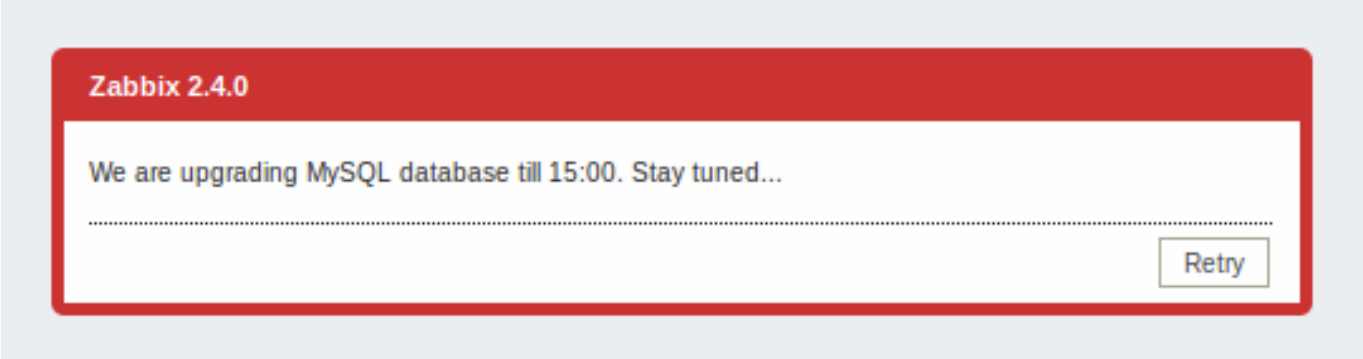
In order to enable maintenance mode, the `maintenance.inc.php` file (located in /conf of the Zabbix HTML document directory on the webserver) must be modified to uncomment the following lines:

```
// Maintenance mode
define('ZBX_DENY_GUI_ACCESS',1);

// IP range, who allowed to connect to FrontEnd
$ZBX_GUI_ACCESS_IP_RANGE = array('127.0.0.1');

// MSG showed on Warning screen!
$_REQUEST['warning_msg'] = 'We are upgrading MySQL database till 15:00. Stay tuned...';
```

Parameter	Details
ZBX_DENY_GUI_ACCESS	Enable maintenance mode: 1 – maintenance mode is enabled, disabled otherwise
ZBX_GUI_ACCESS_IP_RANGE	Connections from these IP addresses will be allowed during the maintenance mode. For example: 192.168.1.1-255
warning_msg	A message you can enter to inform users about the maintenance.

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.

5 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 (&).

Status of triggers

Accessed as *Monitoring* → *Triggers*, page name `tr_status.php`.

Attention:
To set the filter, parameter `filter_set=1` must be passed. Fields that are not specified will be reset to default values.

Generic parameters

- groupid
- hostid
- fullscreen

Page specific parameters

- show_triggers - filter option **Triggers status**, 1 - Recent problem, 2 - Any, 3 - Problem
- show_events - filter option **Events**, 1 - Hide all, 2 - Show all, 3 - Show unacknowledged
- ack_status - filter option **Acknowledge status**, 1 - Any, 2 - With unacknowledged events, 3 - With last event unacknowledged
- show_severity - filter option **Min severity**, 0-5 - corresponding severity
- show_details - filter option **Show details**, 0 - do not show, 1 - show
- status_change_days - filter option **Age less than**, in days
- status_change - filter option **Age less than**, 0 - disabled, 1 - enabled (**status_change_days** will be used)
- txt_select - filter option **Filter by name**, freeform string
- application - filter option **Application**, freeform string
- show_maintenance - filter option **Show hosts in maintenance**, 0 - do not show hosts in maintenance, 1 - show hosts in maintenance

Inventory filter

Since Zabbix 2.4.0, triggers can also be filtered by inventory. Here the syntax is a bit more complicated. Inventory fields and their values are added as zero-based index entries, for example:

```
inventory[0][field]=type_full
inventory[0][value]=Virtual machine
inventory[1][field]=os_full
inventory[1][value]=Linux
```

These must be URL-encoded, though. The passed values would look like:

```
inventory%5B0%5D%5Bfield%5D=type_full
inventory%5B0%5D%5Bvalue%5D=Virtual machine
inventory%5B1%5D%5Bfield%5D=os_full
inventory%5B1%5D%5Bvalue%5D=Linux
```

Inventory field codes can be found in the Zabbix API host object documentation.

Trigger events

Access to events of a specific trigger, which may be useful for notifications is to use a URL like:

```
http://<server_ip_or_name>/zabbix/events.php?triggerid={TRIGGER.ID}&filter_set=1
```

6 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 hour 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*, *Overview* pages and *Data overview* screen element in seconds. By default set to 86400 seconds (24 hours). Unlimited period, if set to 0 seconds.

- 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

- DEFAULT_LATEST_ISSUES_CNT

Controls how many issues are shown in the dashboard's *Last n issues* widget. By default 20 issues are shown.

- SCREEN_REFRESH_TIMEOUT (available since 2.0.4)

Used in screens and defines the timeout seconds for a screen element update. When the defined number of seconds after launching an update pass and the screen element has still not been updated, the screen element will be darkened.

Default: 30

- 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

7 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 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 as *styles/themes/mytheme/main.css*. You can either copy the files from a different theme and create your theme based on it or start from scratch. The rules in the *main.css* file will extend the ones that are defined in the base Zabbix CSS files located in the *styles* folder. Any theme-specific images must be placed in the *styles/themes/mytheme/images* folder.

Step 2

Add your theme to the list of themes returned by the Z::getThemes() method. You can do this by overriding the ZBase::getThemes() method in the Z class. This can be done by adding the following code before the closing brace in *include/classes/core/Z.php*:

```
public static function getThemes() {
    return array_merge(parent::getThemes(), array(
        'mytheme' => _('My theme')
    ));
}
```

Attention:
Note that the name you specify within the first pair of quotes must match the name of the directory under which the theme files have been saved.

To add multiple themes, just list them under the first theme, for example:

```
public static function getThemes() {
    return array_merge(parent::getThemes(), array(
        'mytheme' => _('My 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 colours, entry must be added in the database table *graph_theme*.

Step 3

Activate the new theme.

In Zabbix GUI, 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!

8 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 upper 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.

17. API

Overview Zabbix API allows you to programmatically retrieve and modify the configuration of Zabbix and provides access to historical data. It is widely used to:

- Create new applications to work with Zabbix;
- Integrate Zabbix with third party software;
- Automate routine tasks.

The Zabbix API is a web based API and is shipped as part of the web frontend. It uses the JSON-RPC 2.0 protocol which means two things:

- The API consists of a set of separate methods;
- Requests and responses between the clients and the API are encoded using the JSON format.

More info about the protocol and JSON can be found in the JSON-RPC 2.0 specification and the JSON format homepage.

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 respectfully, but some of the APIs may provide a totally different set of methods.

Performing requests Once you've set up the frontend, you can use remote HTTP requests to call the API. To do that you need to send HTTP POST requests to the `api_jsonrpc.php` file located in the frontend directory. For example, if your Zabbix frontend is installed under *http://company.com/zabbix*, the HTTP request to call the `apiinfo.version` method may look like this:

```
POST http://company.com/zabbix/api_jsonrpc.php HTTP/1.1
Content-Type: application/json-rpc

{"jsonrpc":"2.0","method":"apiinfo.version","id":1,"auth":null,"params":{}}
```

The request must have the `Content-Type` header set to one of these values: `application/json-rpc`, `application/json` or `application/jsonrequest`.

> **Note:**
> You can use any HTTP client or a JSON-RPC testing tool to perform API requests manually, but for developing applications we suggest you use one of the community maintained libraries.

Example workflow The following section will walk you through some usage examples in more detail.

Authentication Before you can access any data inside of Zabbix you'll need to log in and obtain an authentication token. This can be done using the user.login method. Let us suppose that you want to log in as a standard 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 $1",
        "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, starting from 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

Events

Retrieve events generated by triggers, network discovery and other Zabbix systems for more flexible situation management or third-party tool integration.

Event API

Service monitoring

Retrieve detailed service layer availability information about any IT service.

IT service SLA calculation

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

Screens

Edit global and template-level screens or each screen item individually.

Screen API | Screen item API | Template screen API | Template screen item 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

IT services

Manage IT services for service-level monitoring and retrieve detailed SLA information about any service.

IT service 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 | Discovery host API | Discovery 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 | Media API

General

Change certain global configuration options.

Icon map API | Image API | User macro 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.exists - check if an action exists
- 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)	integer	Default operation step duration. Must be greater than 60 seconds.
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.
def_longdata	string	Problem message text.
def_shortdata	string	Problem message subject.
r_longdata	string	Recovery message text.
r_shortdata	string	Recovery message subject.
recovery_msg	integer	Whether recovery messages are enabled. Possible values: 0 - *(default)* disabled; 1 - enabled.
status	integer	Whether the action is enabled or disabled. Possible values: 0 - *(default)* enabled; 1 - disabled.

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.
actionid	string	ID of the action that the operation belongs to.
esc_period	integer	Duration of an escalation step in seconds. Must be greater than 60 seconds. If set to 0, the default action escalation period will be used. Default: 0.
esc_step_from	integer	Step to start escalation from. Default: 1.

Property	Type	Description
esc_step_to	integer	Step to end escalation at. Default: 1.
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.
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.
opconditions	array	Required for remote command operations if `opcommand_grp` is not set. 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.
opmessage	object	Required for ”add to host group” and ”remove from host group” operations. Object containing the data about the message sent by the operation. The operation message object is described in detail below.
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.
		Required for message operations if `opmessage_usr` is not set.

Property	Type	Description
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.
optemplate	array	Templates to link the hosts to 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.

Action operation command

The operation command object contains data about the command that will be run by the operation.

Property	Type	Description
operationid	string	*(readonly)* ID of the operation.
command (required)	string	Command to run.
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. 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.

Property	Type	Description
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.
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
operationid	string	*(readonly)* ID of the action operation.
default_msg	integer	Whether to use the default action message text and subject. Possible values: 0 - *(default)* use the data from the operation; 1 - use the data from the action.
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 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.
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; 5 - trigger value; 6 - time period; 13 - host template; 15 - application; 16 - maintenance status. 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 auto-registration 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.
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.
operator	integer	Condition operator. Possible values: 0 - *(default)* =; 1 - <>; 2 - like; 3 - not like; 4 - in; 5 - >=; 6 - <=; 7 - not in.

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	=, <>	Host group ID.
1	Host	=, <>	Host ID.
2	Trigger	=, <>	Trigger ID.
3	Trigger name	like, not like	Trigger name.
4	Trigger severity	=, <>, >=, <=	Trigger severity. Refer to the trigger ”severity” property for a list of supported trigger severities.
5	Trigger value	=	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	=, <>	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	=, <>	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	=, <>	One or several port ranges separated by commas.
10	Discovery status	=	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	>=, <=	Time indicating how long has the discovered object been in the current status in seconds.
12	Received values	=, <>, >=, <=, like, not like	Value returned when performing a Zabbix agent, SNMPv1, SNMPv2 or SNMPv3 discovery check.
13	Host template	=, <>	Linked template ID.
15	Application	=, like, not like	Name of the application.
16	Maintenance status	in, not in	No value required: using the ”in” operator means that the host must be in maintenance, ”not in” - not in maintenance.
18	Discovery rule	=, <>	ID of the discovery rule.
19	Discovery check	=, <>	ID of the discovery check.
20	Proxy	=, <>	ID of the proxy.

Condition	Condition name	Supported operators	Expected value
21	Discovery object	=	Type of object that triggered the discovery event. Possible values: 1 - discovered host; 2 - discovered service.
22	Host name	like, not like	Host name.
23	Event type	=	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.
24	Host metadata	like, not like	Metadata of the auto-registered host.

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
operations (required)	array	Action operations to create for the action.
filter	object	Action filter object 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 an action that will be run when a trigger from host ”30045” that has the word ”memory” in its name goes into problem state. The action must first send a message to all users in user group ”7”. If the event is not resolved in 4 minutes, it will run script ”3” on all hosts in group ”2”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Trigger action",
        "eventsource": 0,
        "status": 0,
```

```
        "esc_period": 120,
        "def_shortdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}",
        "def_longdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}\r\nLast value: {ITEM.LASTVALUE}\r\n\r\n{TRIGGER.
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10084"
                },
                {
                    "conditiontype": 3,
                    "operator": 2,
                    "value": "memory"
                }
            ]
        },
        "operations": [
            {
                "operationtype": 0,
                "esc_period": 0,
                "esc_step_from": 1,
                "esc_step_to": 2,
                "evaltype": 0,
                "opmessage_grp": [
                    {
                        "usrgrpid": "7"
                    }
                ],
                "opmessage": {
                    "default_msg": 1,
                    "mediatypeid": "1"
                }
            },
            {
                "operationtype": 1,
                "esc_step_from": 3,
                "esc_step_to": 4,
                "evaltype": 0,
                "opconditions": [
                    {
                        "conditiontype": 14,
                        "operator": 0,
                        "value": "0"
                    }
                ],
                "opcommand_grp": [
                    {
                        "groupid": "2"
                    }
                ],
                "opcommand": {
                    "type": 4,
                    "scriptid": "3"
                }
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "17"
        ]
    },
    "id": 1
}
```

Create a discovery action

Create an action that will link discovered hosts to template ”30085”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Discovery action",
        "eventsource": 1,
        "status": 0,
        "esc_period": 0,
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "conditiontype": 21,
                    "value": "1"
                },
                {
                    "conditiontype": 10,
                    "value": "2"
                }
            ]
        },
        "operations": [
            {
                "esc_step_from": 1,
                "esc_period": 0,
                "optemplate": [
                    {
                        "templateid": "10091"
                    }
                ],
                "operationtype": 6,
                "esc_step_to": 1
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "18"
        ]
    },
```

```
    "id": 1
}
```

Using a custom expression filter

Create a trigger action that will use a custom filter condition. The action must send a message for each trigger with severity higher or equal to ”Warning” for hosts ”10084” and ”10106”. The formula IDs ”A”, ”B” and ”C” have been chosen arbitrarily.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Trigger action",
        "eventsource": 0,
        "status": 0,
        "esc_period": 120,
        "def_shortdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}",
        "def_longdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}\r\nLast value: {ITEM.LASTVALUE}\r\n\r\n{TRIGGER.
        "filter": {
            "evaltype": 3,
            "formula": "A and (B or C)",
            "conditions": [
                {
                    "conditiontype": 4,
                    "operator": 5,
                    "value": "2",
                    "formulaid": "A"
                },
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10084",
                    "formulaid": "B"
                },
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10106",
                    "formulaid": "C"
                }
            ]
        },
        "operations": [
            {
                "operationtype": 0,
                "esc_period": 0,
                "esc_step_from": 1,
                "esc_step_to": 2,
                "evaltype": 0,
                "opmessage_grp": [
                    {
                        "usrgrpid": "7"
                    }
                ],
                "opmessage": {
                    "default_msg": 1,
                    "mediatypeid": "1"
                }
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "18"
        ]
    },
    "id": 1
}
```

See also

- Action filter
- Action operation

Source

CAction::create() in *frontends/php/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 *frontends/php/include/classes/api/services/CAction.php*.

action.exists

Description

`boolean action.exists(object filter)`

This method checks if at least one action that matches the given filter criteria exists.

> **Warning:**
> This method is deprecated and will be removed in the future. Please use action.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
actionid	string/array	IDs of actions.
name	string/array	Names of actions.

Return values

`(boolean)` Returns `true` if at least one action that matches the given filter criteria exists.

Examples

Check action by name

Check if an action named "Auto discovery. Linux servers." exists.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.exists",
    "params": {
        "name": "Auto discovery. Linux servers."
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CAction::exists() in *frontends/php/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	Returns the action filter in the `filter` property.
selectOperations	query	Return action operations in the `operations` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `actionid`, `name` and `status`.
countOutput	flag	These parameters being common for all `get` methods are described in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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 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": "0",
            "def_shortdata": "",
            "def_longdata": "",
            "recovery_msg": "0",
            "r_shortdata": "",
            "r_longdata": "",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [
                    {
                        "conditiontype": "10",
                        "operator": "0",
                        "value": "0",
                        "formulaid": "B"
                    },
                    {
                        "conditiontype": "8",
                        "operator": "0",
                        "value": "9",
                        "formulaid": "C"
                    },
                    {
                        "conditiontype": "12",
                        "operator": "2",
                        "value": "Linux",
                        "formulaid": "A"
                    }
                ],
                "eval_formula": "A and B and C"
            },
            "operations": [
                {
                    "operationid": "1",
                    "actionid": "2",
                    "operationtype": "6",
                    "esc_period": "0",
                    "esc_step_from": "1",
                    "esc_step_to": "1",
                    "evaltype": "0",
                    "opconditions": [],
                    "optemplate": [
                        {
                            "operationid": "1",
                            "templateid": "10001"
                        }
                    ]
                },
                {
```

```
                    "operationid": "2",
                    "actionid": "2",
                    "operationtype": "4",
                    "esc_period": "0",
                    "esc_step_from": "1",
                    "esc_step_to": "1",
                    "evaltype": "0",
                    "opconditions": [],
                    "opgroup": [
                        {
                            "operationid": "2",
                            "groupid": "2"
                        }
                    ]
                }
            ]
        }
    ],
    "id": 1
}
```


See also

- Action filter
- Action operation

Source

CAction::get() in *frontends/php/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.

Return values

`(object)` Returns an object containing the IDs of the updated actions under the `actionids` property.

Examples

Disable action

Disable action, that is, set its status to "1".

Request:


```
{
    "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 *frontends/php/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.
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.

Property	Type	Description
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. Possible values for command alerts: 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.

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.
eventobject	integer	Return only alerts generated by events related to objects of the given type. Refer to the event ”object” property for a list of supported object types. Default: 0 - trigger.
eventsource	integer	Return only alerts generated by events of the given type. Refer to the event ”source” property for a list of supported event types. Default: 0 - trigger events.

Parameter	Type	Description
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 the hosts that triggered the action operation in the `hosts` property.
selectMediatypes	query	Return the media type that was used for the message alert as an array in the `mediatypes` property.
selectUsers	query	Return the user that the message was addressed to as an array in the `users` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `alertid`, `clock`, `eventid` and `status`.
countOutput	flag	These parameters being common for all `get` methods are described in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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"
        }
    ],
    "id": 1
}
```


See also

- Host
- Media type
- User

Source

CAlert::get() in *frontends/php/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": "2.4.0",
    "id": 1
}
```

Source

CAPIInfo::version() in *frontends/php/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.exists - checking if applications exist
- 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
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 *frontends/php/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 *frontends/php/include/classes/api/services/CApplication.php*.

application.exists

Description

`boolean application.exists(object filter)`

This method checks if at least one application that matches the given filter criteria exists.

Warning:
This method is deprecated and will be removed in the future. Please use application.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
hostid	string/array	IDs of the hosts the applications must belong to.
name	string/array	Names of the applications

Return values

`(boolean)` Returns `true` if at least one application that matches the given filter criteria exists.

Examples

Check application on host

Check if application ”Memory” exists on host ”10084.”

Request:

```
{
    "jsonrpc": "2.0",
    "method": "application.exists",
    "params": {
        "hostid": "10084",
        "name": "Memory"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CApplication::exists() in *frontends/php/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 the host that the application belongs to in the `host` property.
selectItems	query	Return the items contained in the application in the `items` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `applicationid` and `name`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	
~~expandData~~ (deprecated)	flag	Return the `name` of the host that the application belongs to as a property of the application object.
~~selectHosts~~ (deprecated)	query	Return the hosts that the application belongs to in the `hosts` property.

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 *frontends/php/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 *frontends/php/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 *frontends/php/include/classes/api/services/CApplication.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. `screens` - `(array)` IDs of screens to export; `templates` - `(array)` IDs of templates to export;

Return values

`(string)` Returns a serialized string containing the requested configuration data.

Examples

Exporting a host

Export the configuration of a host as an XML string.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "configuration.export",
    "params": {
        "options": {
            "hosts": [
                "10161"
            ]
        },
        "format": "xml"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "<!--?xml version=\"1.0\" encoding=\"UTF-8\"?-->\n<zabbix_export><version>2.0</version><date>
    "id": 1
}
```

Source

CConfiguration::export() in *frontends/php/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`; `updateExisting` - `(boolean)` if set to `true`, existing applications will be updated; 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`.

Parameter	Type	Description
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`.
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`.
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`.
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`.
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`.

Parameter	Type	Description
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`.
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`.

Return values

`(boolean)` Returns `true` if importing has been successful.

Examples

Importing hosts and items

Import the host and items contained in the XML string. If any items in XML are missing, they will be deleted from the database, and everything else will be left unchanged.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "configuration.import",
    "params": {
        "format": "xml",
        "rules": {
            "hosts": {
                "createMissing": true,
                "updateExisting": true
            },
            "items": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            }
        },
        "source": "<!--?xml version=\"1.0\" encoding=\"UTF-8\"?-→<zabbix_export><version>2.0</version><dat
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CConfiguration::import() in *frontends/php/include/classes/api/services/CConfiguration.php*.

Discovered host

This class is designed to work with discovered hosts.

Object references:

- Discovered host

Available methods:

- dhost.exists - check if a discovered host exists
- 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.
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.exists

Description

`boolean dhost.exists(object filter)`

This method checks if at least one discovered host that matches the given filter criteria exists.

Warning:
This method is deprecated and will be removed in the future. Please use dhost.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
dhostid	string/array	IDs of the discovered hosts.

Return values

(boolean) Returns true if at least one discovered host that matches the given filter criteria exists.

Examples

Check multiple discovered hosts

Check if discovered hosts with IDs "1" and "2" exists.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dhost.exists",
    "params": {
        "dhostid": [
            "1",
            "2"
        ]
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CDHost::exists() in *frontends/php/include/classes/api/services/CDHost.php*.

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 the discovery rule that detected the host as an array in the drules property.
selectDServices	query	Return the discovered services running on the host in the dservices property. 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.

Parameter	Type	Description
countOutput	flag	These parameters being common for all get methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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 *frontends/php/include/classes/api/services/CDHost.php*.

Discovered service

This class is designed to work with discovered services.

Object references:

- Discovered service

Available methods:

- dservice.exists - check if a discovered service exists
- 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.
key_	string	Key used by a Zabbix agent discovery check to locate 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.

Property	Type	Description
status	integer	Status of the service. Possible values: 0 - service up; 1 - service down.
type	integer	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.
value	string	Value returned by the service when performing a Zabbix agent, SNMPv1, SNMPv2 or SNMPv3 discovery check.

dservice.exists

Description

`boolean dservice.exists(object filter)`

This method checks if at least one discovered service that matches the given filter criteria exists.

Warning:
This method is deprecated and will be removed in the future. Please use dservice.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
dserviceid	string/array	IDs of discovered services.

Return values

`(boolean)` Returns `true` if at least one discovered service that matches the given filter criteria exists.

Examples

Check multiple discovered services

Check if discovered services with IDs "121" and "73" exist.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dservice.exists",
    "params": {
        "dserviceid": [
            "121",
            "73"
        ]
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
```

```
    "id": 1
}
```

Source

CDService::exists() in *frontends/php/include/classes/api/services/CDService.php*.

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 the discovery rule that detected the service as an array in the `drules` property.
selectDHosts	query	Return the discovered host that service belongs to as an array in the `dhosts` property.
selectHosts	query	Return the hosts with the same IP address as the service in the `hosts` property. 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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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",
            "type": "4",
            "key_": "",
            "value": "",
            "port": "80",
            "status": "1",
            "lastup": "0",
            "lastdown": "1348650607",
            "dcheckid": "5",
            "ip": "192.168.1.134",
            "dns": "john.local"
        },
        {
            "dserviceid": "13",
            "dhostid": "11",
            "type": "3",
            "key_": "",
            "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 *frontends/php/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	ID of the discovery rule that the check belongs to.
key_	string	The value of this property differs depending on the type 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.
snmpv3_authpassphrase	string	Auth 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	Priv 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.

Property	Type	Description
type	integer	Type of check. Possible values: 0 - *(default)* 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.

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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	

Parameter	Type	Description
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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"
        }
    ],
    "id": 1
}
```

Source

CDCheck::get() in *frontends/php/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.exists - check if a discovery rule exists
- drule.get - retrieve discovery rules
- drule.isreadable - check if discovery rules are readable
- drule.iswritable - check if discovery rules are writable
- 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	integer	Execution interval of the discovery rule in seconds. Default: 3600.
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.

drule.create

Description

`object drule.create(object/array discroveryRules)`

This method allows to create new discrovery rules.

Parameters

`(object/array)` Discrovery rules to create.

Additionally to the standard discrovery 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 discrovery rules under the druleids property. The order of the returned IDs matches the order of the passed discrovery 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 *frontends/php/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 *frontends/php/include/classes/api/services/CDRule.php*.

drule.exists

Description

`boolean drule.exists(object filter)`

This method checks if at least one discrovery rule that matches the given filter criteria exists.

> **Warning:**
> This method is deprecated and will be removed in the future. Please use drule.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
druleids	string/array	IDs of discovery rules.
name	string/array	Names of discovery rules.

Return values

`(boolean)` Returns `true` if at least one discrovery rule that matches the given filter criteria exists.

Examples

Check a discovery rule by name

Check if a discovery rule called ”Local network” exists.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "drule.exists",
    "params": {
        "name": "Local network"
```

```
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- drule.isreadable
- drule.iswritable

Source

CDRule::exists() in *frontends/php/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 discovery checks used by the discovery rule in the `dchecks` property. Supports `count`.
selectDHosts	query	Return the discovered hosts that the discovery rule created in the `dhosts` property. 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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	

Parameter	Type	Description
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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": "5",
            "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"
                },
                {
                    "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"
                }
            ]
        },
        {
            "druleid": "6",
            "proxy_hostid": "0",
            "name": "Zabbix agent discovery",
            "iprange": "192.168.1.1-255",
            "delay": "3600",
            "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"
                }
            ]
        }
    ],
    "id": 1
}
```


See also

- Discovered host
- Discovery check

Source

CDRule::get() in *frontends/php/include/classes/api/services/CDRule.php*.

drule.isreadable

Description

`boolean drule.isreadable(array discoveryRuleIds)`

This method checks if the given discovery rules are available for reading.

Parameters

`(array)` IDs of the discovery rules to check.

Return values

(boolean) Returns true if the given discovery rules are available for reading.

Examples

Check multiple discovery rules

Check if the two discovery rules are readable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "drule.isreadable",
    "params": [
        "5",
        "8"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- drule.exists
- drule.iswritable

Source

CDRule::isReadable() in *frontends/php/include/classes/api/services/CDRule.php*.

drule.iswritable

Description

`boolean drule.iswritable(array discoveryRuleIds)`

This method checks if the given discovery rules are available for writing.

Parameters

(array) IDs of the discovery rules to check.

Return values

(boolean) Returns true if the given discovery rules are available for writing.

Examples

Check multiple discovery rules

Check if the two discovery rules are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "drule.iswritable",
    "params": [
        "5",
        "8"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- drule.isreadable
- drule.exists

Source

CDRule::isWritable() in *frontends/php/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 *frontends/php/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.
acknowledged	integer	Whether the event has been acknowledged.
clock	timestamp	Time when the event was created.
ns	integer	Nanoseconds when the event was created.
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 auto-registration events: 3 - auto-registered host. Possible values for internal events: 0 - trigger; 4 - item; 5 - LLD rule.
objectid	string	ID of the related object.

Property	Type	Description
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 auto-registration; 3 - internal event.
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 auto-registration events.

event.acknowledge

Description

`object event.acknowledge(object/array parameters)`

This method allows to acknowledge events and add an acknowledgement message. If an event is already acknowledged, a new message will still be added.

Attention:
Only trigger events can be acknowledged.

Parameters

`(object/array)` Parameters containing the IDs of the events acknowledge and a message.

Parameter	Type	Description
eventids (required)	string/object	IDs of the events to acknowledge.
message	string	Text of the acknowledgement message.

Return values

`(object)` Returns an object containing the IDs of the acknowledged 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",
        "message": "Problem resolved."
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": [
            "20427"
        ]
    },
    "id": 1
}
```

Source

CEvent::acknowledge() in *frontends/php/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.

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.
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.
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.
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.
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.

Parameter	Type	Description
value	integer/array	Return only events with the given values.
selectHosts	query	Return hosts containing the object that created the event in the `hosts` property. Supported only for events generated by triggers, items or LLD rules.
selectRelatedObject	query	Return the object that created the event in the `relatedObject` property. The type of object returned depends on the event type.
select_alerts	query	Return alerts generated by the event in the `alerts` property. Alerts are sorted in reverse chronological order.
select_acknowledges	query	Return event's acknowledges in the `acknowledges` property. Acknowledges are sorted in reverse chronological order. The event acknowledgement object has the following properties: `acknowledgeid` - `(string)` acknowledgement's ID; `userid` - `(string)` ID of the user that acknowledged the event; `eventid` - `(string)` ID of the acknowledged event; `clock` - `(timestamp)` time when the event was acknowledged; `message` - `(string)` text of the acknowledgement message; `alias` - `(string)` alias of the user that acknowledged the event; `name` - `(string)` name of the user that acknowledged the event; `surname` - `(string)` surname of the user that acknowledged the event. Supports `count`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `eventid`, `objectid` and `clock`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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",
        "objectids": "13926",
        "sortfield": ["clock", "eventid"],
        "sortorder": "DESC"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "acknowledges": [
                {
                    "acknowledgeid": "1",
                    "userid": "1",
                    "eventid": "9695",
                    "clock": "1350640590",
                    "message": "Problem resolved.\n\r----[BULK ACKNOWLEDGE]----",
                    "alias": "Admin"
                }
            ],
            "eventid": "9695",
            "source": "0",
            "object": "0",
            "objectid": "13926",
            "clock": "1347970410",
            "value": "1",
            "acknowledged": "1",
            "ns": "413316245"
        },
        {
            "acknowledges": [],
            "eventid": "9671",
            "source": "0",
            "object": "0",
            "objectid": "13926",
            "clock": "1347970347",
            "value": "0",
            "acknowledged": "0",
            "ns": "0"
        }
    ],
    "id": 1
}
```

Retrieving events by time period

Retrieve all events that have been created between October 9 and 10, 2012, in reverse chronological order.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "event.get",
    "params": {
        "output": "extend",
```

```
        "time_from": "1349797228",
        "time_till": "1350661228",
        "sortfield": ["clock", "eventid"],
        "sortorder": "desc"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "eventid": "20616",
            "source": "0",
            "object": "0",
            "objectid": "14282",
            "clock": "1350477814",
            "value": "1",
            "acknowledged": "0",
            "ns": "0"
        },
        {
            "eventid": "20617",
            "source": "0",
            "object": "0",
            "objectid": "14283",
            "clock": "1350477814",
            "value": "0",
            "acknowledged": "0",
            "ns": "0"
        },
        {
            "eventid": "20618",
            "source": "0",
            "object": "0",
            "objectid": "14284",
            "clock": "1350477815",
            "value": "1",
            "acknowledged": "0",
            "ns": "0"
        }
    ],
    "id": 1
}
```

See also

- Alert
- Item
- Host
- LLD rule
- Trigger

Source

CEvent::get() in *frontends/php/include/classes/api/services/CEvent.php*.

Graph

This class is designed to work with items.

Object references:

- Graph

Available methods:

- graph.create - creating new graphs
- graph.delete - deleting graphs
- graph.exists - checking if graphs exists
- graph.get - retrieving graphs
- graph.getobjects - retrieving graphs by filters
- 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.
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.
templateid	string	*(readonly)* ID of the parent template graph.

Property	Type	Description
yaxismax	float	The fixed maximum value for the Y axis. Default: 100.
yaxismin	float	The fixed minimum value for the Y axis. 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.
ymin_type	integer	Minimum value calculation method for the Y axis. Possible values: 0 - *(default)* calculated; 1 - fixed; 2 - item.

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",
                "sortorder": "0"
            },
            {
                "itemid": "22829",
                "color": "3333FF",
                "sortorder": "1"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652"
        ]
    },
    "id": 1
}
```

See also

- Graph item

Source

CGraph::create() in *frontends/php/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 *frontends/php/include/classes/api/services/CGraph.php*.

graph.exists

Description

`boolean graph.exists(object filter)`

This method checks if at least one graph that matches the given filter criteria exists.

Warning:
This method is deprecated and will be removed in the future. Please use graph.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
host	string/array	Technical names of the hosts that the graphs belong to.
hostids	string/array	IDs of the hosts that the graphs belong to.
name	string/array	Names of the graphs.

Return values

`(boolean)` Returns `true` if at least one graph that matches the given filter criteria exists.

Examples

Checking graph by name

Check if a graph named ”CPU utilization” already exists on host ”Zabbix server”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graph.exists",
    "params": {
        "name": "CPU utilization",
        "host": "Zabbix server"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CGraph::exists() in *frontends/php/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 the host groups that the graph belongs to in the `groups` property.
selectTemplates	query	Return the templates that the graph belongs to in the `templates` property.
selectHosts	query	Return the hosts that the graph belongs to in the `hosts` property.
selectItems	query	Return the items used in the graph in the `items` property.
selectGraphDiscovery	query	Return the graph discovery object in the `graphDiscovery` property. 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 the graph items used in the graph in the `gitems` property.
selectDiscoveryRule	query	Return the low-level discovery rule that created the graph in the `discoveryRule` 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 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`.

Parameter	Type	Description
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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": "0.0000",
            "yaxismax": "100.0000",
            "templateid": "439",
            "show_work_period": "1",
            "show_triggers": "1",
            "graphtype": "0",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
```

```
            "flags": "0"
        },
        {
            "graphid": "613",
            "name": "CPU load",
            "width": "900",
            "height": "200",
            "yaxismin": "0.0000",
            "yaxismax": "100.0000",
            "templateid": "433",
            "show_work_period": "1",
            "show_triggers": "1",
            "graphtype": "0",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "ymin_type": "1",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "614",
            "name": "CPU utilization",
            "width": "900",
            "height": "200",
            "yaxismin": "0.0000",
            "yaxismax": "100.0000",
            "templateid": "387",
            "show_work_period": "1",
            "show_triggers": "0",
            "graphtype": "1",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "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": "0.0000",
            "yaxismax": "0.0000",
            "templateid": "0",
            "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",
            "flags": "4"
        }
    ],
    "id": 1
}
```

See also

- graph.getobjects
- Discovery rule
- Graph item
- Item
- Host
- Host group
- Template

Source

CGraph::get() in *frontends/php/include/classes/api/services/CGraph.php*.

graph.getobjects

Description

`array graph.getobjects(object filter)`

This method allows to retrieve graphs that match the given filter criteria.

Warning:
This method is deprecated and will be removed in the future. Please use graph.get instead.

Parameters

`(object)` Graph properties to search by.

Return values

`(array)` Returns an array of objects with all properties.

Examples

Retrieving graphs from a host

Retrieve all graphs from host "Zabbix server".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graph.getobjects",
    "params": {
        "host": "Zabbix server"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "graphid": "612",
            "name": "CPU jumps",
            "width": "900",
            "height": "200",
            "yaxismin": "0.0000",
```

```
            "yaxismax": "100.0000",
            "templateid": "439",
            "show_work_period": "1",
            "show_triggers": "1",
            "graphtype": "0",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "613",
            "name": "CPU load",
            "width": "900",
            "height": "200",
            "yaxismin": "0.0000",
            "yaxismax": "100.0000",
            "templateid": "433",
            "show_work_period": "1",
            "show_triggers": "1",
            "graphtype": "0",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "ymin_type": "1",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "614",
            "name": "CPU utilization",
            "width": "900",
            "height": "200",
            "yaxismin": "0.0000",
            "yaxismax": "100.0000",
            "templateid": "387",
            "show_work_period": "1",
            "show_triggers": "0",
            "graphtype": "1",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "ymin_type": "1",
            "ymax_type": "1",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        }
    ],
    "id": 1
}
```


See also

- graph.get

Source

CGraph::getObject() in *frontends/php/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 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 *frontends/php/include/classes/api/services/CGraph.php*.

Graph item

This class is designed to work with hosts.

Object references:

- Graph item

Available methods:

- graphitem.get - retrieving graph items

> Graph item object

The following objects are directly related to the `graphitem` API.

Graph item

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: 0.
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
gitemids	string/array	Return only graph items with the given IDs.
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 the graph that the item belongs to as an array in the `graphs` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `gitemid`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
limit	integer	
output	query	
preservekeys	flag	
sortorder	string/array	
~~expandData~~ (deprecated)	flag	Return additional data about the item and the host. Adds the following properties to each graph item: `key_` - *(string)* key of the item; `hostid` - *(string)* ID of the host; `flags` - *(string)* origin of the item; `host` - *(string)* technical name of the host.

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",
        "expandData": 1,
        "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",
            "key_": "system.cpu.util[,steal]",
            "hostid": "10001",
            "flags": "0",
            "host": "Template OS Linux"
        },
        {
            "gitemid": "1243",
            "graphid": "387",
            "itemid": "22668",
            "drawtype": "1",
            "sortorder": "2",
            "color": "55FF55",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0",
            "key_": "system.cpu.util[,softirq]",
            "hostid": "10001",
            "flags": "0",
            "host": "Template OS Linux"
        },
        {
            "gitemid": "1244",
            "graphid": "387",
            "itemid": "22671",
            "drawtype": "1",
            "sortorder": "3",
            "color": "009999",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0",
            "key_": "system.cpu.util[,interrupt]",
            "hostid": "10001",
            "flags": "0",
            "host": "Template OS Linux"
        }
    ],
    "id": 1
}
```

See also

- Graph

Source

CGraphItem::get() in *frontends/php/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.exists - checking if graph prototypes exist
- graphprototype.get - retrieving graph prototypes
- graphprototype.getobjects - retrieving graph objects by filters
- 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.
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’ 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.

Property	Type	Description
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.
yaxismin	float	The fixed minimum value for the Y axis.
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.

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 *frontends/php/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 *frontends/php/include/classes/api/services/CGraphPrototype.php*.

graphprototype.exists

Description

`boolean graphprototype.exists(object filter)`

This method checks if at least one graph prototype that matches the given filter criteria exists.

Warning:
This method is deprecated and will be removed in the future. Please use graphprototype.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
host	string/array	Technical names of the hosts that the graph prototypes belong to.
hostids	string/array	IDs of the hosts that the graph prototypes belong to.
name	string/array	Names of the graph prototypes.

Return values

`(boolean)` Returns `true` if at least one graph prototype that matches the given filter criteria exists.

Examples

Checking a graph prototype on a host

Check if graph prototype ”Disk space usage {#FSNAME}” exists on host ”Zabbix server”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.exists",
    "params": {
        "name": "Disk space usage {#FSNAME}",
        "host": "Zabbix server"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CGraphPrototype::exists() in *frontends/php/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 the LLD rule that the graph prototype belongs to in the `discoveryRule` property.
selectGraphItems	query	Return the graph items used in the graph prototype in the `gitems` property.
selectGroups	query	Return the host groups that the graph prototype belongs to in the `groups` property.
selectHosts	query	Return the hosts that the graph prototype belongs to in the `hosts` property.
selectItems	query	Return the items and item prototypes used in the graph prototype in the `items` property.
selectTemplates	query	Return the templates that the graph prototype belongs to in the `templates` 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 graph prototype belongs to; `hostid` - ID of the host that the graph prototype belongs to.

Parameter	Type	Description
sortfield	string/array	Sort the result by the given properties. Possible values are: `graphid`, `name` and `graphtype`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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"
        }
    ],
    "id": 1
}
```

See also

- graphprototype.getobjects
- Discovery rule
- Graph item
- Item
- Host
- Host group
- Template

Source

CGraphPrototype::get() in *frontends/php/include/classes/api/services/CGraphPrototype.php*.

graphprototype.getobjects

Description

`array graphprototype.getobjects(object filter)`

This method allows to retrieve graph prototypes that match the given filter criteria.

Parameters

`(object)` Graph prototype properties to search by.

Return values

`(array)` Returns an array of objects with all properties.

Examples

Retrieving graph prototypes from a host

Retrieve all graph prototypes from host ”Zabbix server”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.getobjects",
    "params": {
        "host": "Zabbix server"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "graphid": "1017",
            "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"
        }
    ],
    "id": 1
}
```


See also

- graphprototype.get

Source

CGraphPrototype::getObject() in *frontends/php/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 *frontends/php/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.
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.

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.

Parameter	Type	Description
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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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": "0.0850",
            "ns": "563157632"
        },
        {
            "itemid": "23296",
            "clock": "1351090936",
            "value": "0.1600",
            "ns": "549216402"
```

```
        },
        {
            "itemid": "23296",
            "clock": "1351090876",
            "value": "0.1800",
            "ns": "537418114"
        },
        {
            "itemid": "23296",
            "clock": "1351090816",
            "value": "0.2100",
            "ns": "522659528"
        },
        {
            "itemid": "23296",
            "clock": "1351090756",
            "value": "0.2150",
            "ns": "507809457"
        },
        {
            "itemid": "23296",
            "clock": "1351090696",
            "value": "0.2550",
            "ns": "495509699"
        },
        {
            "itemid": "23296",
            "clock": "1351090636",
            "value": "0.3600",
            "ns": "477708209"
        },
        {
            "itemid": "23296",
            "clock": "1351090576",
            "value": "0.3750",
            "ns": "463251343"
        },
        {
            "itemid": "23296",
            "clock": "1351090516",
            "value": "0.3150",
            "ns": "447947017"
        },
        {
            "itemid": "23296",
            "clock": "1351090456",
            "value": "0.2750",
            "ns": "435307141"
        }
    ],
    "id": 1
}
```


Source

CHistory::get() in *frontends/php/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.exists - checking if a host exists
- host.get - retrieving hosts
- host.isreadable - checking if hosts are readable
- host.iswritable - checking if hosts are writable
- 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.
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 - disabled; 0 - *(default)* 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+.

Property	Type	Description
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.
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.

Property	Type	Description
status	integer	Status and function of the host. Possible values are: 0 - *(default)* monitored host; 1 - unmonitored host.

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
4	alias	string	Alias.
11	asset_tag	string	Asset tag.
28	chassis	string	Chassis.
23	contact	string	Contact person.
32	contract_number	string	Contract number.
47	date_hw_decomm	string	HW decommissioning date.
46	date_hw_expiry	string	HW maintenance expiry date.
45	date_hw_install	string	HW installation date.
44	date_hw_purchase	string	HW purchase date.
34	deployment_status	string	Deployment status.
14	hardware	string	Hardware.
15	hardware_full	string	Detailed hardware.
39	host_netmask	string	Host subnet mask.
38	host_networks	string	Host networks.
40	host_router	string	Host router.
30	hw_arch	string	HW architecture.
33	installer_name	string	Installer name.
24	location	string	Location.
25	location_lat	string	Location latitude.
26	location_lon	string	Location longitude.
12	macaddress_a	string	MAC address A.
13	macaddress_b	string	MAC address B.
29	model	string	Model.
3	name	string	Name.
27	notes	string	Notes.
41	oob_ip	string	OOB IP address.
42	oob_netmask	string	OOB host subnet mask.
43	oob_router	string	OOB router.
5	os	string	OS name.
6	os_full	string	Detailed OS name.
7	os_short	string	Short OS name.
61	poc_1_cell	string	Primary POC mobile number.
58	poc_1_email	string	Primary email.
57	poc_1_name	string	Primary POC name.
63	poc_1_notes	string	Primary POC notes.
59	poc_1_phone_a	string	Primary POC phone A.
60	poc_1_phone_b	string	Primary POC phone B.
62	poc_1_screen	string	Primary POC screen name.
68	poc_2_cell	string	Secondary POC mobile number.
65	poc_2_email	string	Secondary POC email.
64	poc_2_name	string	Secondary POC name.
70	poc_2_notes	string	Secondary POC notes.
66	poc_2_phone_a	string	Secondary POC phone A.
67	poc_2_phone_b	string	Secondary POC phone B.
69	poc_2_screen	string	Secondary POC screen name.
8	serialno_a	string	Serial number A.

ID	Property	Type	Description
9	serialno_b	string	Serial number B.
48	site_address_a	string	Site address A.
49	site_address_b	string	Site address B.
50	site_address_c	string	Site address C.
51	site_city	string	Site city.
53	site_country	string	Site country.
56	site_notes	string	Site notes.
55	site_rack	string	Site rack location.
52	site_state	string	Site state.
54	site_zip	string	Site ZIP/postal code.
16	software	string	Software.
18	software_app_a	string	Software application A.
19	software_app_b	string	Software application B.
20	software_app_c	string	Software application C.
21	software_app_d	string	Software application D.
22	software_app_e	string	Software application E.
17	software_full	string	Software details.
10	tag	string	Tag.
1	type	string	Type.
2	type_full	string	Type details.
35	url_a	string	URL A.
36	url_b	string	URL B.
37	url_c	string	URL C.
31	vendor	string	Vendor.

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.
templates	object/array	Templates to be linked to the host. The templates must have the `templateid` property defined.
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, 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"
            }
        ],
        "templates": [
            {
                "templateid": "20045"
            }
        ],
        "inventory_mode": 0,
        "inventory": {
            "macaddress_a": "01234",
            "macaddress_b": "56768"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "107819"
        ]
    },
    "id": 1
}
```

See also

- Host group
- Template
- User macro
- Host interface
- Host inventory

Source

CHost::create() in *frontends/php/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 *frontends/php/include/classes/api/services/CHost.php*.

host.exists

Description

`boolean host.exists(object filter)`

This method checks if at least one host that matches the given filter criteria exists.

> **Warning:**
> This method is deprecated and will be removed in the future. Please use host.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
hostid	string/array	Host IDs.
host	string/array	Technical names of the hosts.
name	string/array	Visible names of the hosts.

Return values

(boolean) Returns true if at least one host that matches the given filter criteria exists.

Examples

Check host by name

Check if a host with the technical name ”Zabbix Server” exists.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.exists",
    "params": {
        "host": "Zabbix Server"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- host.isreadable
- host.iswritable

Source

CHost::exists() in *frontends/php/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.
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.

Parameter	Type	Description
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_applications	flag	Return only hosts that have applications.
with_graphs	flag	Return only hosts that have graphs.
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.
withInventory	flag	Return only hosts that have inventory data.
selectGroups	query	Return the host groups that the host belongs to in the `groups` property.
selectApplications	query	Return the applications from the host in the `applications` property. Supports `count`.
selectDiscoveries	query	Return the low level discoveries from the host in the `discoveries` property. Supports `count`.
selectDiscoveryRule	query	Return the LLD rule that created the host in the `discoveryRule` property.
selectGraphs	query	Return the graphs from the host in the `graphs` property. Supports `count`.
selectHostDiscovery	query	Return the host discovery object in the `hostDiscovery` property. 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 the web scenarios from the host in the `httpTests` property. Supports `count`.

Parameter	Type	Description
selectInterfaces	query	Return the host interfaces in the `interfaces` property. Supports `count`.
selectInventory	query	Return the host inventory from the host in the `inventory` property.
selectItems	query	Return the items from the host in the `items` property. Supports `count`.
selectMacros	query	Return the macros from the host in the `macros` property.
selectParentTemplates	query	Return the templates that the host is linked to in the `parentTemplates` property. Supports `count`.
selectScreens	query	Return the screens from the host in the `screens` property. Supports `count`.
selectTriggers	query	Return the triggers from the host in the `triggers` property. 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.
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 wildcard search. 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 with 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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.

Parameter	Type	Description
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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": {
        "output": "extend",
        "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."
        },
        {
            "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": ""
        }
    ],
    "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
}
```

See also

- host.getobjects
- Host group
- Template

- User macro
- Host interface

Source

CHost::get() in *frontends/php/include/classes/api/services/CHost.php*.

host.getobjects

Description

`array host.getobjects(object filter)`

This method allows to retrieve hosts that match the given filter criteria.

Warning:
This method is deprecated and will be removed in the future. Please use host.get instead.

Parameters

`(object)` Host properties to search by.

Return values

`(array)` Returns an array of objects with all properties.

Examples

Retrieving a host by name

Retrieve the host with the technical name "Zabbix server".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.getobjects",
    "params": {
        "name": "Zabbix server"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "maintenances": [],
            "hostid": "10084",
            "proxy_hostid": "0",
            "host": "Zabbix server",
            "status": "0",
            "disable_until": "0",
            "error": "",
            "available": "1",
            "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."
        }
    ],
    "id": 1
}
```


See also

- host.get

Source

CHost::getObject() in *frontends/php/include/classes/api/services/CHost.php*.

host.isreadable

Description

`boolean host.isreadable(array hostIds)`

This method checks if the given hosts are available for reading.

Parameters

`(array)` IDs of the hosts to check.

Return values

`(boolean)` Returns `true` if the given hosts are available for reading.

Examples

Check multiple hosts

Check if the two hosts are readable.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "host.isreadable",
    "params": [
        "143",
        "943"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```


Response:


```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```


See also

- host.exists
- host.iswritable

Source

CHost::isReadable() in *frontends/php/include/classes/api/services/CHost.php*.

host.iswritable

Description

`boolean host.iswritable(array hostIds)`

This method checks if the given hosts are available for writing.

Parameters

`(array)` IDs of the hosts to check.

Return values

`(boolean)` Returns `true` if the given hosts are available for writing.

Examples

Check multiple hosts

Check if the two hosts are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.iswritable",
    "params": [
        "143",
        "943"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- host.isreadable
- host.exists

Source

CHost::isWritable() in *frontends/php/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"
            }
        ]
    },
    "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 *frontends/php/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 *frontends/php/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.
inventory_mode	integer	Host inventory population mode. Refer to the host inventory object page for a list of supported inventory modes.
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 *frontends/php/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.

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.
interfaces	object/array	Host interfaces to replace the current host interfaces.
inventory	object	Host inventory properties.
macros	object/array	User macros to replace the current user macros.
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 host. The templates must have the `templateid` property defined.

Note:
As opposed to the Zabbix frontend, when `name` is the same as `host`, updating `host` 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"
            }
        ]
    },
    "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": 2
}
```

See also

- host.massadd
- host.massupdate
- host.massremove
- Host group
- Template
- User macro
- Host interface
- Host inventory

Source

CHost::update() in *frontends/php/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.exists - checking if a host group exists
- hostgroup.get - retrieving host groups
- hostgroup.getobjects - retrieving host groups by filters
- hostgroup.isreadable - checking if host groups are readable
- hostgroup.iswritable - checking if host groups are writable
- 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.

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 *frontends/php/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's marked as internal;
- it is used by a host prototype;
- it is used in a global script.

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 *frontends/php/include/classes/api/services/CHostGroup.php*.

hostgroup.exists

Description

`boolean hostgroup.exists(object filter)`

This method checks if at least one host group that matches the given filter criteria exists.

> **Warning:**
> This method is deprecated and will be removed in the future. Please use hostgroup.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
groupid	string/array	Host group IDs.
name	string/array	Names of the host groups.

Return values

`(boolean)` Returns `true` if at least one host group that matches the given filter criteria exists.

Examples

Check host group by name

Check if a host group called ”Zabbix servers” exists.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.exists",
    "params": {
        "name": "Linux servers"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- hostgroup.isreadable
- hostgroup.iswritable

Source

CHostGroup::exists() in *frontends/php/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_hosts_and_templates	flag	Return only host groups that contain hosts *or* templates.
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_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 the LLD rule that created the host group in the `discoveryRule` property.

Parameter	Type	Description
selectGroupDiscovery	query	Return the host group discovery object in the `groupDiscovery` property. 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 goup 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 the hosts that belong to the host group in the `hosts` property. Supports `count`.
selectTemplates	query	Return the templates that belong to the host group in the `templates` property. 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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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

- hostgroup.getobjects
- Host
- Template

Source

CHostGroup::get() in *frontends/php/include/classes/api/services/CHostGroup.php*.

hostgroup.getobjects

Description

`array hostgroup.getobjects(object filter)`

This method allows to retrieve host groups that match the given filter criteria.

Warning:
This method is deprecated and will be removed in the future. Please use hostgroup.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
groupid	string/array	Host group IDs.
name	string/array	Names of the host groups.

Return values

(array) Returns an array of objects with all properties.

Examples

Retrieve a host group by name

Retrieve a host group called "Zabbix servers."

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.getobjects",
    "params": {
        "name": "Linux servers"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "groupid": "2",
            "name": "Linux servers",
            "internal": "0"
        }
    ],
    "id": 16
}
```

See also

- hostgroup.get

Source

CHostGroup::getObject() in *frontends/php/include/classes/api/services/CHostGroup.php*.

hostgroup.isreadable

Description

`boolean hostgroup.isreadable(array hostGroupIds)`

This method checks if the given host groups are available for reading.

Parameters

(array) IDs of the host groups to check.

Return values

(boolean) Returns `true` if the given host groups are available for reading.

Examples

Check multiple host groups

Check if the two host groups are readable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.isreadable",
    "params": [
        "5",
        "7"
    ],
```

```
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- hostgroup.exists
- hostgroup.iswritable

Source

CHostGroup::isReadable() in *frontends/php/include/classes/api/services/CHostGroup.php*.

hostgroup.iswritable

Description

`boolean hostgroup.iswritable(array hostGroupIds)`

This method checks if the given host groups are available for writing.

Parameters

`(array)` IDs of the host groups to check.

Return values

`(boolean)` Returns `true` if the given host groups are available for writing.

Examples

Check multiple host groups

Check if the two host groups are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.iswritable",
    "params": [
        "5",
        "7"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- hostgroup.isreadable
- hostgroup.exists

Source

CHostGroup::isWritable() in *frontends/php/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 *frontends/php/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 *frontends/php/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 *frontends/php/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 *frontends/php/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.exists - checking if a host interface exists
- 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.

Property	Type	Description
port (required)	string	Port number used by the interface. Can contain user macros.
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.
bulk	integer	Whether to use bulk SNMP requests. Possible values are: 0 - don't use bulk requests; 1 - *(default)* use bulk requests.

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",
        "dns": "",
        "ip": "127.0.0.1",
        "main": 0,
        "port": "10050",
        "type": 1,
        "useip": 1
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
```

```
    "result": {
        "interfaceids": [
            "30062"
        ]
    },
    "id": 1
}
```

See also

- hostinterface.massadd
- host.massadd

Source

CHostInterface::create() in *frontends/php/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 *frontends/php/include/classes/api/services/CHostInterface.php*.

hostinterface.exists

Description

`boolean hostinterface.exists(object filter)`

This method checks if at least one host interface that matches the given filter criteria exists.

> **Warning:**
> This method is deprecated and will be removed in the future. Please use hostinterface.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
dns	string/array	DNS of the host interfaces.
hostid	string/array	IDs of the hosts that the host interfaces must belong to.
interfaceid	string/array	Host interface IDs.
ip	string/array	IPs of the host interfaces.

Return values

`(boolean)` Returns `true` if at least one host interface that matches the given filter criteria exists.

Examples

Check interface on host

Check if a host interface with IP 127.0.0.1 exists on host 30037.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.exists",
    "params": {
        "hostid": "30037",
        "ip": "127.0.0.1"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CHostInterface::exists() in *frontends/php/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 the items that use the interface in the `items` property. Supports `count`.
selectHosts	query	Return the host that uses the interface as an array in the `hosts` property.
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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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",
            "bulk": "1"
        },
        {
            "interfaceid": "30067",
            "hostid": "30057",
            "main": "0",
            "type": "1",
            "useip": "0",
            "ip": "",
            "dns": "localhost",
            "port": "10050",
            "bulk": "1"
        },
        {
            "interfaceid": "30068",
            "hostid": "30057",
            "main": "1",
            "type": "2",
            "useip": "1",
            "ip": "127.0.0.1",
            "dns": "",
            "port": "161",
            "bulk": "1"
        }
    ],
    "id": 1
}
```


See also

- Host
- Item

Source

CHostInterface::get() in *frontends/php/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 *frontends/php/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 *frontends/php/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 *frontends/php/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 *frontends/php/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.isreadable - checking if host prototypes are readable
- hostprototype.iswritable - checking if host prototypes are writable
- 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.
templateid	string	*(readonly)* ID of the parent template host prototype.

Host prototype inventory

The host prototype inventory object has the following properties.

Property	Type	Description
inventory_mode	integer	Host prototype inventory population mode. Possible values are: -1 - disabled; 0 - *(default)* manual; 1 - automatic.

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.
inventory	object	Host prototype inventory properties.
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
- Host prototype inventory

Source

CHostPrototype::create() in *frontends/php/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 *frontends/php/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 the LLD rule that the host prototype belongs to in the `discoveryRule` property.
selectGroupLinks	query	Return the group links of the host prototype in the `groupLinks` property.
selectGroupPrototypes	query	Return the group prototypes of the host prototype in the `groupPrototypes` property.
selectInventory	query	Return the host prototype inventory in the `inventory` property.
selectParentHost	query	Return the host that the host prototype belongs to in the `parentHost` property.
selectTemplates	query	Return the templates linked to the host prototype in the `templates` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `hostid`, `host`, `name` and `status`.
countOutput	flag	These parameters being common for all `get` methods are described in detail on the Generic Zabbix API information page.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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",
            "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
- Host prototype inventory

Source

CHostPrototype::get() in *frontends/php/include/classes/api/services/CHostPrototype.php*.

hostprototype.isreadable

Description

`boolean hostprototype.isreadable(array hostPrototypeIds)`

This method checks if the given host prototypes are available for reading.

Parameters

`(array)` IDs of the host prototypes to check.

Return values

(boolean) Returns true if the given host prototypes are available for reading.

Examples

Check multiple host prototypes

Check if the two host prototypes are readable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.isreadable",
    "params": [
        "10092",
        "10093"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- hostprototype.iswritable

Source

CHostPrototype::isReadable() in *frontends/php/include/classes/api/services/CHostPrototype.php*.

hostprototype.iswritable

Description

boolean hostprototype.iswritable(array hostPrototypeIds)

This method checks if the given host prototypes are available for writing.

Parameters

(array) IDs of the host prototypes to check.

Return values

(boolean) Returns true if the given host prototypes are available for writing.

Examples

Check multiple host prototypes

Check if the two host prototypes are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.iswritable",
    "params": [
        "10092",
        "10093"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- hostprototype.isreadable

Source

CHostPrototype::isWritable() in *frontends/php/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.
inventory	object	Host prototype inventory properties.
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
- Host prototype inventory

Source

CHostPrototype::update() in *frontends/php/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.isreadable - check if an icon map is readable
- iconmap.iswritable - check if an icon map is writable
- 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 (reqiured)	string	ID of the default icon.
name (required)	string	Name of the icon map.

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.

Property	Type	Description
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	Position of the icon mapping in the icon map. Default: 0.

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 *frontends/php/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 *frontends/php/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 used icon mappings in the `mappings` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `iconmapid` and `name`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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 *frontends/php/include/classes/api/services/CIconMap.php*.

iconmap.isreadable

Description

`boolean iconmap.isreadable(array iconMapIds)`

This method checks if the given icon maps are available for reading.

Parameters

`(array)` IDs of the icon maps to check.

Return values

`(boolean)` Returns `true` if the given icon maps are available for reading.

Examples

Check multiple icon maps

Check if the two icon maps are readable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.isreadable",
    "params": [
        "4", "6"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- iconmap.iswritable

Source

CIconMap::isReadable() in *frontends/php/include/classes/api/services/CIconMap.php*.

iconmap.iswritable

Description

`boolean iconmap.iswritable(array iconMapIds)`

This method checks if the given icon maps are available for writing.

Parameters

`(array)` IDs of the icon maps to check.

Return values

`(boolean)` Returns `true` if the given icon maps are available for writing.

Examples

Check multiple icon maps

Check if the two icon maps are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.iswritable",
    "params": [
        "4", "6"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- iconmap.isreadable

Source

CIconMap::isWritable() in *frontends/php/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 *frontends/php/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.exists - check if an image exists
- image.get - retrieve images
- image.getobjects - retrieve images by filters

- 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.

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
image (required)	string	Base64 encoded image. The maximum size of the encoded image is 1 MB.

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 *frontends/php/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 *frontends/php/include/classes/api/services/CImage.php*.

image.exists

Description

`boolean image.exists(object filter)`

This method checks if at least one image that matches the given filter criteria exists.

> **Warning:**
> This method is deprecated and will be removed in the future. Please use image.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
imageid	string/array	IDs of images.
imagetype	integer/array	Types of images. Refer to the image "imagetype" property for a list of supported types.
name	string/array	Names of images.

Return values

`(boolean)` Returns `true` if at least one image that matches the given filter criteria exists.

Examples

Check image by name

Check if an image called "Cloud_(96)" exists.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "image.exists",
    "params": {
        "name": "Cloud_(96)"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CImage::exists() in *frontends/php/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 the Base64 encoded image in the `image` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `imageid` and `name`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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
}
```

See also

- image.getobjects

Source

CImage::get() in *frontends/php/include/classes/api/services/CImage.php*.

image.getobjects

Description

`array image.getobjects(object filter)`

This method allows to retrieve images that match the given filter criteria.

> **Warning:**
> This method is deprecated and will be removed in the future. Please use image.get instead.

Parameters

`(object)` Image properties to search by.

Return values

`(array)` Returns an array of objects with all properties.

Examples

Retrieve image by name

Retrieve image called ”Cloud_(24)”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "image.getobjects",
    "params": {
        "name": "Cloud_(24)"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "imageid": "2",
            "imagetype": "1",
            "name": "Cloud_(24)"
        }
    ],
    "id": 1
}
```

See also

- image.get

Source

CImage::getObject() in *frontends/php/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.

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 *frontends/php/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.exists - checking if items exists
- item.get - retrieving items
- item.getobjects - retrieving items by filters

- item.isreadable - checking if items are readable
- item.iswritable - checking if items are writable
- 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)	integer	Update interval of the item in seconds.
hostid (required)	string	ID of the host or template that the item belongs to.
interfaceid (required)	string	ID of the item's host interface. Not required for template items. Optional for Zabbix agent (active), Zabbix internal, Zabbix trapper, Zabbix aggregate, database monitor and calculated 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; 1 - SNMPv1 agent; 2 - Zabbix trapper; 3 - simple check; 4 - SNMPv2 agent; 5 - Zabbix internal; 6 - SNMPv3 agent; 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.
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
authtype	integer	SSH authentication method. Used only by SSH agent items. Possible values: 0 - *(default)* password; 1 - public key.
data_type	integer	Data type of the item. Possible values: 0 - *(default)* decimal; 1 - octal; 2 - hexadecimal; 3 - boolean.
delay_flex	string	Flexible intervals as a serialized string. Each serialized flexible interval consists of an update interval and a time period separated by a forward slash. Multiple intervals are separated by a colon.
delta	integer	Value that will be stored. Possible values: 0 - *(default)* as is; 1 - Delta, speed per second; 2 - Delta, simple change.
description	string	Description of the item.
error	string	*(readonly)* Error text if there are problems updating the item.
flags	integer	*(readonly)* Origin of the item. Possible values: 0 - a plain item; 4 - a discovered item.
formula	integer/float	Custom multiplier. Default: 1.
history	integer	Number of days to keep item's history data. Default: 90.
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.
lastclock	timestamp	*(readonly)* Time when the item 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 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.
mtime	timestamp	Time when the monitored log file was last updated. Used only by log items.

Property	Type	Description
multiplier	integer	Whether to use a custom multiplier.
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 and JMX items.
port	string	Port monitored by the item. Used only by SNMP items.
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.
snmp_community	string	SNMP community. Used only by SNMPv1 and SNMPv2 items.
snmp_oid	string	SNMP OID.
snmpv3_authpassphrase	string	SNMPv3 auth passphrase. Used only by SNMPv3 items.
snmpv3_authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 items. Possible values: 0 - *(default)* MD5; 1 - SHA.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 items.
snmpv3_privpassphrase	string	SNMPv3 priv passphrase. Used only by SNMPv3 items.
snmpv3_privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 items. Possible values: 0 - *(default)* DES; 1 - AES.
snmpv3_securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 items. Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	SNMPv3 security name. Used only by SNMPv3 items.
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.
templateid	string	(readonly) ID of the parent template item. *Hint*: Use the `hostid` property to specify the template that the item belongs to.
trapper_hosts	string	Allowed hosts. Used only by trapper items.
trends	integer	Number of days to keep item's trends data. Default: 365.
units	string	Value units.
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor and JMX items. Required by SSH and Telnet items.

Property	Type	Description
valuemapid	string	ID of the associated value map.

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.

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 $1",
        "key_": "vfs.fs.size[/home/joe/,free]",
        "hostid": "30074",
        "type": 0,
        "value_type": 3,
        "interfaceid": "30084",
        "applications": [
            "609",
            "610"
        ],
        "delay": 30
    },
    "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": 10,
        "inventory_link": 5
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24759"
        ]
    },
    "id": 1
}
```

Source

CItem::create() in *frontends/php/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.

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 *frontends/php/include/classes/api/services/CItem.php*.

item.exists

Description

`boolean item.exists(object filter)`

This method checks if at least one item that matches the given filter criteria exists.

Warning:
This method is deprecated and will be removed in the future. Please use item.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
key_ (required)	string/array	Keys of the items.
host	string/array	Names of the hosts that the items must belong to.
hostid	string/array	IDs of the hosts that the items must belong to.

Return values

`(boolean)` Returns `true` if at least one item that matches the given filter criteria exists.

Examples

Check item by key

Check if an item with key "vm.memory.size[available]" exists on the host "Linux Server."

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.exists",
    "params": {
```

```
        "host": "Linux Server",
        "key_": "vm.memory.size[available]"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- item.isreadable
- item.iswritable

Source

CItem::exists() in *frontends/php/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.

Parameter	Type	Description
selectHosts	query	Returns the host that the item belongs to as an array in the `hosts` property.
selectInterfaces	query	Returns the host interface used by the item as an array in the `interfaces` property.
selectTriggers	query	Return triggers that the item is used in in the `triggers` property. Supports `count`.
selectGraphs	query	Return graphs that contain the item in the `graphs` property. Supports `count`.
selectApplications	query	Return the applications that the item belongs to in the `applications` property.
selectDiscoveryRule	query	Return the LLD rule that created the item in the `discoveryRule` property.
selectItemDiscovery	query	Return the item discovery object in the `itemDiscovery` property. 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.
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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
search	object	

Parameter	Type	Description
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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_community": "",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "Context switches per second",
            "key_": "system.cpu.switches",
            "delay": "60",
            "history": "7",
            "trends": "365",
            "lastvalue": "2552",
            "lastclock": "1351090998",
            "prevvalue": "2641",
            "state": "0",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "sps",
            "multiplier": "0",
            "delta": "1",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
```

```
        "formula": "1",
        "error": "",
        "lastlogsize": "0",
        "logtimefmt": "",
        "templateid": "22680",
        "valuemapid": "0",
        "delay_flex": "",
        "params": "",
        "ipmi_sensor": "",
        "data_type": "0",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "mtime": "0",
        "lastns": "564054253",
        "flags": "0",
        "filter": "",
        "interfaceid": "1",
        "port": "",
        "description": "",
        "inventory_link": "0",
        "lifetime": "0"
    },
    {
        "itemid": "23299",
        "type": "0",
        "snmp_community": "",
        "snmp_oid": "",
        "hostid": "10084",
        "name": "CPU $2 time",
        "key_": "system.cpu.util[,idle]",
        "delay": "60",
        "history": "7",
        "trends": "365",
        "lastvalue": "86.031879",
        "lastclock": "1351090999",
        "prevvalue": "85.306944",
        "state": "0",
        "status": "0",
        "value_type": "0",
        "trapper_hosts": "",
        "units": "%",
        "multiplier": "0",
        "delta": "0",
        "snmpv3_securityname": "",
        "snmpv3_securitylevel": "0",
        "snmpv3_authpassphrase": "",
        "snmpv3_privpassphrase": "",
        "formula": "1",
        "error": "",
        "lastlogsize": "0",
        "logtimefmt": "",
        "templateid": "17354",
        "valuemapid": "0",
        "delay_flex": "",
        "params": "",
        "ipmi_sensor": "",
        "data_type": "0",
        "authtype": "0",
        "username": "",
```

```
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "lastns": "564256864",
            "flags": "0",
            "filter": "",
            "interfaceid": "1",
            "port": "",
            "description": "The time the CPU has spent doing nothing.",
            "inventory_link": "0",
            "lifetime": "0"
        },
        {
            "itemid": "23300",
            "type": "0",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "CPU $2 time",
            "key_": "system.cpu.util[,interrupt]",
            "delay": "60",
            "history": "7",
            "trends": "365",
            "lastvalue": "0.008389",
            "lastclock": "1351091000",
            "prevvalue": "0.000000",
            "state": "0",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "%",
            "multiplier": "0",
            "delta": "0",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "formula": "1",
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "22671",
            "valuemapid": "0",
            "delay_flex": "",
            "params": "",
            "ipmi_sensor": "",
            "data_type": "0",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "lastns": "564661387",
            "flags": "0",
            "filter": "",
            "interfaceid": "1",
            "port": "",
            "description": "The amount of time the CPU has been servicing hardware interrupts.",
            "inventory_link": "0",
            "lifetime": "0"
```

```
        }
    ],
    "id": 1
}
```

See also

- item.getobjects
- Application
- Discovery rule
- Graph
- Host
- Host interface
- Trigger

Source

CItem::get() in *frontends/php/include/classes/api/services/CItem.php*.

item.getobjects

Description

`array item.getobjects(object filter)`

This method allows to retrieve items that match the given filter criteria.

Warning:
This method is deprecated and will be removed in the future. Please use item.get instead.

Parameters

`(object)` Criteria to search by.

Additionally to the standard standard item properties the following parameters are supported as search criteria.

Parameter	Type	Description
host	string/array	Technical name of the host that the item belongs to.

Return values

`(array)` Returns an array of objects with all properties.

Examples

Retrieving items from a host

Retrieve all items from the host ”Zabbix server.”

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.getobjects",
    "params": {
        "host": "Zabbix server"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23327",
```

```
        "type": "0",
        "snmp_community": "",
        "snmp_oid": "",
        "hostid": "10084",
        "name": "Host name of zabbix_agentd running",
        "key_": "agent.hostname",
        "delay": "3600",
        "history": "7",
        "trends": "365",
        "lastvalue": "trapper-host",
        "lastclock": "1351088927",
        "prevvalue": "0",
        "state": "0",
        "status": "0",
        "value_type": "1",
        "trapper_hosts": "",
        "units": "",
        "multiplier": "0",
        "delta": "0",
        "snmpv3_securityname": "",
        "snmpv3_securitylevel": "0",
        "snmpv3_authpassphrase": "",
        "snmpv3_privpassphrase": "",
        "formula": "1",
        "error": "",
        "lastlogsize": "0",
        "logtimefmt": "",
        "templateid": "23319",
        "valuemapid": "0",
        "delay_flex": "",
        "params": "",
        "ipmi_sensor": "",
        "data_type": "0",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "mtime": "0",
        "lastns": "40510111",
        "flags": "0",
        "filter": "",
        "interfaceid": "1",
        "port": "",
        "description": "",
        "inventory_link": "0",
        "lifetime": "30"
    },
    {
        "itemid": "23287",
        "type": "0",
        "snmp_community": "",
        "snmp_oid": "",
        "hostid": "10084",
        "name": "Agent ping",
        "key_": "agent.ping",
        "delay": "60",
        "history": "7",
        "trends": "365",
        "lastvalue": "1",
        "lastclock": "1351090987",
        "prevvalue": "1",
```

```
            "state": "0",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "multiplier": "0",
            "delta": "0",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "formula": "1",
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "10020",
            "valuemapid": "10",
            "delay_flex": "",
            "params": "",
            "ipmi_sensor": "",
            "data_type": "0",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "lastns": "560794191",
            "flags": "0",
            "filter": "",
            "interfaceid": "1",
            "port": "",
            "description": "The agent always returns 1 for this item. It could be used in combination with
            "inventory_link": "0",
            "lifetime": "0"
        },
        {
            "itemid": "23288",
            "type": "0",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "Version of zabbix_agent(d) running",
            "key_": "agent.version",
            "delay": "3600",
            "history": "7",
            "trends": "365",
            "lastvalue": "2.0.0",
            "lastclock": "1351088888",
            "prevvalue": "0",
            "state": "0",
            "status": "0",
            "value_type": "1",
            "trapper_hosts": "",
            "units": "",
            "multiplier": "0",
            "delta": "0",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "formula": "1",
```

```
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "10059",
            "valuemapid": "0",
            "delay_flex": "",
            "params": "",
            "ipmi_sensor": "",
            "data_type": "0",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "lastns": "8826267",
            "flags": "0",
            "filter": "",
            "interfaceid": "1",
            "port": "",
            "description": "",
            "inventory_link": "0",
            "lifetime": "0"
        }
    ],
    "id": 1
}
```


See also

- item.get

Source

CItem::getObject() in *frontends/php/include/classes/api/services/CItem.php*.

item.isreadable

Description

`boolean item.isreadable(array itemIds)`

This method checks if the given items are available for reading.

Parameters

`(array)` IDs of the items to check.

Return values

`(boolean)` Returns `true` if the given items are available for reading.

Examples

Check multiple items

Check if the two items are readable.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "item.isreadable",
    "params": [
        "23298",
        "23323"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- item.exists
- item.iswritable

Source

CItem::isReadable() in *frontends/php/include/classes/api/services/CItem.php*.

item.iswritable

Description

`boolean item.iswritable(array itemIds)`

This method checks if the given items are available for writing.

Parameters

`(array)` IDs of the items to check.

Return values

`(boolean)` Returns `true` if the given items are available for writing.

Examples

Check multiple items

Check if the two items are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.iswritable",
    "params": [
        "23298",
        "23323"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- item.isreadable
- item.exists

Source

CItem::isWritable() in *frontends/php/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.

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
}
```

Source

CItem::update() in *frontends/php/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.exists - checking if item prototypes exist
- itemprototype.get - retrieving item prototypes
- itemprototype.isreadable - checking if item prototypes are readable
- itemprototype.iswritable - checking if item prototypes are writable
- 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)	integer	Update interval of the item prototype in seconds.
hostid (required)	string	ID of the host that the item prototype belongs to.
interfaceid (required)	string	ID of the item prototype's host interface. Used only for host item prototypes. Optional for Zabbix agent (active), Zabbix internal, Zabbix trapper, Zabbix aggregate, database monitor and calculated 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; 1 - SNMPv1 agent; 2 - Zabbix trapper; 3 - simple check; 4 - SNMPv2 agent; 5 - Zabbix internal; 6 - SNMPv3 agent; 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.

Property	Type	Description
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.
authtype	integer	SSH authentication method. Used only by SSH agent item prototypes. Possible values: 0 - *(default)* password; 1 - public key.
data_type	integer	Data type of the item prototype. Possible values: 0 - *(default)* decimal; 1 - octal; 2 - hexadecimal; 3 - boolean.
delay_flex	string	Flexible intervals as a serialized string. Each serialized flexible interval consists of an update interval and a time period separated by a forward slash. Multiple intervals are separated by a colon.
delta	integer	Value that will be stored. Possible values: 0 - *(default)* as is; 1 - Delta, speed per second; 2 - Delta, simple change.
description	string	Description of the item prototype.
formula	integer/float	Custom multiplier. Default: 1.
history	integer	Number of days to keep item prototype's history data. Default: 90.
ipmi_sensor	string	IPMI sensor. Used only by IPMI item prototypes.
logtimefmt	string	Format of the time in log entries. Used only by log item prototypes.
multiplier	integer	Whether to use a custom multiplier.
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 and JMX item prototypes.
port	string	Port monitored by the item prototype. Used only by SNMP items prototype.
privatekey	string	Name of the private key file.
publickey	string	Name of the public key file.
snmp_community	string	SNMP community. Used only by SNMPv1 and SNMPv2 item prototypes.
snmp_oid	string	SNMP OID.
snmpv3_authpassphrase	string	SNMPv3 auth passphrase. Used only by SNMPv3 item prototypes.

Property	Type	Description
snmpv3_authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 items. Possible values: 0 - *(default)* MD5; 1 - SHA.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 item prototypes.
snmpv3_privpassphrase	string	SNMPv3 priv passphrase. Used only by SNMPv3 item prototypes.
snmpv3_privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 items. Possible values: 0 - *(default)* DES; 1 - AES.
snmpv3_securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 item prototypes. Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	SNMPv3 security name. Used only by SNMPv3 item prototypes.
status	integer	Status of the item prototype. Possible values: 0 - *(default)* enabled item prototype; 1 - disabled item prototype; 3 - unsupported item prototype.
templateid	string	(readonly) ID of the parent template item prototype.
trapper_hosts	string	Allowed hosts. Used only by trapper item prototypes.
trends	integer	Number of days to keep item prototype's trends data. Default: 365.
units	string	Value units.
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor and JMX item prototypes. Required by SSH and Telnet item prototypes.
valuemapid	string	ID of the associated value map.

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.

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 disc space on a discovered file system. Discovered items should be numeric Zabbix agent items updated every 30 seconds.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "name": "Free disk space on $1",
        "key_": "vfs.fs.size[{#FSNAME},free]",
        "hostid": "10197",
        "ruleid": "27665",
        "type": 0,
        "value_type": 3,
        "interfaceid": "112",
        "delay": 30
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27666"
        ]
    },
    "id": 1
}
```

Source

CItemPrototype::create() in *frontends/php/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.

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 *frontends/php/include/classes/api/services/CItemPrototype.php*.

itemprototype.exists

Description

`boolean itemprototype.exists(object filter)`

This method checks if at least one item prototype that matches the given filter criteria exists.

Warning:
This method is deprecated and will be removed in the future. Please use itemprototype.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
key_ (required)	string/array	Keys of the item prototypes.
host	string/array	Names of the hosts that the item prototypes must belong to.
hostid	string/array	IDs of the hosts that the item prototypes must belong to.

Return values

`(boolean)` Returns `true` if at least one item prototype that matches the given filter criteria exists.

Examples

Checking if an item prototype exists on a host

Check if item prototype with key "net.if.in[{#IFNAME}]" exists on host "Zabbix server".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.exists",
    "params": {
        "host": "Zabbix server",
        "key_": "net.if.in[{#IFNAME}]"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- itemprototype.isreadable
- itemprototype.iswritable

Source

CItemPrototype::exists() in *frontends/php/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 applications that the item prototype belongs to in the `applications` property.
selectDiscoveryRule	query	Return the low-level discovery rule that the graph prototype belongs to in the `discoveryRule` property.

Parameter	Type	Description
selectGraphs	query	Return graph prototypes that the item prototype is used in in the `graphs` property. Supports `count`.
selectHosts	query	Returns the host that the item prototype belongs to as an array in the `hosts` property.
selectTriggers	query	Return trigger prototypes that the item prototype is used in in the `triggers` property. 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. 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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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": "27427",
            "type": "0",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10202",
            "name": "Incoming network traffic on $1 23",
            "key_": "2net.if.in[{#IFNAME}]",
            "delay": "60",
            "history": "7",
            "trends": "365",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "bps",
            "multiplier": "1",
            "delta": "1",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "formula": "8",
            "logtimefmt": "",
            "templateid": "23881",
            "valuemapid": "0",
            "delay_flex": "",
            "params": "",
            "ipmi_sensor": "",
            "data_type": "0",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "filter": "",
            "interfaceid": "119",
            "port": "",
            "description": "",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0"
        },
        {
            "itemid": "27428",
            "type": "0",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10202",
            "name": "Incoming network traffic on $1",
            "key_": "net.if.in[{#IFNAME}]",
            "delay": "60",
            "history": "7",
            "trends": "365",
```

```
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "bps",
            "multiplier": "1",
            "delta": "1",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "formula": "8",
            "logtimefmt": "",
            "templateid": "22446",
            "valuemapid": "0",
            "delay_flex": "",
            "params": "",
            "ipmi_sensor": "",
            "data_type": "0",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0",
            "filter": "",
            "interfaceid": "119",
            "port": "",
            "description": "",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0"
        }
    ],
    "id": 1
}
```


See also

- Application
- Host
- Graph prototype
- Trigger prototype

Source

CItemPrototype::get() in *frontends/php/include/classes/api/services/CItemPrototype.php*.

itemprototype.isreadable

Description

`boolean itemprototype.isreadable(array itemPrototypeIds)`

This method checks if the given item prototypes are available for reading.

Parameters

`(array)` IDs of the item prototypes to check.

Return values

`(boolean)` Returns `true` if the given item prototypes are available for reading.

Examples

Check multiple item prototypes

Check if the two item prototypes are readable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.isreadable",
    "params": [
        "27352",
        "27356"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- itemprototype.exists
- itemprototype.iswritable

Source

CItemPrototype::isReadable() in *frontends/php/include/classes/api/services/CItemPrototype.php*.

itemprototype.iswritable

Description

`boolean itemprototype.iswritable(array itemPrototypeIds)`

This method checks if the given item prototypes are available for writing.

Parameters

`(array)` IDs of the item prototypes to check.

Return values

`(boolean)` Returns `true` if the given item prototypes are available for writing.

Examples

Check multiple item prototypes

Check if the two item prototypes are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.iswritable",
    "params": [
        "27352",
        "27356"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- itemprototype.isreadable
- itemprototype.exists

Source

CItemPrototype::isWritable() in *frontends/php/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.

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
}
```

Source

CItemPrototype::update() in *frontends/php/include/classes/api/services/CItemPrototype.php*.

IT service

This class is designed to work with IT services.

Object references:

- IT 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.isreadable - checking if IT services are readable
- service.iswritable - checking if IT services are writable
- service.update - updating IT services

> IT Service object

The following objects are directly related to the `service` API.

IT Service

The IT service object has the following properties.

Property	Type	Description
serviceid	string	*(readonly)* ID of the IT service.
algorithm (required)	integer	Algorithm used to calculate the state of the IT 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 IT service.
showsla (required)	integer	Whether SLA should be calculated. Possible values: 0 - do not calculate; 1 - calculate.
sortorder (required)	integer	Position of the IT service used for sorting.
goodsla	float	Minimum acceptable SLA value. If the SLA drops lower, the IT service is considered to be in problem state. Default: 99.9.

Property	Type	Description
status	integer	*(readonly)* Whether the IT service is in OK or problem state. If the IT 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 IT service in problem state. If the IT service is in OK state, `status` is equal to 0.
triggerid	string	Trigger associated with the IT service. Can only be set for IT services that don't have children. Default: 0

Service time

The service time object defines periods, when an IT 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 IT 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 Tue, 2:00 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 Tue, 2:00 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 IT services. It has the following properties.

Property	Type	Description
linkid	string	*(readonly)* ID of the service dependency.
servicedownid (required)	string	ID of the IT service, that a service depends on, that is, the child service. An IT service can have multiple children.
serviceupid (required)	string	ID of the IT service, that is dependent on a service, that is, the parent service. An IT service can have multiple parents forming a directed graph.

Property	Type	Description
soft (required)	integer	Type of dependency between IT services. Possible values: 0 - hard dependency; 1 - soft dependency. An IT 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 IT service tree. Additional parents can be added as soft dependencies forming a graph. An IT 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 IT service's state change. It has the following properties.

Property	Type	Description
servicealarmid	string	ID of the service alarm.
serviceid	string	ID of the IT service.
clock	timestamp	Time when the IT service state change has happened.
value	integer	Status of the IT service. Refer the the IT 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 IT services.

Parameters

`(object/array)` Service dependencies to create.

Each service dependency has the following parameters.

Parameter	Type	Description
serviceid	string	ID of the IT service that depends on a service, that is, the parent service.
dependsOnServiceid	string	ID of the IT 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 IT services under the `serviceids` property.

Examples

Creating a hard dependency

Make IT 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 *frontends/php/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 IT services under the `serviceids` property.

Examples

Adding a scheduled downtime

Add a downtime for IT service ”2” 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 *frontends/php/include/classes/api/services/CService.php*.

service.create

Description

`object service.create(object/array itServices)`

This method allows to create new IT services.

Parameters

`(object/array)` IT services to create.

Additionally to the standard IT 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 IT service the service depends on, that is, the child IT 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 IT service.
times	array	Service times to be created for the IT service.

Return values

`(object)` Returns an object containing the IDs of the created IT services under the `serviceids` property. The order of the returned IDs matches the order of the passed IT services.

Examples

Creating an IT service

Create an IT 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 *frontends/php/include/classes/api/services/CService.php*.

service.delete

Description

`object service.delete(array itServiceIds)`

This method allows to delete IT services.

IT services with hard-dependent child services cannot be deleted.

Parameters

`(array)` IDs of the IT services to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted IT services under the `serviceids` property.

Examples

Deleting multiple IT services

Delete two IT 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 *frontends/php/include/classes/api/services/CService.php*.

service.deletedependencies

Description

`object service.deletedependencies(string/array serviceIds)`

This method allows to delete all dependencies from IT services.

Parameters

(`string/array`) IDs of the IT services to delete all dependencies from.

Return values

(`object`) Returns an object containing the IDs of the affected IT services under the `serviceids` property.

Examples

Deleting dependencies from an IT service

Delete all dependencies from IT 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 *frontends/php/include/classes/api/services/CService.php*.

service.deletetimes

Description

`object service.deletetimes(string/array serviceIds)`

This method allows to delete all service times from IT services.

Parameters

`(string/array)` IDs of the IT services to delete all service times from.

Return values

`(object)` Returns an object containing the IDs of the affected IT services under the `serviceids` property.

Examples

Deleting service times from an IT service

Delete all service times from IT 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 *frontends/php/include/classes/api/services/CService.php*.

service.get

Description

`integer/array service.get(object parameters)`

The method allows to retrieve IT 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 IT services with the given IDs.
parentids	string/array	Return only IT services with the given hard-dependent parent IT services.

Parameter	Type	Description
childids	string/array	Return only IT services that are hard-dependent on the given child IT services.
selectParent	query	Return the hard-dependent parent IT service in the `parent` property.
selectDependencies	query	Return child service dependencies in the `dependencies` property.
selectParentDependencies	query	Return parent service dependencies in the `parentDependencies` property.
selectTimes	query	Return service times in the `times` property.
selectAlarms	query	Return service alarms in the `alarms` property.
selectTrigger	query	Return the associated trigger in the `trigger` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `name` and `sortorder`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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 IT services

Retrieve all data about all IT 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 *frontends/php/include/classes/api/services/CService.php*.

service.getsla

Description

`object service.getsla(object parameters)`

This method allows to calculate availability information about IT services.

Parameters

`(object)` Parameters containing the IDs of the IT services and time intervals to calculate SLA.

Parameter	Type	Description
serviceids	string/array	IDs of IT 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 IT service under the corresponding service ID.

Property	Type	Description
status	integer	Current status of the IT service. Refer to the IT service object page for more information on service statuses.
problems	array	Triggers that are currently in problem state and are linked either to the IT 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 IT 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 *frontends/php/include/classes/api/services/CService.php*.

service.isreadable

Description

`boolean service.isreadable(array serviceIds)`

This method checks if the given IT services are available for reading.

Parameters

`(array)` IDs of the IT services to check.

Return values

`(boolean)` Returns `true` if the given IT services are available for reading.

Examples

Check multiple IT services

Check if the two IT services are readable.

Request:


```
{
    "jsonrpc": "2.0",
```

```
    "method": "service.isreadable",
    "params": [
        "3", "4"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- service.iswritable

Source

CService::isReadable() in *frontends/php/include/classes/api/services/CService.php*.

service.iswritable

Description

`boolean service.iswritable(array serviceIds)`

This method checks if the given IT services are available for writing.

Parameters

`(array)` IDs of the IT services to check.

Return values

`(boolean)` Returns `true` if the given IT services are available for writing.

Examples

Check multiple IT services

Check if the two IT services are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.iswritable",
    "params": [
        "3", "4"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- service.isreadable

Source

CService::isWritable() in *frontends/php/include/classes/api/services/CService.php*.

service.update

Description

`object service.update(object/array itServices)`

This method allows to update existing IT services.

Parameters

`(object/array)` IT service properties to be updated.

The `serviceid` property must be defined for each IT service, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard IT 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 IT service the service depends on, that is, the child IT 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 IT service.
times	array	Service times to replace the current service times.

Return values

`(object)` Returns an object containing the IDs of the updated IT services under the `serviceids` property.

Examples

Setting the parent of an IT service

Make IT 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 *frontends/php/include/classes/api/services/CService.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.exists - checking if LLD rules exist
- discoveryrule.get - retrieving LLD rules
- discoveryrule.isreadable - checking if LLD rules are readable
- discoveryrule.iswritable - checking if LLD rules are writable
- 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)	integer	Update interval of the LLD rule in seconds.
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. Optional for Zabbix agent (active), Zabbix internal, Zabbix trapper and database monitor LLD rules.
key_ (required)	string	LLD rule key.
name (required)	string	Name of the LLD rule.

Property	Type	Description
type (required)	integer	Type of the LLD rule. Possible values: 0 - Zabbix agent; 1 - SNMPv1 agent; 2 - Zabbix trapper; 3 - simple check; 4 - SNMPv2 agent; 5 - Zabbix internal; 6 - SNMPv3 agent; 7 - Zabbix agent (active); 10 - external check; 11 - database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent; 16 - JMX agent.
authtype	integer	SSH authentication method. Used only by SSH agent LLD rules. Possible values: 0 - *(default)* password; 1 - public key.
delay_flex	string	Flexible intervals as a serialized string. Each serialized flexible interval consists of an update interval and a time period separated by a forward slash. Multiple intervals are separated by a colon.
description	string	Description of the LLD rule.
error	string	*(readonly)* Error text if there are problems updating the LLD rule.
ipmi_sensor	string	IPMI sensor. Used only by IPMI LLD rules.
lifetime	integer	Time period after which items that are no longer discovered will be deleted, in days. Default: 30.
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 and JMX LLD rules.
port	string	Port used by the LLD rule. Used only by SNMP LLD rules.
privatekey	string	Name of the private key file.
publickey	string	Name of the public key file.
snmp_community	string	SNMP community. Required for SNMPv1 and SNMPv2 LLD rules.
snmp_oid	string	SNMP OID.
snmpv3_authpassphrase	string	SNMPv3 auth passphrase. Used only by SNMPv3 LLD rules.
snmpv3_authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 LLD rules. Possible values: 0 - *(default)* MD5; 1 - SHA.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 checks.
snmpv3_privpassphrase	string	SNMPv3 priv passphrase. Used only by SNMPv3 LLD rules.

Property	Type	Description
snmpv3_privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 LLD rules. Possible values: 0 - *(default)* DES; 1 - AES.
snmpv3_securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 LLD rules. Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	SNMPv3 security name. Used only by SNMPv3 LLD rules.
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.
templateid	string	(readonly) ID of the parent template LLD rule.
trapper_hosts	string	Allowed hosts. Used only by trapper LLD rules.
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor and JMX LLD rules. Required by SSH and Telnet LLD rules.

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.
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.

Note:
To better understand how to use filters with various types of expressions, see examples on the discoveryrule.get and discoveryrule.create method pages.

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 *frontends/php/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.

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": 30
    },
    "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": 30,
        "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
}
```

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": 30,
        "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
}
```

See also

- LLD rule filter

Source

CDiscoveryRule::create() in *frontends/php/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 `itemids` property.

Examples

Deleting multiple LLD rules

Delete two LLD rules.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.delete",
    "params": [
        "27665",
        "27668"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "ruleids": [
            "27665",
            "27668"
        ]
    },
    "id": 1
}
```

Source

CDiscoveryRule::delete() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.exists

Description

`boolean discoveryrule.exists(object filter)`

This method checks if at least one LLD rule that matches the given filter criteria exists.

Warning:
This method is deprecated and will be removed in the future. Please use discoveryrule.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
key_ (required)	string/array	Keys of the LLD rules.
host	string/array	Names of the hosts that the LLD rules must belong to.
hostid	string/array	IDs of the hosts that the LLD rules must belong to.

Return values

`(boolean)` Returns `true` if at least one LLD rule that matches the given filter criteria exists.

Examples

Checking if an LLD rule exists on a host

Check if the LLD rule with the key ”vfs.fs.discovery” exists on host ”Zabbix server”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.exists",
    "params": {
        "host": "Zabbix server",
        "key_": "vfs.fs.discovery"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- discoveryrule.isreadable
- discoveryrule.iswritable

Source

CDiscoveryRule::exists() in *frontends/php/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.
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	Returns the filter used by the LLD rule in the `filter` property.
selectGraphs	query	Returns graph prototypes that belong to the LLD rule in the `graphs` property. Supports `count`.
selectHostPrototypes	query	Returns host prototypes that belong to the LLD rule in the `hostPrototypes` property. Supports `count`.
selectHosts	query	Returns the host that the LLD rule belongs to as an array in the `hosts` property.

Parameter	Type	Description
selectItems	query	Returns item prototypes that belong to the LLD rule in the `items` property. Supports `count`.
selectTriggers	query	Returns trigger prototypes that belong to the LLD rule in the `triggers` property. 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. Supports additional filters: `host` - technical name of the host that the LLD rule belongs to.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selctItems`; `selectGraphs`; `selectTriggers`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `itemid`, `name`, `key_`, `delay`, `type` and `status`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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_community": "",
            "snmp_oid": "",
            "hostid": "10202",
            "name": "Network interface discovery",
            "key_": "net.if.discovery",
            "delay": "3600",
            "state": "0",
            "status": "0",
            "trapper_hosts": "",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "error": "",
            "templateid": "22444",
            "delay_flex": "",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "interfaceid": "119",
            "port": "",
            "description": "Discovery of network interfaces as defined in global regular expression \"Netw
            "lifetime": "30",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0"
        },
        {
            "itemid": "27426",
            "type": "0",
            "snmp_community": "",
            "snmp_oid": "",
            "hostid": "10202",
            "name": "Mounted filesystem discovery",
            "key_": "vfs.fs.discovery",
            "delay": "3600",
            "state": "0",
            "status": "0",
            "trapper_hosts": "",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "error": "",
            "templateid": "22450",
            "delay_flex": "",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
```

```
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "interfaceid": "119",
            "port": "",
            "description": "Discovery of file systems of different types as defined in global regular expr
            "lifetime": "30",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0"
        }
    ],
    "id": 2
}
```

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
}
```

See also

- Graph prototype
- Host
- Item prototype
- LLD rule filter
- Trigger prototype

Source

CDiscoveryRule::get() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.isreadable

Description

`boolean discoveryrule.isreadable(array lldRuleIds)`

This method checks if the given LLD rules are available for reading.

Parameters

`(array)` IDs of the LLD rules to check.

Return values

`(boolean)` Returns `true` if the given LLD rules are available for reading.

Examples

Check multiple LLD rules

Check if the two LLD rules are readable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.isreadable",
    "params": [
        "27425",
        "27429"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- discoveryrule.exists
- discoveryrule.iswritable

Source

CDiscoveryRule::isReadable() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.iswritable

Description

`boolean discoveryrule.iswritable(array lldRuleIds)`

This method checks if the given LLD rules are available for writing.

Parameters

`(array)` IDs of the LLD rules to check.

Return values

`(boolean)` Returns `true` if the given LLD rules are available for writing.

Examples

Check multiple LLD rules

Check if the two LLD rules are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.iswritable",
    "params": [
        "27425",
        "27429"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- discoveryrule.isreadable
- discoveryrule.exists

Source

CDiscoveryRule::isWritable() in *frontends/php/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.

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": "24682",
        "filter": {
            "evaltype": 1,
            "conditions": [
                {
                    "macro": "{#FSTYPE}",
                    "value": "@File systems for discovery"
                }
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22450"
        ]
    },
    "id": 1
}
```

Source

CDiscoveryRule::update() in *frontends/php/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.exists - checking if a maintenance exists
- 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	timestamp	Time when the maintenance becomes active. Default: current time.
active_till	timestamp	Time when the maintenance stops being active. Default: the next day.
description	string	Description of the maintenance.
maintenance_type	integer	Type of maintenance. Possible values: 0 - *(default)* with data collection; 1 - without data collection.

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. Days are stored in binary form with each bit representing the corresponding day. For example, 4 equals 100 in binary and means, that maintenance will be enabled on Wednesday. Used for weekly and monthly time periods. Required only for weekly time periods.
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. Months are stored in binary form with each bit representing the corresponding month. For example, 5 equals 101 in binary and means, that maintenance will be enabled in January and March. Required only for monthly time periods.

Property	Type	Description
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.

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.

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". 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,
        "groupids": [
            "2"
        ],
        "timeperiods": [
            {
                "timeperiod_type": 3,
                "every": 1,
                "dayofweek": 64,
                "start_time": 64800,
                "period": 3600
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Time period

Source

CMaintenance::create() in *frontends/php/include/classes/api/services/CMaintenance.php*.

maintenance.delete

Description

`object maintenance.delete(array maintenanceIds)`

This method allows to delete maintenances.

Parameters

`(array)` IDs of the maintenances to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted maintenances under the `maintenanceids` property.

Examples

Deleting multiple maintenances

Delete two maintenanaces.

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 *frontends/php/include/classes/api/services/CMaintenance.php*.

maintenance.exists

Description

`boolean maintenance.exists(object filter)`

This method checks if at least one maintenance that matches the given filter criteria exists.

Warning:
This method is deprecated and will be removed in the future. Please use maintenace.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
maintenanceid	string/array	IDs of the maintenances
name	string/array	Names of the maintenances

Return values

`(boolean)` Returns `true` if at least one maintenance that matches the given filter criteria exists.

Examples

Checking maintenance by name

Check if maintenance with the name ”Sunday maintenance” already exists.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.exists",
    "params": {
        "name": "Sunday maintenance"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CMaintenance::exists() in *frontends/php/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 host groups assigned to the maintenance in the `groups` property.
selectHosts	query	Return hosts assigned to the maintenance in the `hosts` property.
selectTimeperiods	query	Return the maintenance's time periods in the `timeperiods` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `maintenanceid`, `name` and `maintenance_type`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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, hosts and defined time periods.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.get",
    "params": {
        "output": "extend",
        "selectGroups": "extend",
        "selectTimeperiods": "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",
            "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"
                }
            ]
        }
    ],
    "id": 1
}
```

See also

- Host
- Host group
- Time period

Source

CMaintenance::get() in *frontends/php/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.

> **Attention:**
> At this time, partial maintenance update is not supported, all parameters are mandatory. See ZBX-6167 for current status.

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.

> **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 *frontends/php/include/classes/api/services/CMaintenance.php*.

Map

This class is designed to work with maps.

Object references:

- Map
- Map element
- Map link
- Map URL

Available methods:

- map.create - create new maps
- map.delete - delete maps
- map.exists - check if a map exists
- map.get - retrieve maps
- map.getobjects - retrieve maps by filters
- map.isreadable - check if maps are readable
- map.iswritable - check if maps are writable
- 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.
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 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.

Property	Type	Description
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.
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.

Property	Type	Description
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.
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.

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.
elementid (required)	string	ID of the object that the map element represents. Required for host, host group, trigger and map type elements.

Property	Type	Description
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.
label_location	integer	Location of the map element label. Possible values: -1 - *(default)* default location; 0 - bottom; 1 - left; 2 - right; 3 - top.
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.

Property	Type	Description
y	integer	Y-coordinates of the element in pixels. Default: 0.

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.
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 (reqiuired)	string	ID of the trigger used as a link indicator.
color	string	Indicator color as a hexadecimal color code. Default: DD0000.

Property	Type	Description
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.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.

Note:
To create map links you'll need to set a map elements `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": [
            {
                "elementid": "1033",
                "selementid": "1",
                "elementtype": 0,
                "iconid_off": "2"
            },
            {
                "elementid": "1037",
                "selementid": "2",
                "elementtype": 0,
                "iconid_off": "2"
            }
        ],
        "links": [
            {
                "selementid1": "1",
                "selementid2": "2"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ]
    },
    "id": 1
}
```

See also

- Map element
- Map link
- Map URL

Source

CMap::create() in *frontends/php/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 *frontends/php/include/classes/api/services/CMap.php*.

map.exists

Description

`boolean map.exists(object filter)`

This method checks if at least one map that matches the given filter criteria exists.

> **Warning:**
> This method is deprecated and will be removed in the future. Please use map.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
name	string/array	Names of the maps.
sysmapid	string/array	IDs of the maps.

Return values

`(boolean)` Returns `true` if at least one map that matches the given filter criteria exists.

Examples

Check a map by name

Check if map ”Local network” exists.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.exists",
    "params": {
        "name": "Local network"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- map.isreadable
- map.iswritable

Source

CMap::exists() in *frontends/php/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	Return only maps with the given IDs.
expandUrls	flag	Adds global map URLs to the corresponding map elements and expands macros in all map element URLs.
selectIconMap	query	Returns the icon map used on the map in the `iconmap` property.
selectLinks	query	Returns map links between elements in the `links` property.
selectSelements	query	Returns the map elements from the map in the `selements` property.
selectUrls	query	Returns the map URLs in the `urls` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `name`, `width` and `height`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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",
        "sysmapids": "3"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
```

```
    {
        "selements": [
            {
                "selementid": "10",
                "sysmapid": "3",
                "elementid": "0",
                "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": []
            },
            {
                "selementid": "11",
                "sysmapid": "3",
                "elementid": "0",
                "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": []
            }
        ],
        "links": [
            {
                "linkid": "23",
                "sysmapid": "3",
                "selementid1": "10",
                "selementid2": "11",
                "drawtype": "0",
                "color": "00CC00",
                "label": "",
                "linktriggers": []
            }
        ],
        "sysmapid": "3",
        "name": "Local nerwork",
        "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"
        }
    ],
    "id": 1
}
```


See also

- map.getobjects
- Icon map
- Map element
- Map link
- Map URL

Source

CMap::get() in *frontends/php/include/classes/api/services/CMap.php*.

map.getobjects

Description

`array map.getobjects(object filter)`

This method allows to retrieve maps that match the given filter criteria.

> **Warning:**
> This method is deprecated and will be removed in the future. Please use map.get instead.

Parameters

`(object)` Map properties to search by.

Return values

`(array)` Returns an array of objects with all properties.

Examples

Retrieve a map by name

Retrieve a map called ”Local network”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.getobjects",
    "params": {
        "name": "Local nerwork"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "urls": [],
            "sysmapid": "3",
            "name": "Local nerwork",
            "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"
        }
    ],
    "id": 1
}
```

See also

- map.get

Source

CMap::getObject() in *frontends/php/include/classes/api/services/CMap.php*.

map.isreadable

Description

`boolean map.isreadable(array sysmapIds)`

This method checks if the given maps are available for reading.

Parameters

`(array)` IDs of the maps to check.

Return values

`(boolean)` Returns `true` if the given maps are available for reading.

Examples

Check multiple maps

Check if the two maps are readable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.isreadable",
    "params": [
        "32", "6"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- map.exists
- map.iswritable

Source

CMap::isReadable() in *frontends/php/include/classes/api/services/CMap.php*.

map.iswritable

Description

`boolean map.iswritable(array sysmapIds)`

This method checks if the given maps are available for writing.

Parameters

`(array)` IDs of the maps to check.

Return values

`(boolean)` Returns `true` if the given maps are available for writing.

Examples

Check multiple maps

Check if the two maps are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.iswritable",
    "params": [
        "32", "7"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- map.isreadable
- map.exists

Source

CMap::isWritable() in *frontends/php/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 liks to replace the existing links.
selements	array	Map elements to replace the existing elements.
urls	array	Map URLs to replace the existing URLs.

Note:
To create map links between new map elements you'll need to set an elements `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
}
```

See also

- Map element
- Map link
- Map URL

Source

CMap::update() in *frontends/php/include/classes/api/services/CMap.php*.

Media

This class is designed to work with media.

Object references:

- Media

Available methods:

- usermedia.get - retrieving media

Methods to configure media via the `user` API:

- user.addmedia - creating media
- user.updatemedia - updating media
- user.deletemedia - deleting media

> Media object

The following objects are directly related to the `usermedia` API.

Media

> **Note:**
> Media are created, updated and deleted via the the user API.

The media object defines how a media type should be used for a user. It has the following properties.

Property	Type	Description
mediaid	string	*(readonly)* ID of the media.
active (required)	integer	Whether the media is enabled. Possible values: 0 - enabled; 1 - disabled.
mediatypeid (required)	string	ID of the media type used by the media.

Property	Type	Description
period (required)	string	Time when the notifications can be sent as a time period.
sendto (required)	string	Address, user name or other identifier of the recipient.
severity (required)	integer	Trigger severities to send notifications about. Severities are stored in binary form with each bit representing the corresponding severity. For example, 12 equals 1100 in binary and means, that notifications will be sent from triggers with severities warning and average. Refer to the trigger object page for a list of supported trigger severities.
userid (required)	string	ID of the user that uses the media.

usermedia.get

Description

`integer/array usermedia.get(object parameters)`

The method allows to retrieve media according to the given parameters.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
mediaids	string/array	Return only media with the given IDs.
usrgrpids	string/array	Return only media that are used by users in the given user groups.
userids	string/array	Return only media that are used by the given users.
mediatypeids	string/array	Return only media that use the given media types.
sortfield	string/array	Sort the result by the given properties. Possible values are: `mediaid`, `userid` and `mediatypeid`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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 by user

Retrieve all media for the given user.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermedia.get",
    "params": {
        "output": "extend",
        "userids": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mediaid": "8",
            "userid": "1",
            "mediatypeid": "3",
            "sendto": "+3711231233",
            "active": "0",
            "severity": "48",
            "period": "1-5,09:00-18:00"
        },
        {
            "mediaid": "9",
            "userid": "1",
            "mediatypeid": "1",
            "sendto": "john@company.com",
            "active": "0",
            "severity": "63",
            "period": "1-7,00:00-24:00"
        }
    ],
    "id": 1
}
```

Source

CUserMedia::get() in *frontends/php/include/classes/api/services/CUserMedia.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.
description (required)	string	Name of the media type.
type (required)	integer	Transport used by the media type. Possible values: 0 - email; 1 - script; 2 - SMS; 3 - Jabber; 100 - Ez Texting.
exec_path	string	For script media types `exec_path` contains the name of the executed script. For Ez Texting `exec_path` contains the message text limit. Possible text limit values: 0 - USA (160 characters); 1 - Canada (136 characters). Required for script and Ez Texting media types.
gsm_modem	string	Serial device name of the GSM modem. Required for SMS media types.
passwd	string	Authentication password. Required for Jabber and Ez Texting 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.
status	integer	Whether the media type is enabled. Possible values: 0 - *(default)* enabled; 1 - disabled.
username	string	Username or Jabber identifier. Required for Jabber and Ez Texting media types.

mediatype.create

Description

`object mediatype.create(object/array mediaTypes)`

This method allows to create new media types.

Parameters

(object/array) Media types to create.

The method accepts media types with the standard media type properties.

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 a media type

Create a new e-mail media type.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.create",
    "params": {
        "description": "E-mail",
        "type": 0,
        "smtp_server": "rootmail@company.com",
        "smtp_helo": "company.com",
        "smtp_email": "zabbix@company.com"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "7"
        ]
    },
    "id": 1
}
```

Source

CMediaType::create() in *frontends/php/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 *frontends/php/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.

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.
selectUsers	query	Return the users that use the media type in the `users` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `mediatypeid`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mediatypeid": "1",
            "type": "0",
            "description": "Email",
            "smtp_server": "mail.company.com",
            "smtp_helo": "company.com",
            "smtp_email": "zabbix@company.com",
            "exec_path": "",
            "gsm_modem": "",
            "username": "",
            "passwd": "",
            "status": "0"
        },
        {
            "mediatypeid": "2",
            "type": "3",
            "description": "Jabber",
            "smtp_server": "",
            "smtp_helo": "",
            "smtp_email": "",
            "exec_path": "",
            "gsm_modem": "",
            "username": "jabber@company.com",
            "passwd": "zabbix",
            "status": "0"
        },
        {
            "mediatypeid": "3",
            "type": "2",
            "description": "SMS",
            "smtp_server": "",
            "smtp_helo": "",
            "smtp_email": "",
            "exec_path": "",
            "gsm_modem": "/dev/ttyS0",
            "username": "",
            "passwd": "",
```

```
            "status": "0"
        }
    ],
    "id": 1
}
```

See also

- User

Source

CMediaType::get() in *frontends/php/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.

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 *frontends/php/include/classes/api/services/CMediaType.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.isreadable - check if a proxy is readable
- proxy.iswritable - check if a proxy is writable
- 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.

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.
~~interfaces~~ (deprecated)	array	Host interface to be created for the passive proxy passed as an array.

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 *frontends/php/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 *frontends/php/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 hosts monitored by the proxy in the `hosts` property.
selectInterface	query	Return the proxy interface used by a passive proxy in the `interface` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `hostid`, `host` and `status`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	

Parameter	Type	Description
startSearch	flag	

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": "",
            "proxyid": "30091",
            "interface": []
        },
        {
            "host": "Passive proxy",
            "status": "6",
            "lastaccess": "0",
            "description": "",
            "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 *frontends/php/include/classes/api/services/CProxy.php*.

proxy.isreadable

Description

`boolean proxy.isreadable(array proxyIds)`

This method checks if the given proxies are available for reading.

Parameters

`(array)` IDs of the proxies to check.

Return values

`(boolean)` Returns `true` if the given proxies are available for reading.

Examples

Check multiple proxies

Check if the two proxies are readable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.isreadable",
    "params": [
        "30091",
        "30092"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- proxy.iswritable

Source

CProxy::isReadable() in *frontends/php/include/classes/api/services/CProxy.php*.

proxy.iswritable

Description

`boolean proxy.iswritable(array proxyIds)`

This method checks if the given proxies are available for writing.

Parameters

`(array)` IDs of the proxies to check.

Return values

`(boolean)` Returns `true` if the given proxies are available for writing.

Examples

Check multiple proxies

Check if the two proxies are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.iswritable",
    "params": [
        "30091",
        "30092"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- proxy.isreadable

Source

CProxy::isWritable() in *frontends/php/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.
~~interfaces~~ (deprecated)	array	Host interface to be created for the passive proxy passed as an array.

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 *frontends/php/include/classes/api/services/CProxy.php*.

Screen

This class is designed to work with screen.

Object references:

- Screen

Available methods:

- screen.create - creating new screen
- screen.delete - deleting screens
- screen.exists - checking if a screen exists
- 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.
hsize	integer	Width of the screen. Default: 1
vsize	integer	Height of the screen. Default: 1

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.

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": 0,
                "colspan": 0,
                "x": 0,
                "y": 0
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "26"
        ]
    },
    "id": 1
}
```

See also

- Screen item

Source

CScreen::create() in *frontends/php/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 *frontends/php/include/classes/api/services/CScreen.php*.

screen.exists

Description

`boolean screen.exists(object filter)`

This method checks if at least one screen that matches the given filter criteria exists.

> **Warning:**
> This method is deprecated and will be removed in the future. Please use screen.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
name	string/array	Names of the screens.
screenid	string/array	IDs of the screens.

Return values

`(boolean)` Returns `true` if at least one screen that matches the given filter criteria exists.

Examples

Checking a screen by name

Check if a screen named ”Graphs” already exists.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screen.exists",
    "params": {
        "name": "Graphs"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CScreen::exists() in *frontends/php/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.
screenitemids	string/array	Return only screen that contain the given screen items.
selectScreenItems	query	Return the screen items that are used in the screen.
sortfield	string/array	Sort the result by the given properties. Possible values are: `screenid` and `name`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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",
        "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"
                }
            ],
            "screenid": "26",
            "name": "CPU Graphs",
            "hsize": "3",
            "vsize": "2",
            "templateid": "0"
        }
    ],
    "id": 1
}
```

See also

- Screen item

Source

CScreen::get() in *frontends/php/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.

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
}
```

See also

- Screen item
- screenitem.create
- screenitem.update
- screenitem.updatebyposition

Source

CScreen::update() in *frontends/php/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.isreadable - checking if screen items are readable
- screenitem.iswritable - checking if screen items are writable

- 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 - server info; 7 - clock; 8 - screen; 9 - triggers overview 10 - data overview; 11 - URL; 12 - history of actions; 13 - history of events; 14 - latest host group issues; 15 - system status; 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.

Property	Type	Description
max_columns	integer	Specifies the maximum amount of columns a graph prototype or simple graph prototype screen element can have. Default: 3.
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, screen, simple graph and trigger overview screen items. Unused by local and server time clocks, history of actions, history of events, hosts info, server info, system status 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.

Property	Type	Description
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.
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 *frontends/php/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 *frontends/php/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`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary page page.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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 *frontends/php/include/classes/api/services/CScreenItem.php*.

screenitem.isreadable

Description

`boolean screenitem.isreadable(array screenItemIds)`

This method checks if the given screen items are available for reading.

Parameters

`(array)` IDs of the screen items to check.

Return values

`(boolean)` Returns `true` if the given screen items are available for reading.

Examples

Check multiple screen items

Check if the two screen items are readable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screenitem.isreadable",
    "params": [
        "20",
        "21"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
```

```
    "id": 1
}
```

See also

- screenitem.iswritable

Source

CScreenItem::isReadable() in *frontends/php/include/classes/api/services/CScreenItem.php*.

screenitem.iswritable

Description

`boolean screenitem.iswritable(array screenItemIds)`

This method checks if the given screen items are available for writing.

Parameters

`(array)` IDs of the screen items to check.

Return values

`(boolean)` Returns `true` if the given screen items are available for writing.

Examples

Check multiple screen items

Check if the two screen items are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "screenitem.iswritable",
    "params": [
        "20",
        "21"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- screenitem.isreadable

Source

CScreenItem::isWritable() in *frontends/php/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 *frontends/php/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 *frontends/php/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.

Property	Type	Description
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 - *(default)* run on Zabbix server.
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.

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 *frontends/php/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 *frontends/php/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 values: `success` or `failed`.
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 *frontends/php/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 host groups that the script can be run on in the `groups` property.
selectHosts	query	Return hosts that the script can be run on in the `hosts` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `scriptid` and `name`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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 *frontends/php/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 *frontends/php/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 *frontends/php/include/classes/api/services/CScript.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.exists - checking if a template exists
- template.get - retrieving templates
- template.isreadable - checking if templates are readable
- template.iswritable - checking if templates are writable
- 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 host. Default: `host` property 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.
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.

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 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"
            }
```

```
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086"
        ]
    },
    "id": 1
}
```

Source

CTemplate::create() in *frontends/php/include/classes/api/services/CTemplate.php*.

template.delete

Description

`object template.delete(array templateIds)`

This method allows to delete templates.

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 *frontends/php/include/classes/api/services/CTemplate.php*.

template.exists

Description

`boolean template.exists(object filter)`

This method checks if at least one template that matches the given filter criteria exists.

Warning:
This method is deprecated and will be removed in the future. Please use template.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
host	string/array	Technical names of the templates.
name	string/array	Visible names of the templates.
templateid	string/array	Template IDs.

Return values

`(boolean)` Returns `true` if at least one template that matches the given filter criteria exists.

Examples

Check template by name

Check if a template with the technical name ”Linux template” exists.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.exists",
    "params": {
        "host": "Linux template"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- template.isreadable
- template.iswritable

Source

CTemplate::exists() in *frontends/php/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 children of the given templates.
hostids	string/array	Return only templates that are linked to the given hosts.
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.
selectGroups	query	Return the host groups that the template belongs to in the `groups` property.
selectHosts	query	Return the hosts that are linked to the template in the `hosts` property. Supports `count`.
selectTemplates	query	Return the child templates in the `templates` property. Supports `count`.
selectParentTemplates	query	Return the parent templates in the `parentTemplates` property. Supports `count`.
selectHttpTests	query	Return the web scenarios from the template in the `httpSteps` 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`.

Parameter	Type	Description
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`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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": ""
        },
        {
            "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": ""
        }
    ],
    "id": 1
}
```


See also

- template.getobjects
- Host group
- Template
- User macro
- Host interface

Source

CTemplate::get() in *frontends/php/include/classes/api/services/CTemplate.php*.

template.getobjects

Description

`array template.getobjects(object filter)`

This method allows to retrieve templates that match the given filter criteria.

Warning:
This method is deprecated and will be removed in the future. Please use template.get instead.

Parameters

`(object)` Template properties to search by.

Return values

`(array)` Returns an array of objects with all properties.

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.getobjects",
    "params": {
        "host": [
            "Template OS Linux",
            "Template OS Windows"
        ]
    },
```

```
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "host": "Template OS Linux",
            "name": "Template OS Linux",
            "templateid": "10001",
            "description": ""
        },
        {
            "host": "Template OS Windows",
            "name": "Template OS Windows",
            "templateid": "10081",
            "description": ""
        }
    ],
    "id": 1
}
```

See also

- template.get

Source

CTemplate::getObject() in *frontends/php/include/classes/api/services/CTemplate.php*.

template.isreadable

Description

`boolean template.isreadable(array templateIds)`

This method checks if the given templates are available for reading.

Parameters

`(array)` IDs of the templates to check.

Return values

`(boolean)` Returns `true` if the given templates are available for reading.

Examples

Check multiple templates

Check if the two templates are readable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.isreadable",
    "params": [
        "10001",
        "10081"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- template.exists
- template.iswritable

Source

CTemplate::isReadable() in *frontends/php/include/classes/api/services/CTemplate.php*.

template.iswritable

Description

`boolean template.iswritable(array templateIds)`

This method checks if the given templates are available for writing.

Parameters

`(array)` IDs of the templates to check.

Return values

`(boolean)` Returns `true` if the given templates are available for writing.

Examples

Check multiple templates

Check if the two templates are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.iswritable",
    "params": [
        "10001",
        "10081"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- template.isreadable
- template.exists

Source

CTemplate::isWritable() in *frontends/php/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 *frontends/php/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.
macros	string/array	User macros to delete from the given templates.
templateids_clear	string/array	Templates to unlink and clear from the given templates.
templateids_link	string/array	Templates to unlink from the given templates.

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 *frontends/php/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 *frontends/php/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.
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
}
```

Source

CTemplate::update() in *frontends/php/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.exists - check if a template screen exists
- templatescreen.get - retrieve template screens
- templatescreen.isreadable - check if template screens are readable
- templatescreen.iswritable - check if template screens are writable
- 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 *frontends/php/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 *frontends/php/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 *frontends/php/include/classes/api/services/CTemplateScreen.php*.

templatescreen.exists

Description

`boolean templatescreen.exists(object filter)`

This method checks if at least one template screen that matches the given filter criteria exists.

Warning:
This method is deprecated and will be removed in the future. Please use templatescreen.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
name	string/array	Names of the screens.
screenid	string/array	IDs of the screens.
templateids	string/array	IDs of the templates that the screens belong to

Return values

`(boolean)` Returns `true` if at least one template screen that matches the given filter criteria exists.

Examples

Check screen by name

Check if screen ”Zabbix server health” exists on template ”10047”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreen.exists",
    "params": {
        "name": "Zabbix server health",
        "templateid": "10047"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- templatescreen.isreadable
- templatescreen.iswritable

Source

CTemplateScreen::exists() in *frontends/php/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/arary	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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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 *frontends/php/include/classes/api/services/CTemplateScreen.php*.

templatescreen.isreadable

Description

`boolean templatescreen.isreadable(array templateScreenIds)`

This method checks if the given template screens are available for reading.

Parameters

`(array)` IDs of the template screens to check.

Return values

`(boolean)` Returns `true` if the given template screens are available for reading.

Examples

Check multiple template screens

Check if the two template screens are readable.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "templatescreen.isreadable",
    "params": [
        "3",
        "5"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- templatescreen.exists
- templatescreen.iswritable

Source

CTemplateScreen::isReadable() in *frontends/php/include/classes/api/services/CTemplateScreen.php*.

templatescreen.iswritable

Description

`boolean templatescreen.iswritable(array templateScreenIds)`

This method checks if the given template screens are available for writing.

Parameters

`(array)` IDs of the template screens to check.

Return values

`(boolean)` Returns `true` if the given template screens are available for writing.

Examples

Check multiple template screens

Check if the two template screens are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templatescreen.iswritable",
    "params": [
        "3",
        "5"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- templatescreen.isreadable
- templatescreen.exists

Source

CTemplateScreen::isWritable() in *frontends/php/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	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 *frontends/php/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.

Property	Type	Description
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.
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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	

Parameter	Type	Description
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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 *frontends/php/include/classes/api/services/CTemplateScreenItem.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.exists - checking if a trigger exists
- trigger.get - retrieving triggers
- trigger.getobjects - retrieving triggers by filters
- trigger.isreadable - checking if triggers are readable
- trigger.iswritable - checking if triggers are writable
- 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.
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.

Property	Type	Description
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.
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.

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

Source

CTrigger::addDependencies() in *frontends/php/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.

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": "14062"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "14102"
        ]
    },
    "id": 1
}
```

Source

CTrigger::create() in *frontends/php/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 *frontends/php/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 *frontends/php/include/classes/api/services/CTrigger.php*.

trigger.exists

Description

`boolean trigger.exists(object filter)`

This method checks if at least one trigger that matches the given filter criteria exists.

Warning:
This method is deprecated and will be removed in the future. Please use trigger.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
expression (required)	string	Exploded trigger expression.
host (required)	string/array	Technical names of the hosts the triggers must belong to.
hostid (required)	string/array	IDs of the hosts the triggers must belong to.
description	string/array	Names of the triggers.

Note:
Only one of the three parameters is required: `host`, `hostid` or `expression`.

Return values

`(boolean)` Returns `true` if at least one trigger that matches the given filter criteria exists.

Examples

Check a trigger by expression

Check if a trigger with the given expression exists.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.exists",
    "params": {
        "expression": "{Linux server:vfs.file.cksum[/etc/passwd].diff()}>0"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- trigger.isreadable
- trigger.iswritable

Source

CTrigger::exists() in *frontends/php/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 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.
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.
expandComment	flag	Expand macros in the trigger description.
expandDescription	flag	Expand macros in the name of the trigger.

Parameter	Type	Description
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.
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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Parameter	Type	Description
~~expandData~~ (deprecated)	flag	Return additional data about the first host in the trigger expression. Adds the following properties to each trigger: `hostname` - *(string)* visible name of the host; `host` - *(string)* technical name of the host; `hostid` - *(string)* ID of the host.

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": [
        {
            "functions": [
                {
                    "functionid": "13513",
                    "itemid": "24350",
                    "function": "diff",
                    "parameter": "0"
                }
            ],
            "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"
        }
    ],
```

```
    "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
}
```

See also

- trigger.getobjects
- Discovery rule
- Item
- Host
- Host group

Source

CTrigger::get() in *frontends/php/include/classes/api/services/CTrigger.php*.

trigger.getobjects

Description

`array trigger.getobjects(object filter)`

This method allows to retrieve triggers that match the given filter criteria.

Warning:
This method is deprecated and will be removed in the future. Please use trigger.get instead.

Parameters

`(object)` Criteria to search by.

Additionally to the standard standard trigger properties the following parameters are supported as search criteria.

Parameter	Type	Description
host	string/array	Technical name of the host that the trigger belongs to.
hostid	string/array	ID of the host that the trigger belongs to.

Return values

`(array)` Returns an array of objects with all properties.

Examples

Retrieving triggers by name

Retrieve triggers with the name "/etc/passwd has been changed on {HOST.NAME}" from two hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.getobjects",
    "params": {
        "description": "/etc/passwd has been changed on {HOST.NAME}",
        "hostid": [
            "30069",
            "30049"
        ]
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "13938",
            "expression": "{13385}>0",
            "description": "/etc/passwd has been changed on {HOST.NAME}",
            "url": "",
            "status": "0",
            "value": "0",
            "priority": "2",
            "lastchange": "0",
            "comments": "",
            "error": "Agent is unavailable.",
            "templateid": "10016",
            "type": "0",
            "value_flags": "1",
            "flags": "0"
        },
        {
            "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",
            "value_flags": "0",
            "flags": "0"
        }
    ],
    "id": 1
}
```

See also

- trigger.get

Source

CTrigger::getObject() in *frontends/php/include/classes/api/services/CTrigger.php*.

trigger.isreadable

Description

`boolean trigger.isreadable(array triggerIds)`

This method checks if the given triggers are available for reading.

Parameters

`(array)` IDs of the triggers to check.

Return values

`(boolean)` Returns `true` if the given triggers are available for reading.

Examples

Check multiple triggers

Check if the two triggers are readable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.isreadable",
    "params": [
        "13938",
        "14062"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- trigger.exists
- trigger.iswritable

Source

CTrigger::isReadable() in *frontends/php/include/classes/api/services/CTrigger.php*.

trigger.iswritable

Description

`boolean trigger.iswritable(array triggerIds)`

This method checks if the given triggers are available for writing.

Parameters

`(array)` IDs of the triggers to check.

Return values

`(boolean)` Returns `true` if the given triggers are available for writing.

Examples

Check multiple triggers

Check if the two triggers are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.iswritable",
    "params": [
        "13938",
        "14062"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- trigger.isreadable
- trigger.exists

Source

CTrigger::isWritable() in *frontends/php/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.

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
}
```

See also

- trigger.adddependencies
- trigger.deletedependencies

Source

CTrigger::update() in *frontends/php/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

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.
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.

triggerprototype.create

Description

`object triggerprototype.create(object/array triggerPrototypes)`

This method allows to create new trigger prototypes.

Parameters

`(object/array)` Trigger prototypes to create.

The method accepts trigger prototypes with the standard trigger prototype properties.

> **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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "15331"
        ]
    },
    "id": 1
}
```

Source

CTriggerPrototype::create() in *frontends/php/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 *frontends/php/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.

Parameter	Type	Description
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.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: `host` - technical name of the host that the trigger prototype belongs to; `hostid` - ID of the host that the trigger prototype belongs to.
limitSelects	integer	Limits the number of records returned by subselects. 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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	
~~expandData~~ (deprecated)	flag	Return additional data about the first host in the trigger expression. Adds the following properties to each trigger prototype: `hostname` - *(string)* visible name of the host; `host` - *(string)* technical name of the host; `hostid` - *(string)* ID of the host.

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": [
        {
            "functions": [
                {
                    "functionid": "12598",
                    "itemid": "22454",
                    "function": "last",
                    "parameter": "0"
                }
            ],
            "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"
        },
        {
            "functions": [
                {
                    "functionid": "13500",
                    "itemid": "22686",
                    "function": "last",
                    "parameter": "0"
                }
            ],
            "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"
        }
    ],
    "id": 1
}
```

See also

- Discovery rule
- Item
- Host
- Host group

Source

CTriggerPrototype::get() in *frontends/php/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.

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
}
```

Source

CTriggerPrototype::update() in *frontends/php/include/classes/api/services/CTriggerPrototype.php*.

User

This class is designed to work with users.

Object references:

- User

Available methods:

- user.addmedia - adding media to users
- user.create - creating new users
- user.delete - deleting users
- user.deletemedia - deleting media from users
- user.get - retrieving users
- user.isreadable - checking if users are readable
- user.iswritable - checking if users are writable
- user.login - logging in to the API
- user.logout - logging out of the API
- user.update - updating users
- user.updatemedia - updating user media
- user.updateprofile - updating the currently logged in user

> 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	integer	User session life time in seconds. If set to 0, the session will never expire. Default: 900.
lang	string	Language code of the user's language. Default: `en_GB`.
name	string	Name of the user.
refresh	integer	Automatic refresh period in seconds. Default: 30.
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; `classic` - Classic; `originalblue` - Original blue; `darkblue` - Black & Blue; `darkorange` - Dark orange.
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.

user.addmedia

Description

`object user.addmedia(object parameters)`

This method allows to add new media to multiple users.

Parameters

`(object)` Parameters defining the media to create and the users to add them to.

Parameter	Type	Description
medias (required)	object/array	Media to create for the given users. The media `userid` property must not be defined.
users (required)	object/array	Users to add the media to. The users must have the `userid` property defined.

Return values

`(object)` Returns an object containing the IDs of the created media under the `mediaids` property.

Examples

Adding a media to multiple users

Create a common e-mail media for two users. The media must send notifications about all alerts at any time.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.addmedia",
    "params": {
        "users": [
            {
                "userid": "1"
            },
            {
                "userid": "2"
            }
        ],
        "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": {
        "mediaids": [
            "12",
            "13"
        ]
    },
    "id": 1
}
```

See also

- user.update
- user.updatemedia
- Media
- User

Source

CUser::addMedia() in *frontends/php/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.
usrgrps (required)	array	User groups to add the user to. The user groups must have the `usrgrpid` property defined.
user_medias	array	Media to create for the user. The media `userid` property must not be defined.

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 *frontends/php/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 *frontends/php/include/classes/api/services/CUser.php*.

user.deletemedia

Description

`object user.deletemedia(string/array mediaIds)`

This method allows to delete media.

Parameters

`(string/array)` IDs of the media to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted media under the `mediaids` property.

Examples

Deleting multiple media

Delete two media.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.deletemedia",
    "params": [
        "11",
        "13"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediaids": [
            "11",
```

```
            "13"
        ]
    },
    "id": 1
}
```

See also

- user.update
- user.updatemedia

Source

CUser::deleteMedia() in *frontends/php/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.

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.
sortfield	string/array	Sort the result by the given properties. Possible values are: `userid` and `alias`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	

Parameter	Type	Description
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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": "0",
            "lang": "ru_RU",
            "refresh": "0",
            "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": "900",
            "lang": "en_GB",
            "refresh": "30",
            "type": "1",
            "theme": "default",
            "attempt_failed": "0",
```

```
            "attempt_ip": "",
            "attempt_clock": "0",
            "rows_per_page": "50"
        }
    ],
    "id": 1
}
```

See also

- Media
- Media type
- User group

Source

CUser::get() in *frontends/php/include/classes/api/services/CUser.php*.

user.isreadable

Description

`boolean user.isreadable(array userIds)`

This method checks if the given users are available for reading.

Parameters

`(array)` IDs of the users to check.

Return values

`(boolean)` Returns `true` if the given users are available for reading.

Examples

Check multiple users

Check if the two users are readable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.isreadable",
    "params": [
        "4",
        "6"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- user.iswritable

Source

CUser::isReadable() in *frontends/php/include/classes/api/services/CUser.php*.

user.iswritable

Description

`boolean user.iswritable(array userIds)`

This method checks if the given users are available for writing.

Parameters

`(array)` IDs of the users to check.

Return values

`(boolean)` Returns `true` if the given users are available for writing.

Examples

Check multiple users

Check if the two users are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.iswritable",
    "params": [
        "4",
        "6"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- user.isreadable

Source

CUser::isWritable() in *frontends/php/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.

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. Unused for HTTP authentication.

Parameter	Type	Description
user (required)	string	User name.
userData	flag	Return information about the authenticated user.

Attention:
When using HTTP authentication, the user name in the API request must match the one used in the `Authorization` header. The password will not be validated and can be omitted.

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 *frontends/php/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 *frontends/php/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.
usrgrps	array	User groups to replace existing user groups. The user groups must have the `usrgrpid` property defined.

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
}
```

See also

- user.updateprofile

Source

CUser::update() in *frontends/php/include/classes/api/services/CUser.php*.

user.updatemedia

Description

`object user.updatemedia(object parameters)`

This method allows to update media for multiple users.

Parameters

`(object)` Parameters defining the media and users to be updated.

Parameter	Type	Description
medias (required)	object/array	Media to replace existing media. If a media has the `mediaid` property defined it will be updated, otherwise a new media will be created.
users (required)	object/array	Users to update. The users must have the `userid` property defined.

Return values

`(object)` Returns an object containing the IDs of the updated users under the `userids` property.

Examples

Replacing media for multiple users

Replace all media used by the two users with a common e-mail media. The media must send notifications about all alerts at any time.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.updatemedia",
    "params": {
        "users": [
            {
                "userid": "1"
            },
            {
                "userid": "2"
```

```
            }
        ],
        "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": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

See also

- user.addmedia
- user.deletemedia
- user.updatemedia
- Media
- User

Source

CUser::updateMedia() in *frontends/php/include/classes/api/services/CUser.php*.

user.updateprofile

Description

`object user.updateprofile(object parameters)`

This method allows to update the currently logged in user.

Parameters

`(object/array)` User properties to be updated.

The `userid` property must not be defined. 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.
usrgrps	array	User groups to replace existing user groups. The user groups must have the `usrgrpid` property defined.

Return values

`(object)` Returns an object containing the ID of the updated user under the `userids` property.

Examples

Renaming the current user

Rename the current user to John Doe.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.updateprofile",
    "params": {
        "name": "John",
        "lastname": "Doe"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "1"
        ]
    },
    "id": 1
}
```

See also

- user.update

Source

CUser::update() in *frontends/php/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.exists - checking if a user group exists
- usergroup.get - retrieving user groups
- usergroup.getobjects - retrieving user groups by filters
- usergroup.isreadable - checking if user groups are readable
- usergroup.iswritable - checking if user groups are writable
- usergroup.massadd - adding permissions and users to user groups
- usergroup.massupdate - simultaneously updating multiple 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.
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 - disable access to the frontend.
users_status	integer	Whether the user group is enabled or disabled. Possible values are: 0 - *(default)* enabled; 1 - disabled.

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.

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
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 *frontends/php/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 *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.exists

Description

`boolean usergroup.exists(object filter)`

This method checks if at least one user group that matches the given filter criteria exists.

Warning:
This method is deprecated and will be removed in the future. Please use usergroup.get instead.

Parameters

`(object)` Criteria to search by.

The following parameters are supported as search criteria.

Parameter	Type	Description
name	string/array	Names of the user groups.

Return values

`(boolean)` Returns `true` if at least one user group that matches the given filter criteria exists.

Examples

Checking if a user group exists

Check if user group "Zabbix administrators"

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.exists",
    "params": {
        "name": "Zabbix administrators"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
```

```
    "id": 1
}
```

See also

- usergroup.isreadable
- usergroup.iswritable

Source

CUserGroup::exists() in *frontends/php/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.
with_gui_access	integer	Return only user groups with the given frontend authentication method. Refer to the user group page for a list of supported methods.
selectUsers	query	Return the users from the user group in the `users` property.
limitSelects	integer	Limits the number of records returned by subselects.
sortfield	string/array	Sort the result by the given properties. Possible values are: `usrgrpid`, `name`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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

- usergroup.getobjects
- User

Source

CUserGroup::get() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.getobjects

Description

`array usergroup.getobjects(object filter)`

This method allows to retrieve user groups that match the given filter criteria.

Warning:
This method is deprecated and will be removed in the future. Please use usergroup.get instead.

Parameters

`(object)` Criteria to search by.

Additionally to the standard standard user group properties the following parameters are supported as search criteria.

Parameter	Type	Description
name	string	Name of the user group.

Return values

`(array)` Returns an array of objects with all properties.

Examples

Retrieving a user group by name

Retrieve all data about the user group ”Zabbix administrators”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.getobjects",
    "params": {
        "name": "Zabbix administrators"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "usrgrpid": "7",
            "name": "Zabbix administrators",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "1"
        }
    ],
```

```
    "id": 1
}
```

See also

- usergroup.get

Source

CUserGroup::getObject() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.isreadable

Description

`boolean usergroup.isreadable(array userGroupIds)`

This method checks if the given user groups are available for reading.

Parameters

`(array)` IDs of the user groups to check.

Return values

`(boolean)` Returns `true` if the given user groups are available for reading.

Examples

Check multiple user groups

Check if the two user groups are readable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.isreadable",
    "params": [
        "21",
        "22"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- usergroup.exists
- usergroup.iswritable

Source

CUserGroup::isReadable() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.iswritable

Description

`boolean usergroup.iswritable(array userGroupIds)`

This method checks if the given user groups are available for writing.

Parameters

(array) IDs of the user groups to check.

Return values

(boolean) Returns true if the given user groups are available for writing.

Examples

Check multiple user groups

Check if the two user groups are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.iswritable",
    "params": [
        "21",
        "22"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- usergroup.isreadable
- usergroup.exists

Source

CUserGroup::isWritable() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.massadd

Description

`object usergroup.massadd(object parameters)`

This method allows to simultaneously add permissions and users to multiple user groups.

Parameters

(object) Parameters containing the IDs of the user groups to update and the permissions and users to add.

The method accepts the following parameters.

Parameter	Type	Description
usrgrpids (required)	string/array	IDs of user groups to update.
rights	object/array	Permissions to assign to the user groups.
userids	string/array	IDs of the users to add to the user groups.

Return values

(object) Returns an object containing the IDs of the updated user groups under the usrgrpids property.

Examples

Denying access to host group

Deny two user groups access to host group "2".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.massadd",
    "params": {
        "usrgrpids": [
            "17",
            "19"
        ],
        "rights": {
            "permission": 0,
            "id": "2"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "17",
            "19"
        ]
    },
    "id": 1
}
```

See also

- Permission
- usergroup.massupdate
- usergroup.update

Source

CUserGroup::massAdd() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.massupdate

Description

`object usergroup.massupdate(object parameters)`

This method allows to simultaneously update properties, users or permissions for multiple user groups.

Parameters

`(object)` Parameters containing the IDs of the user groups to update and the properties that should be updated.

Additionally to the standard user group properties, the method accepts the following parameters.

Parameter	Type	Description
usrgrpids (required)	string/array	IDs of user groups to update.
rights	string/array	Permissions to replace the current permissions assigned to the user group.
userids	object/array	IDs of the users to replace the users in the group.

Return values

`(object)` Returns an object containing the IDs of the updated user groups under the `usrgrpids` property.

Examples

Changing permissions for a user group

Update the permissions for two user groups to only allow read-write access to two host groups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.massupdate",
    "params": {
        "usrgrpids": [
            "17",
            "19"
        ],
        "rights": [
            {
                "permission": 3,
                "id": "2"
            },
            {
                "permission": 3,
                "id": "3"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "17",
            "19"
        ]
    },
    "id": 1
}
```

See also

- Permission
- usergroup.massadd
- usergroup.update

Source

CUserGroup::massUpdate() in *frontends/php/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.
userids	string/array	IDs of the users to replace the users in the 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
- usergroup.massadd
- usergroup.massupdate

Source

CUserGroup::update() in *frontends/php/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.

Host macro

The host macro object defines a macro available on a host 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.

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

Creat 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 *frontends/php/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 *frontends/php/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 *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.deleteglobal

Description

`object usermacro.deleteglobal(array globalMacroIds)`

This method allows to delete global macros.

Parameters

`(string/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 *frontends/php/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.
hostids	string/array	Return only host macros that belong to the given hosts.
hostmacroids	string/array	Return only host macros with the given IDs.
templateids	string/array	Return only host macros that belong to the given templates.
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.

Parameter	Type	Description
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	flag	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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": "{$INTERFACE}",
            "value": "eth0"
        },
        {
            "hostmacroid": "11",
            "hostid": "10198",
            "macro": "{$SNMP_COMMUNITY}",
            "value": "public"
        }
```

```
    ],
    "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"
        }
    ],
    "id": 1
}
```

Source

CUserMacro::get() in *frontends/php/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 *frontends/php/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 *frontends/php/include/classes/api/services/CUserMacro.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.isreadable - checking if web scenarios are readable
- httptest.iswritable - checking if web scenarios are writable
- 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.
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	integer	Execution interval of the web scenario in seconds. Default: 60.
headers	string	HTTP headers that will be sent when performing a request.
http_password	string	Password used for authentication. Required for web scenarios with 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 authentication. Required for web scenarios with basic HTTP or NTLM authentication.

Property	Type	Description
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.
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	string	Web scenario variables.
verify_host	integer	Whether to verify that the host name specified in the SSL certificate matches the one used in the scenario. Possible values are: 0 - *(default)* skip host verification; 1 - verify host.
verify_peer	integer	Whether to verify the SSL certificate of the web server. Possible values are: 0 - *(default)* skip peer verification; 1 - verify peer.

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	string	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	HTTP POST variables as a string.
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.

Property	Type	Description
status_codes	string	Ranges of required HTTP status codes separated by commas.
timeout	integer	Request timeout in seconds. Default: 15.
variables	string	Scenario step variables.

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.

Note:
The `hostid` parameter can be omitted if the `applicationid` parameter is given. In that case, the web scenario will be assigned to the host that the application belongs to.

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://mycompany.com",
                "status_codes": 200,
                "no": 1
            },
            {
                "name": "Homepage / About",
                "url": "http://mycompany.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 *frontends/php/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 *frontends/php/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 host that the web scenario belongs to as an array in the `hosts` property.
selectSteps	query	Return web scenario steps in the `steps` property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `httptestid` and `name`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

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": "60",
            "status": "0",
            "variables": "",
            "agent": "",
            "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://mycompany.com",
                    "timeout": "15",
                    "posts": "",
                    "required": "",
                    "status_codes": "200",
                    "variables": "",
                    "follow_redirects": "1",
                    "retrieve_mode": "0",
                    "headers": ""
                },
```

```
                {
                    "httpstepid": "37",
                    "httptestid": "9",
                    "name": "Homepage / About",
                    "no": "2",
                    "url": "http://mycompany.com/about",
                    "timeout": "15",
                    "posts": "",
                    "required": "",
                    "status_codes": "200",
                    "variables": "",
                    "follow_redirects": "1",
                    "retrieve_mode": "0",
                    "headers": ""
                }
            ]
        }
    ],
    "id": 1
}
```


See also

- Host
- Scenario step

Source

CHttpTest::get() in *frontends/php/include/classes/api/services/CHttpTest.php*.

httptest.isreadable

Description

`boolean httptest.isreadable(array webScenarioIds)`

This method checks if the given web scenarios are available for reading.

Parameters

`(array)` IDs of the web scenarios to check.

Return values

`(boolean)` Returns `true` if the given web scenarios are available for reading.

Examples

Check multiple web scenarios

Check if the two web scenarios are readable.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "httptest.isreadable",
    "params": [
        "3",
        "5"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```


Response:


```
{
    "jsonrpc": "2.0",
    "result": true,
```

```
    "id": 1
}
```

See also

- httptest.iswritable

Source

CHttpTest::isReadable() in *frontends/php/include/classes/api/services/CHttpTest.php*.

httptest.iswritable

Description

`boolean httptest.iswritable(array webScenarioIds)`

This method checks if the given web scenarios are available for writing.

Parameters

`(array)` IDs of the web scenarios to check.

Return values

`(boolean)` Returns `true` if the given web scenarios are available for writing.

Examples

Check multiple web scenarios

Check if the two web scenarios are writable.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.iswritable",
    "params": [
        "3",
        "5"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- httptest.isreadable

Source

CHttpTest::isWritable() in *frontends/php/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 *frontends/php/include/classes/api/services/CHttpTest.php*.

Appendix 1. Reference commentary

Notation Data types

The Zabbix API supports the following data types:

Type	Description
bool	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` and `false` otherwise.
integer	A whole number.
float	A floating point number.
string	A text string.
text	A longer text string.

Type	Description
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.

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.

Common "get" method parameters The following parameters are supported by all `get` methods:

Parameter	Type	Description
countOutput	flag	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	flag	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 array, where the keys are property names, and the values are either a single value or an array of values to match against. Doesn't work for `text` fields.
limit	integer	Limit the number of records returned.
output	query	Object properties to be returned. Default: `extend`.
preservekeys	flag	Use IDs as keys in the resulting array.
search	object	Return results that match the given wildcard search. 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. Works only for `string` and `text` fields.
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`.

Parameter	Type	Description
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.
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` - ascending; `DESC` - descending.
startSearch	flag	The `search` parameter will compare the beginning of fields, that is, perform a `LIKE "…%"` search instead.

Appendix 2. Changes from 2.2 to 2.4

Backward incompatible changes General

ZBXNEXT-1343 dropped support of all node-related parameters
ZBXNEXT-2017 dropped support of output ”refer” for all get methods
ZBXNEXT-2017 changed the default value of the ”output” parameter to ”extend”

action

ZBXNEXT-1133 replaced the conditions array with the action filter object

alert

ZBX-7345 alert.get: removed the deprecated triggerids parameter

apiinfo

ZBX-7781 apiinfo.version: calling the method with the ”auth” parameter will now trigger an error

discoveryrule

ZBXNEXT-581 changed the ”filter” property to contain a filter object instead of a string

event

ZBX-7345 event.get: removed the deprecated selectTriggers, selectItems and triggerids parameters
ZBX-7345 event.get: removed the deprecated object value for the sortfield parameter

host

ZBX-7345 host.delete: passing array of objects is no longer supported, only array of IDs

httptest

ZBX-7345 dropped support of the deprecated webcheck alias
ZBX-7345 removed the deprecated HTTP test macros property
ZBX-7345 removed the deprecated HTTP test step webstepid property

image

ZBX-8118 image.update: removed possibility to change image type

proxy

ZBX-7345 proxy.delete: passing array of objects is no longer supported, only array of IDs
ZBX-7345 proxy.get: removed the deprecated selectInterfaces parameter

trigger

ZBX-7345 removed the deprecated value_flag property

user

ZBX-7345 user.authenticate: removed deprecated method
ZBX-7345 user.delete: passing array of objects is no longer supported, only array of IDs
ZBX-7781 user.login: calling the method with the ”auth” parameter will now trigger an error

Other changes and bug fixes General

Changes:
ZBX-7981 all of the ”exists” methods have been deprecated
ZBXNEXT-2398 all of the ”getObjects” methods have been deprecated
ZBXNEXT-2398 all of the ”expandData” parameters have been deprecated

application

Changes:
ZBXNEXT-2398 application.get: added ”selectHost” parameter; the ”selectsHosts” parameter has been deprecated

configuration

Changes:
ZBXNEXT-219 configuration.import: added parameter ”deleteMissing” for parameters ”applications”, ”items”, ”discoveryRules”, ”triggers”, ”graphs” and ”templateScreens”
Bug fixes:
ZBXNEXT-219 configuration.import: fixed ”createMissing” parameter for applications
ZBX-8216 configuration.import: fixed bug importing multiple screens
ZBX-8791 configuration.import: fixed bug when importing maps when linked elements have linked triggers

discoveryrule

Changes:
ZBXNEXT-581 discoveryrule.get: implemented the ”selectFilter” parameter

drule

Bug fixes:
ZBX-8301 restricted ”unique” flag in checks to SNMP, Zabbix agent and IP checks
ZBX-8430 added cascade deletion of action condition when deleting connected checks

graph

Bug fixes:
ZBX-7949 fixed child graph name validation to case be case sensitive

graphprototype

Bug fixes:
ZBX-7949 fixed child graph prototype name validation to case be case sensitive

host

Changes:
ZBXNEXT-2117 implemented the ”description” property
ZBXNEXT-2123 host.get: implemented the ”searchInventory” parameter
Bug fixes:
ZBX-4772 fixed status validation

ZBX-7981 host.massupdate: fixed validation when renaming host to an existing template
ZBX-8530 host.update, host.massupdate: fixed order in which template linking and interface updates are done

hostgroup

Bug fixes:
ZBX-7981 hostgroup.create: fixed duplicate name validation
ZBX-5567 hostgroup.get: removed confusing parameter ”not_proxy_hosts”

hostinterface

Changes:
ZBXNEXT-2301 implemented the ”bulk” property

httptest

Changes:
ZBXNEXT-282 implemented web scenario ”headers”, ”verify_peer”, ”verify_host”, ”ssl_cert_file”, ”ssl_key_file” and ”ssl_key_password” properties
ZBXNEXT-282 implemented web scenario step ”follow_redirects”, ”retrieve_mode” and ”headers” properties
ZBXNEXT-282 increased the length of web scenario step ”url” property to 2048 symbols
Bug fixes:
ZBX-8194 fixed SQL errors generated by missing hosts and duplicate web scenario names, and added missing web step validation
ZBX-8817 httptest.delete: fixed incorrectly generated audit log message with quatation marks instead or brackets
ZBX-8195 httptest.create, httptest.update: fixed step name and URL validation

image

Bug fixes:
ZBX-8132 image.get: fixed not properly returning all images for sysmaps

item

Bug fixes:
ZBX-8568 item.delete: fixed to also delete trigger prototypes using deleted items

itemprototype

Bug fixes:
ZBX-8568 itemprototype.delete: fixed to also delete trigger prototypes using deleted item prototypes

maintenance

Changes:
ZBXNEXT-1910 maintenance.get: allowed sorting by the ”active_till” and ”active_since” properties.
Bug fixes:
ZBX-7981 fixed duplicate name validation
=== map ===

Changes:
ZBXNEXT-2124 implemented the ”application” map element filter
ZBX-8950 map.get: fixed ”selecIconMap” option
=== proxy ===

Changes:
ZBXNEXT-2117 implemented the ”description” property

screenitem

Changes:
ZBXNEXT-1583 implemented two new ”resourcetype” property values: 19 - ”simple graph prototype”, 20 - ”graph prototype”
ZBXNEXT-1583 implemented new property ”max_columns” with interger value range 1-100
Bug fixes:
ZBX-7921 changed validation of colspan and rowspan values to be bigger than 0

ZBX-8764 screenitem.create, screenitem.update: improved validation of resource ID to disallow it for screen element types that do not use a resource
ZBX-8764 screenitem.update: fixed code to set resource ID to 0 for element types that do not use it

service

Bug fixes:
ZBX-8169 service.getsla: fixed SLA period calculation to account for DST changes

template

Changes:
ZBXNEXT-2117 implemented the ”description” property
Bug fixes:
ZBX-7981 template.massupdate: fixed validation when renaming template to an existing host
=== trigger ===

Changes:
ZBXNEXT-1782 implemented the new ”<>”, ”<=”, ”>=”, ”not”, ”or” and ”and” operators
Bug fixes:
ZBX-6174 trigger.get: fixed ”skipDependent” option not handling cases when triggers upon which other triggers depend are disabled (or have disabled items or disabled item hosts).
ZBX-8227 trigger.get: fixed ”applicationids” overriding ”groupids”
ZBX-6834 trigger.get: fixed the ”host” parameter not returning any results
ZBX-6596 trigger.get: fixed trigger sorting by hostname

Zabbix API changes in 2.4

2.4.8 map

Bug fixes:
ZBX-10251 map.get: fixed ”countOutput” calculation for unprivileged users

screen

Bug fixes:
ZBX-10150 screen.get: fixed ”countOutput” calculation for unprivileged users
ZBX-10369 screen.update: fixed unexpected deleting of screen items when updating both screen size and screen items

service

Bug fixes:
ZBX-10232 service.getSla: fixed SQL errors with invalid ”year” parameter in IT services report

2.4.7 General

Bug fixes:
ZBX-9340 fixed ”data” property not being returned by API when error is generated on DB level

discoveryrule

Bug fixes:
ZBX-9866 discoveryrule.copy: fixed error field being needlessly copied for new LLD rule

hostgroup

Bug fixes:
ZBX-9738 hostgroup.delete: fixed deletion of related action operations when deleting a host group

item

Bug fixes:
ZBX-8235 item.update: fixed losing initial values when updating templated items

2.4.5 configuration

Bug fixes:
ZBX-9346 configuration.import: fixed importing trigger dependencies that contain older expression format

maintenance

Bug fixes:
ZBX-5656 fixed ”timeperiods” validation when passing a single timeperiod object

2.4.4 action

Bug fixes:
ZBXNEXT-2160 added support for extended IP ranges like ”192.168.1-10.1-254” in action conditions

drule

Bug fixes:
ZBXNEXT-2160 added support for extended IP ranges like ”192.168.1-10.1-254”

host

Bug fixes:
ZBX-8448 host.update, host.massupdate, host.massadd: fixed ”groups” property to also accept read-only ”groupid” if host currently belongs to both read and read-write groups
ZBX-9093 host.create: fixed ”inventory” property causing SQL errors when using MySQL strict mode

hostgroup

Bug fixes:
ZBX-8448 hostgroup.massupdate, hostgroup.massremove: fixed permissions validation to no longer silently remove hosts and templates to which user has no write permissions

template

Bug fixes:
ZBX-8448 template.massadd: fixed ”groups” property to also accept read-only ”groupid” if template currently belongs to both read and read-write groups
ZBX-8448 template.update, template.massupdate: fixed ”hosts” property to no longer silently remove hosts and templates to which user has no write permissions

user

Bug fixes:
ZBX-9224 user.login: fixed audit log for failed login attempts
==== 2.4.3 ====

hostgroup

Bug fixes:
ZBX-9017 hostgroup.create, hostgroup.update: fixed methods accepting readonly ”internal” param for create/update.

httptest

Bug fixes:
ZBX-8486 fixed web scenario re-linking

screenitem

Bug fixes:
ZBX-9065 fixed the default value for max_columns field

2.4.2 application

Bug fixes:
ZBX-8832 application.create: fixed template application inheritance when template is linked to one or more templates having applications with same name

item

Bug fixes:
ZBX-8904 item.get: fixed ”selectInterfaces” option retrieving all host interfaces

18. 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 1.8 (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 colour 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.

Installation troubleshooting

See the installation-specific troubleshooting section.

See also

* Troubleshooting page on zabbix.org

2 Installation

1 Database creation scripts

Overview

A Zabbix database must be created during the installation of Zabbix server or proxy.

This section provides scripts for creating a Zabbix database. A separate schema script is provided for each supported database.

Note:
`schema.sql`, `images.sql` and `data.sql` files are located in the *database* subdirectory of Zabbix sources. If Zabbix was installed from distribution packages, refer to the distribution documentation.

Attention:
For a Zabbix proxy database, **only** `schema.sql` should be imported (no images.sql nor data.sql)

Scripts

MySQL

```
shell> mysql -uroot -p<password>
mysql> create database zabbix character set utf8 collate utf8_bin;
mysql> grant all privileges on zabbix.* to zabbix@localhost identified by '<password>';
mysql> quit;
shell> mysql -uzabbix -p<password> zabbix < database/mysql/schema.sql
# stop here if you are creating database for Zabbix proxy
shell> mysql -uzabbix -p<password> zabbix < database/mysql/images.sql
shell> mysql -uzabbix -p<password> zabbix < database/mysql/data.sql
```

PostgreSQL

We assume that a *username* user exists and has permissions to create database objects.

```
shell> psql -U <username>
psql> create database zabbix;
psql> \q
shell> cd database/postgresql
shell> psql -U <username> zabbix < schema.sql
# stop here if you are creating database for Zabbix proxy
shell> psql -U <username> zabbix < images.sql
shell> psql -U <username> zabbix < data.sql
```

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.

If you are creating a database for Zabbix server you need to have images somewhere on the Oracle host, for example in /tmp/zabbix_images folder. Copy all images from misc/images/png_modern to /tmp/zabbix_images directory on the Oracle host:

```
shell> cd /path/to/zabbix-sources
shell> ssh user@host "mkdir /tmp/zabbix_images"
shell> scp -r misc/images/png_modern user@host:/tmp/zabbix_images/
```

Edit the database/oracle/images.sql file and set `image_dir` variable to the /tmp/zabbix_images path:

```
CREATE OR REPLACE DIRECTORY image_dir AS '/tmp/zabbix_images'
```

Now prepare the database:

```
shell> sqlplus zabbix/password@host/ORCL
sqlplus> @database/oracle/schema.sql
# stop here if you are creating database for Zabbix proxy
sqlplus> @database/oracle/images.sql
sqlplus> @database/oracle/data.sql
```

Note:
Zabbix requires a UTF-8 database character set. If database is not UTF-8, it can be converted by running: `ALTER DATABASE NATIONAL CHARACTER SET UTF8;`

IBM DB2

```
shell> db2 "create database zabbix using codeset utf-8 territory us pagesize 32768"
shell> cd database/ibm_db2
shell> db2batch -d zabbix -f schema.sql
# stop here if you are creating database for Zabbix proxy
shell> db2batch -d zabbix -f images.sql
shell> db2batch -d zabbix -f data.sql
```

Note:
It is important to set UTF-8 locale for Zabbix server, Zabbix proxy and web server running Zabbix frontend. Otherwise text information from Zabbix will be interpreted by IBM DB2 server as non-UTF-8 and will be additionally converted on the way from Zabbix to the database and back. The database will store corrupted non-ASCII characters.

Zabbix frontend uses `OFFSET` and `LIMIT` clauses in SQL queries. For this to work, IBM DB2 server must have DB2_COMPATIBILITY_VECTOR variable be set to 3. Run the following command before starting the database server:

```
shell> db2set DB2_COMPATIBILITY_VECTOR=3
```

SQLite

```
shell> cd database/sqlite3
shell> sqlite3 /var/lib/sqlite/zabbix.db < schema.sql
# stop here if you are creating database for Zabbix proxy
shell> sqlite3 /var/lib/sqlite/zabbix.db < images.sql
shell> sqlite3 /var/lib/sqlite/zabbix.db < data.sql
```

Note:
If using SQLite with Zabbix proxy, database will be automatically created if it does not exist.

Return to the installation section.

2 Zabbix agent on Microsoft Windows

Configuring agent

Zabbix agent runs as a Windows service.

You can run a single instance of Zabbix agent or multiple instances of the agent on a Microsoft Windows host. A single instance can use the default configuration file `C:\zabbix_agentd.conf` or a configuration file specified in the command line. In case of multiple instances each agent instance must have its own configuration file (one of the instances can use the default configuration file).

An example configuration file is available in Zabbix source archive as `conf/zabbix_agentd.win.conf`.

See the configuration file options for details on configuring Zabbix Windows agent.

Hostname parameter

To perform active checks on a host Zabbix agent needs to have the hostname defined. Moreover, the hostname value set on the agent side should exactly match the "Host name" configured for the host in the frontend.

The hostname value on the agent side can be defined by either the **Hostname** or **HostnameItem** parameter in the agent configuration file - or the default values are used if any of these parameters are not specified.

The default value for **HostnameItem** parameter is the value returned by the "system.hostname" agent key and for Windows platform it returns the NetBIOS host name.

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.

Attention:
The **system.hostname** key always returns the NetBIOS host name which is limited to 15 symbols and in UPPERCASE only - regardless of the length and lowercase/uppercase characters in the real host name.

Starting from Zabbix agent 1.8.6 version for Windows the "system.hostname" key supports an optional parameter - *type* of the name. The default value of this parameter is "netbios" (for backward compatibility) and the other possible value is "host".

Attention:
The **system.hostname[host]** key always returns the full, real (case sensitive) Windows host name.

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]
   and Zabbix agent will use the full, real (case sensitive) Windows host name as the hostname.

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
```

3 Troubleshooting installation issues

This page covers potential issues that could arise during installation of Zabbix

Access control with Apache

Zabbix frontend includes *.htaccess* files that limit access to directories *api*, *conf* and *include*. Since Apache 2.4 introduced a new access control syntax, running versions of Zabbix 2.2.1 and older might cause the following error to appear in the Apache error logs:

```
Invalid command 'Order', perhaps misspelled or defined by a module not included in the server configuratio
```

Using the outdated syntax in the Apache configuration files might prevent Apache from starting at all. To solve this problem either upgrade to Zabbix 2.2.2 or enable the Apache *mod_access_compat* module.

See Apache documentation for details.

3 Daemon configuration

1 Zabbix server

The parameters supported in a Zabbix server configuration file:

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-8G	8M	Size of configuration cache, in bytes. Shared memory size for storing host, item and trigger data. Upper limit used to be 2GB before Zabbix 2.2.3.
CacheUpdateFrequency	no	1-3600	60	How often Zabbix will perform update of configuration cache, in seconds.
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. For SQLite3 path to database file must be provided. DBUser and DBPassword are ignored.
DBPassword	no			Database password. Ignored for SQLite. Comment this line if no password is used.
DBPort	no	1024-65535	3306	Database port when not using local socket. Ignored for SQLite.
DBSchema	no			Schema name. Used for IBM DB2 and PostgreSQL.
DBSocket	no		/tmp/mysql.sock	Path to MySQL socket.
DBUser	no			Database user. Ignored for SQLite.
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 debuging for web and VMware monitoring (can be set only using runtime control options)
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.

Parameter	Mandatory	Range	Default	Description
FpingLocation	no		/usr/sbin/fping	Location of fping. Make sure that fping binary has root ownership and SUID flag set!
HistoryCacheSize	no	128K-2G	8M	Size of history cache, in bytes. Shared memory size for storing history data.
HistoryTextCacheSize	no	128K-2G	16M	Size of text history cache, in bytes. Shared memory size for storing character, text or log history data.
HousekeepingFrequency	no	1-24	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.**
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.

Parameter	Mandatory	Range	Default	Description
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.
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.
LoadModule	no			Module to load at server startup. Modules are used to extend functionality of the server. Format: LoadModule=<module.so> The modules must be located in directory specified by LoadModulePath. It is allowed to include multiple LoadModule parameters.
LoadModulePath	no			Full path to location of server modules. Default depends on compilation options.
LogFile	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.
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	500	No more than 'MaxHousekeeperDelete' rows (corresponding to [tablename], [field], [value]) will be deleted per one task in one housekeeping cycle. SQLite3 does not use this parameter, deletes all corresponding rows without a limit. If set to 0 then no limit is used at all. In this case you must know what you are doing! 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.
SenderFrequency	no	5-3600	30	How often Zabbix will try to send unsent alerts (in seconds).
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.
SourceIP	no			Source IP address for outgoing connections.
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.
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.

Parameter	Mandatory	Range	Default	Description
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 only and is supported since Zabbix 2.4.
StartDBSyncers	no	1-100	4	Number of pre-forked instances of DB Syncers. The upper limit used to be 64 before version 1.8.5. This parameter is supported since Zabbix 1.8.3.
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. The upper limit used to be 255 before version 1.8.5.
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.

Parameter	Mandatory	Range	Default	Description
StartProxyPollers	no	0-250	1	Number of pre-forked instances of pollers for passive proxies. 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.
StartTimers	no	1-1000	1	Number of pre-forked instances of timers. Timers process time-based trigger functions and maintenance periods. Only the first timer process handles the maintenance periods. This parameter is supported since Zabbix 2.2.0.
StartTrappers	no	0-1000	5	Number of pre-forked instances of trappers. 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.
Timeout	no	1-30	3	Specifies how long we wait for agent, SNMP device or external check (in seconds).
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.
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.

Parameter	Mandatory	Range	Default	Description
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
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

Note:
Zabbix supports configuration files only in UTF-8 encoding without BOM.

Comments starting with "#" are only supported in the beginning of the line.

2 Zabbix proxy

The parameters supported in a Zabbix proxy configuration file:

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-8G	8M	Size of configuration cache, in bytes. Shared memory size, for storing hosts and items data. Upper limit used to be 2GB before Zabbix 2.2.3.
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).
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. For SQLite3 path to database file must be provided. DBUser and DBPassword are ignored. 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 IBM DB2 and PostgreSQL.
DBSocket	no		3306	Path to MySQL socket. Database port when not using local socket. Ignored for SQLite.
DBUser				Database user. Ignored for SQLite.

Parameter	Mandatory	Range	Default	Description
DataSenderFrequency	no	1-3600	1	Proxy will send collected data to the server every N seconds. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).
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 debuging for web and VMware monitoring (can be set only using runtime control options)
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	8M	Size of history cache, in bytes. Shared memory size for storing history data.
HistoryTextCacheSize	no	128K-2G	16M	Size of text history cache, in bytes. Shared memory size for storing character, text or log history data.
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: 64

Parameter	Mandatory	Range	Default	Description
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.
HousekeepingFrequency	no	1-24	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.**

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.
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.
LoadModule	no			Module to load at proxy startup. Modules are used to extend functionality of the proxy. Format: LoadModule=<module.so> The modules must be located in directory specified by LoadModulePath. It is allowed to include multiple LoadModule parameters.
LoadModulePath	no			Full path to location of proxy modules. Default depends on compilation options.
LogFile	no			Name of log file. If not set, syslog is used.
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
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.
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			IP address (or hostname) of Zabbix server. Active proxy will get configuration data from the server. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).
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.
SourceIP	no			Source IP address for outgoing connections.

Parameter	Mandatory	Range	Default	Description
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 only and is supported since **Zabbix 2.4.0**.
StartDBSyncers	no	1-100	4	Number of pre-forked instances of DB Syncers. The upper limit used to be 64 before version 1.8.5. This parameter is supported since Zabbix 1.8.3.
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.

Parameter	Mandatory	Range	Default	Description
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.
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.
Timeout	no	1-30	3	Specifies how long we wait for agent, SNMP device or external check (in seconds).
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.

Parameter	Mandatory	Range	Default	Description
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
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

Note:
Zabbix supports configuration files only in UTF-8 encoding without BOM.

Comments starting with "#" are only supported in the beginning of the line.

3 Zabbix agent (UNIX)

The parameters supported in a Zabbix agent configuration file (*zabbix_agentd.conf*):

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. For example, to retrieve 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. Aliases can be used in *HostMetadataItem* but not in *HostnameItem* parameters.
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 (Currently, same as level 4. Can be set only using runtime control options.)
EnableRemoteCommands	no		0	Whether remote commands from Zabbix server are allowed. 0 - not allowed 1 - allowed

Parameter	Mandatory	Range	Default	Description
HostMetadata	no	0-255 characters		Optional parameter that defines host metadata. Host metadata is used only at host auto-registration 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 *EnableRemoteCommands* value. Host metadata is used only at host auto-registration process (active agent). During an auto-registration 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: 64
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 *EnableRemoteCommands* value. 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.
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. Format: LoadModule=<module.so> The modules must be located in directory specified by LoadModulePath. It is allowed to include multiple LoadModule parameters.
LoadModulePath	no			Full path to location of agent modules. Default depends on compilation options.
LogFile	no			Name of log file. If not set, syslog is used.
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

Parameter	Mandatory	Range	Default	Description
MaxLinesPerSecond	no	1-1000	100	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 4 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.
Server	no			List of comma delimited IP addresses (or hostnames) of Zabbix servers. Spaces are allowed since Zabbix 2.2. 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.
ServerActive	no			IP:port (or hostname:port) of Zabbix server or Zabbix proxy for active checks. Multiple comma-delimited addresses can be provided to use several independent Zabbix servers in parallel. Spaces are allowed since Zabbix 2.2. 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.
SourceIP	no			Source IP address for outgoing connections.

Parameter	Mandatory	Range	Default	Description
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
UnsafeUserParameters	no	0,1	0	Allow all characters to be passed in arguments to user-defined parameters. 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

Note:
In Zabbix agent 2.0.0 version configuration parameters related to active and passive checks have been changed.
See the "See also" section at the bottom of this page to read more details about these changes.

Note:
Zabbix supports configuration files only in UTF-8 encoding without BOM.

Comments starting with "#" are only supported in the beginning of the line.

See also

1. Differences in the Zabbix agent configuration for active and passive checks starting from version 2.0.0

4 Zabbix agent (Windows)

The parameters supported in a Zabbix agent (Windows) configuration file:

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. For example, to retrieve 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. Aliases can be used in *HostMetadataItem* but not in *HostnameItem* or *PerfCounter* parameters.
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 (Currently, same as level 4. Can be set only using runtime control options.)
EnableRemoteCommands	no		0	Whether remote commands from Zabbix server are allowed. 0 - not allowed 1 - allowed
HostMetadata	no	0-255 characters		Optional parameter that defines host metadata. Host metadata is used only at host auto-registration 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. Host metadata is used only at host auto-registration process (active agent). During an auto-registration 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: 64
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. 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.
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	no			Name of log file. If not set, Windows Event Log is used.
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
MaxLinesPerSecond	no	1-1000	100	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.

Parameter	Mandatory	Range	Default	Description
PerfCounter	no			Syntax: <parameter_name>,”<perf_counter_path>”,<period> Defines new parameter <parameter_name> which is an average value for system performance counter <perf_counter_path> for the specified time period <period> (in seconds). For example, if you wish to receive average number of processor interrupts per second for last minute, you can define new parameter ”interrupts” as 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.
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.
Server	no			List of comma delimited IP addresses (or hostnames) of Zabbix servers. Spaces are allowed since Zabbix 2.2. 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.

Parameter	Mandatory	Range	Default	Description
ServerActive	no	(*)		IP:port (or hostname:port) of Zabbix server or Zabbix proxy for active checks. Multiple comma-delimited addresses can be provided to use several independent Zabbix servers in parallel. Spaces are allowed since Zabbix 2.2. 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.
SourceIP	no			Source IP address for outgoing connections.
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
UnsafeUserParameters	no	0-1	0	Allow all characters to be passed in arguments to user-defined parameters. 0 - do not allow 1 - allow
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.

Note:
In Zabbix agent 2.0.0 version configuration parameters related to active and passive checks have been changed.
See the "See also" section at the bottom of this page to read more details about these changes.

Note:
Zabbix supports configuration files only in UTF-8 encoding without BOM.

Comments starting with "#" are only supported in the beginning of the line.

See also

1. Differences in the Zabbix agent configuration for active and passive checks starting from version 2.0.0.

5 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 file `settings.sh`. 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.
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. This parameter is supported since Zabbix 2.0.15, 2.2.10 and 2.4.5.

Warning:
Port 10052 is not IANA registered.

6 Archive: Zabbix agent (UNIX, Inetd version)

The parameters supported in a Zabbix agent (UNIX, Inetd version) configuration file:

Parameter	Mandatory	Default value	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. For example, to retrieve 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.
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.
Server	yes	-	Comma-delimited list of IP addresses of ZABBIX Servers or Proxies. Connections from other IP addresses will be rejected.
Timeout	no	3	Do not spend more than Timeout seconds on getting requested value (1-255). The agent does not kill timeouted User Parameters processes!
UnsafeUserParameters	no	0	Allow all characters to be passed in arguments to user-defined parameters
UserParameter	no		User-defined parameter to monitor. There can be several user-defined parameters. Format: UserParame-ter=<key>,<shell command> Note that shell command must not return empty string or EOL only. Example: UserParame-ter=system.test,who|wc -l

7 Special notes on "Include" parameter

If an `Include` parameter is used for including a file, the file must be readable.

If an Include parameter is used for including a directory:

```
- All files in the directory must be readable.
- No particular order of inclusion should be assumed (e.g. files are not included in alphabetical order)
- All files in the directory are included into configuration.
- Beware of file backup copies automatically created by some text editors. For example, if editing the '
```

If an Include parameter is used for including files using a pattern:

```
- All files matching the pattern must be readable.
- No particular order of inclusion should be assumed (e.g. files are not included in alphabetical order)
```

4 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 Parameter / sys- tem	▼▼	▼▼	▼▼	▼▼	▼▼	▼▼	▼▼	▼▼	▼▼	▼▼	▼▼
▼▼	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.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>]	X	X	X	X	X	X	X	X	X	X	X
logrt[file_format,<regexp>,<encoding>,<maxlines>,<mode>,<output>]	X	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 *(default)*	X	X	X	X	X [2]	X	X	-	X	X	r

	packets	X	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X [2]	X	X	-	X	X	r
	dropped	X	X	X	X	-	X	-	-	X	X	r
net.if.out[if,<mode>]		X	X	X	X	X	X [1]	X	-	X	X	r
mode	bytes	X	X	X	X	X [2]	X	X	-	X	X	r
▲	*(default)*											
	packets	X	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X [2]	X	X	-	X	X	r
	dropped	X	X	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 [2]	X	X	-	X	X	r
▲	*(default)*											
	packets	X	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X [2]	X	X	-	X	X	r
	dropped	X	X	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	-	-
		1	2	3	4	5	6	7	8	9	10	11
proc.mem[<name>,<user>,<mode>,<cmdline>]		X	X	X	X	X [3]	-	X	X	-	X	X
mode	sum	-	X	X	X	X	-	X	X	-	X	X
▲	*(default)*											
	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
proc.num[<name>,<user>,<state>,<cmdline>]		X	X	X	X	X [3]	X	X	X	-	X	X
state	all	-	X	X	X	X	X	X	X	-	X	X
▲	*(default)*											
	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
cmdline		-	X	X	X	X	X	X	X	-	X	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	X	X	X	X	X	X	X	X	X	X	X
	(default)											
	percpu	X	X	X	X	X	X	X	-	X	X	X
mode	avg1	X	X	X	X	X	X	X	X	X	X	X
▲	*(default)*											
	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	X	X	X	X	X	X	X	-	X	X	X
▲	*(default)*											
	max	-	X	X	X	X	-	-	-	X	-	-
system.cpu.switches		-	X	X	X	X	-	X	-	-	X	X
system.cpu.util[<cpu>,<type>,<mode>]		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	X	X	X	X	X	X	X	X	-	X	X
	iowait	-	-	X	-	X	-	X	-	-	-	-
	interrupt	-	-	X	X	-	-	-	-	-	X	-
	softirq	-	-	X	-	-	-	-	-	-	-	-
	steal	-	-	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
		1	2	3	4	5	6	7	8	9	10	11
system.hostname[<type>]		X	X	X	X	X	X	X	X	X	X	X
system.hw.chassis[<info>]		X	X	X	-	-	-	-	-	-	-	-
system.hw.cpu[<cpu>,<info>]		X	X	X	-	-	-	-	-	-	-	-
system.hw.devices[<type>]		X	X	X	-	-	-	-	-	-	-	-
system.hw.macaddr[<interface>,<format>]		X	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	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	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
		1	2	3	4	5	6	7	8	9	10	11
vfs.dev.read[<device>,<type>,<mode>]		X	X	X	X	X	-	X	-	-	X	-

type ▲	sectors	-	X	X	-	-	-	-	-	-	-	-
(defaults are different under various OSes)												
	operations	-	X	X	X	X	-	X	-	-	X	-
	bytes	-	-	-	X	X	-	X	-	-	X	-
	sps	-	X	X	-	-	-	-	-	-	-	-
	ops	-	X	X	X	-	-	-	-	-	-	-
	bps	-	-	-	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	-
type ▲	sectors	-	X	X	-	-	-	-	-	-	-	-
(defaults are different under various OSes)												
	operations	-	X	X	X	X	-	X	-	-	X	-
	bytes	-	-	-	X	X	-	X	-	-	X	-
	sps	-	X	X	-	-	-	-	-	-	-	-
	ops	-	X	X	X	-	-	-	-	-	-	-
	bps	-	-	-	X	-	-	-	-	-	-	-
mode ▲ *(compatible only with type in: sps, ops, bps)*	avg1 *(default)*	-	X	X	X	-	-	-	-	-	-	-
	avg5	-	X	X	X	-	-	-	-	-	-	-
	avg15	-	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]		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 ▲	modify *(default)*	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	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.inode[fs,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode ▲	total *(default)*	-	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 ▲	total *(default)*	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 ▲	total *(default)*	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] Items net.if.in, net.if.out and net.if.total 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 utilisation 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.

2 vm.memory.size parameters

- **total** - total physical memory available.
- **free** - memory that is readily available to any entity requesting memory.
- **active** - memory currently in use or very recently used, and so it is in RAM.
- **inactive** - memory that is marked as not used.
- **wired** - memory that is marked to always stay in RAM. It is never moved to disk.
- **pinned** - same as 'wired'.
- **anon** - memory not associated with a file (cannot be re-read from it).
- **exec** - executable code, typically from a (program) file.
- **file** - cache for contents of recently accessed files.
- **buffers** - cache for things like file system metadata.
- **cached** - cache for various things.
- **shared** - memory that may be simultaneously accessed by multiple processes.
- **used** - active + wired memory.
- **pused** - active + wired memory in relation to total.
- **available** - inactive + cached + free memory.
- **pavailable** - inactive + cached + free memory in relation to 'total'.

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, item *vm.memory.size[used]* is designed for informational purposes only, while item *vm.memory.size[available]* is designed to be used in triggers.

Note:
See the "See also" section at the bottom of this page to find more detailed information about memory calculation in different OS.

Platform-specific notes

- on AIX and Solaris **available** and **free** are the same
- on Linux **shared** works only on kernel 2.4
- on Linux **available** is free + buffers + cached
- on Linux **pavailable** is free + buffers + cached in relation to total

See also

1. Detailed information 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.

There are some definitions used in the details of protocols used by Zabbix:

```
<HEADER> - "ZBXD\x01" (5 bytes)
<DATALEN> - data length (8 bytes). 1 will be formatted as 01/00/00/00/00/00/00/00 (eight bytes in HEX, 64
```

To not exhaust memory (potentially) Zabbix server is limited to accept only 128MB in one connection when using the Zabbix protocol.

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

```
<item key>\n
```

Agent response

```
<HEADER><DATALEN><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 **agent.ping\n**
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 **vfs.fs.size[/nono]\n**
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).

Getting the list of items

Agent request

```
<HEADER><DATALEN>{
    "request":"active checks",
    "host":"<hostname>"
}
```

Server response

```
<HEADER><DATALEN>{
    "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.

Sending in collected data

Agent sends

```
<HEADER><DATALEN>{
    "request":"agent data",
    "data":[
        {
            "host":"<hostname>",
            "key":"agent.version",
            "value":"2.4.0",
            "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
            "clock":1400675595,
            "ns":77053975
        },
        {
            "host":"<hostname>",
            "key":"vfs.fs.size[/nono]",
            "state":1,
            "value":"Cannot obtain filesystem information: [2] No such file or directory",
            "clock":1400675595,
            "ns":78154128
        }
    ],
    "clock": 1400675595,
    "ns": 78211329
}
```

Server response

```
<HEADER><DATALEN>{
    "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.

See also

1. More details on Zabbix agent protocol

4 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

5 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.

6 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.

Since Zabbix 2.4.7, if another item check was successful between two failed checks of a problematic item, the problematic item is marked as not supported after the second failed check without affecting host availability. This was removed in Zabbix 2.4.8.

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. 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.

If the unreachable host reappears, the monitoring returns to normal automatically:

```
resuming Zabbix agent checks on host "New host": connection restored
```

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 goes from green 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
```

7 Sensor

Each sensor chip gets its own directory in the sysfs /sys/devices tree. To find all sensor chips, it is easier to follow the device symlinks from /sys/class/hwmon/hwmon*, where * is a real number (0,1,2,...).

The sensor readings are located either in /sys/class/hwmon/hwmon*/ directory for virtual devices, or in /sys/class/hwmon/hwmon*/device directory for non-virtual devices. A file, called name, located inside hwmon* or hwmon*/device directories contains the name of the chip, which corresponds to the name of the kernel driver used by the sensor chip.

There is only one sensor reading value per file. The common scheme for naming the files that contain sensor readings inside any of the directories mentioned above is: <type><number>_<item>, where

- **type** - for sensor chips is ”in” (voltage), ”temp” (temperature), ”fan” (fan), etc.,
- **item** - ”input” (measured value), ”max” (high threshold), ”min” (low threshold), etc.,
- **number** - always used for elements that can be present more than once (usually starts from 1, except for voltages which start from 0). If files do not refer to a specific element they have a simple name with no number.

The information regarding sensors available on the host can be acquired using **sensor-detect** and **sensors** tools (lm-sensors package: http://lm-sensors.org/). **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*.[1]

```
$ zabbix_get -s 127.0.0.1 -k sensor[lm85-i2c-0-2e,in1]
1.301000
```

Not only voltage (in), but also current (curr), temperature (temp) and fan speed (fan) readings can be retrieved by Zabbix.

8 Implementation details of net.tcp.service checks

Overview

Implementation of net.tcp.service checks is detailed in this section for various services specified in the service parameter.

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.

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.

pop

Creates a TCP connection and expects the first 3 characters of the response to be "+OK", then sends "QUIT\r\n". Default port 110 is used if not specified.

smtp

Creates a TCP connection and expects the first 3 characters of the response to be "220", followed by a space, the line ending or a dash. The lines containing a dash belong to a multi-line response and the response will be re-read until a line without the dash is received. Then sends "QUIT\r\n". Default port 25 is used if not specified.

ssh

Creates a TCP connection. If the connection has been established, both sides exchange an identification string (SSH-major.minor-XXXX), where major and minor are protocol versions and XXXX is a string. Zabbix checks if the string matching the specification is found and then sends back the string "SSH-major.minor-zabbix_agent\r\n" or "0\n" on mismatch. Default port 22 is used if not specified.

tcp

Creates a TCP connection without expecting and sending anything. Unlike the other checks requires the port parameter to be specified.

[1] According to specification these are voltages on chip pins and generally speaking may need scaling.

telnet

Creates a TCP connection and expects a login prompt (':' at the end). Default port 23 is used if not specified.

5 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 last and previous values.		Supported value types: float, int, str, text, log For example: (previous value;last 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(3600) → average value for an hour => avg(3600,86400) → 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) or **#num** - the Nth most recent value **mask** (mandatory) - 64-bit unsigned integer (0 - 18446744073709551615) **time_shift** (optional) - see avg()	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 last and previous values.	Supported value types: float, int, str, text, log For example: (previous value;last value=change) 1;5=+4 3;1=-2 0;-2.5=-2.5 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 (supported since Zabbix 2.2.0). Note that: *eq* (default), *ne*, *gt*, *ge*, *lt*, *le*, *band* are supported for integer items *eq* (default), *ne*, *gt*, *ge*, *lt*, *le* are supported for float	Supported value types: float, integer, string, text, log Float items match with the precision of 0.000001. With *band* as third parameter, the second parameter can be specified as two numbers, separated by '/': **number_to_compare_with/mask** count() calculates "bitwise AND" from the value and the *mask* and compares the result to *number_to_compare_with*. If the result of "bitwise AND" is equal to *number_to_compare_with*, the value is counted. If *number_to_compare_with* and *mask* are equal, only the *mask* need be specified (without '/'). Examples: => count(600)

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) - see avg()	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

fuzzytime (sec)

FUNCTION			
	Checking how much an item timestamp value differs from the Zabbix server time.	**sec** - seconds	Supported value types: float, int Returns: 0 - if difference between item timestamp value and Zabbix server timestamp is over T seconds 1 - otherwise. Usually used with system.localtime to check that local time is in sync with local time of Zabbix server. Can be used also with vfs.file.time[/path/file,mo key to check that file didn't get updates for long time. Example: => fuzzytime(60)=0 → detect a problem if time difference is over 60 seconds
iregexp (pattern,<sec|#num>)			

FUNCTION			
	This function is a non case-sensitive analogue of regexp().	see regexp()	Supported value types: str, log, text

last (sec|#num,<time_shift>)

FUNCTION				
		The most recent value.	**sec** (ignored) or **#num** - the Nth most recent value **time_shift** (optional) - see avg()	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			
	Check if event ID of the last log entry matches a regular expression.	**pattern** - regular expression describing the required pattern, POSIX extended 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 parameter.	**pattern** - required string	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) - see avg()	Supported value types: float, int The `time_shift` parameter is supported since Zabbix 1.8.2.
min (sec|#num,<time_shift>)			
	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) - see avg()	Supported value types: float, int The `time_shift` parameter is supported since Zabbix 1.8.2.
nodata (sec)			

FUNCTION				
		Checking for no data received.	**sec** - evaluation period in seconds. The period should not be less than 30 seconds.	Supported value types: *any* Returns: 1 - if no data received during the defined period of time 0 - otherwise Note that this function will display an error if, within the period of the 1st parameter: - there's no data and Zabbix server was restarted - there's no data and maintenance was completed - there's no data and the item was added or re-enabled Errors are displayed in the *Info* column in trigger configuration.
now				

FUNCTION			
	Number of seconds since the Epoch (00:00:00 UTC, January 1, 1970).		Supported value types: *any*
prev	Previous value.		Supported value types: float, int, str, text, log Returns the same as last(#2).
regexp (pattern,<sec|#num>)	Checking if the latest (most recent) value matches regular expression.	**pattern** - regular expression, POSIX extended 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 This function is case-sensitive.
str (pattern,<sec|#num>)			

FUNCTION			
	Finding a string in the latest (most recent) value.	**pattern** - 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 This function is case-sensitive.
strlen (sec|#num,<time_shift>)			

FUNCTION			
	Length of the latest (most recent) value in characters (not bytes).	**sec** (ignored) or **#num** - the Nth most recent value **time_shift** (optional) - see avg()	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(,86400) → 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) - see avg()	Supported value types: float, int The function is evaluated starting with the first received value. 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

Warning:
Important notes:
1) All functions return numeric values only. Comparison to strings is not supported.
2) Some of the functions cannot be used for non-numeric values!
3) String arguments should be double quoted. Otherwise, they might get misinterpreted.
4) 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 function is evaluated starting with the first received value (unless the `timeshift` parameter is used).

6 Macros

1 Macros supported by location

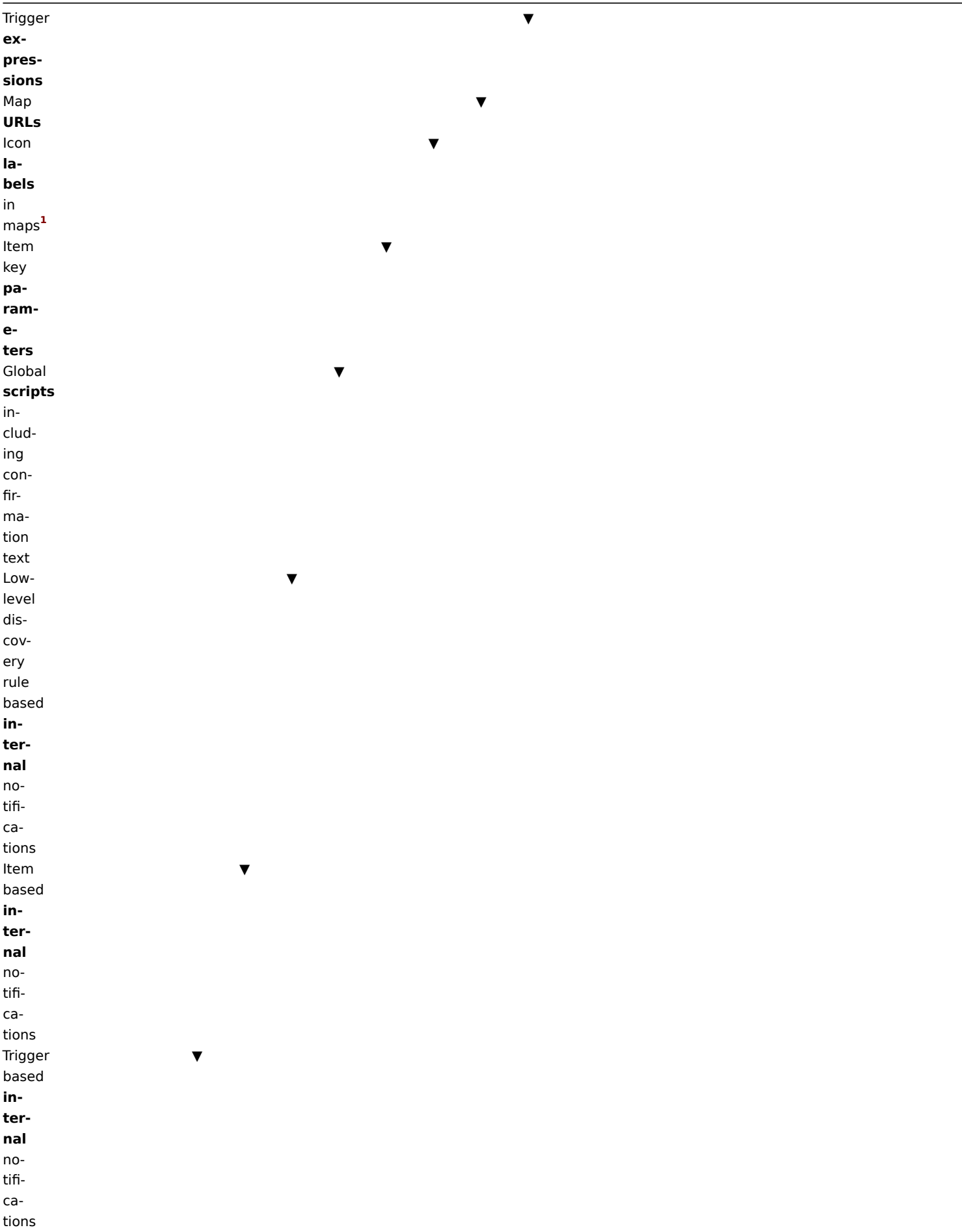
Overview

The table contains a complete list of macros supported by Zabbix.

- **X** means "supported" in that location
- The numbered macro syntax of {MACRO**<1-9>**} is used to reference hosts in the order in which they appear in a trigger expression. Thus, macros like {HOST.IP1}, {HOST.IP2}, {HOST.IP3} will expand to the IP of the first, second and third host in the trigger expression, providing the expression contains those hosts. Additionally {HOST.HOST**<1-9>**} is supported within

{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.

| | | |


Trigger expressions
Map URLs
Icon labels in maps[1]
Item key parameters
Global scripts including confirmation text
Low-level discovery rule based internal notifications
Item based internal notifications
Trigger based internal notifications

Auto registration **notifications**	▼																		
Discovery **notifications**	▼																		
Trigger-based **notifications** and commands	▼																		
▼MACRO ▼	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
{ACTION.ID}	X	X	X	X	X														*Numeric ID of the trigger action*
{ACTION.NAME}	X	X	X	X	X														*Name of the trigger action*
{DATE}	X	X	X	X	X														*Current date in yyyy.mm.dd format*

{DISCOVERY.DEVICE.IPADDRESS}	*IP ad dr of th di co er de vi* Av al wa do no de pe or ho be in ad
{DISCOVERY.DEVICE.DNS}	*DI na of th di co er de vi* Av al wa do no de pe or ho be in ad
{DISCOVERY.DEVICE.STATUS}	*St of th di co er de vi* ca be ei th UI or DC

{DISCOVERY.DEVICE.UPTIME}	*Ti si th la ch of di co er st tu fo a pa tio ul de vi* Fo ex ar pl 1h 29 Fo de vi wi st tu DC th is th pe ri of th do ti

{DISCOVERY.RULE.NAME}	*Na of th di co er ru th di co er th pr er or ab se of th de vi or se vi*
{DISCOVERY.SERVICE.NAME}	*Na of th se vi th wa di co er* Fo ex ar pl HT
{DISCOVERY.SERVICE.PORT}	*Po of th se vi th wa di co er* Fo ex ar pl 80

{DISCOVERY.SERVICE.STATUS}

{EVENT.X.ACK.HISTORY}						*Lo of ac kn ec m or th pr le*
{EVENT.X.ACK.STATUS}						*Ac st tu of th ev (Y*
{EVENT.X.AGE}	X	X	X	X	X	*Ag of th ev th tri ge ar ac tio* Us in es ca la m sa
{EVENT.X.DATE}	X	X	X	X	X	*Da of th ev th tri ge ar ac tio*
{EVENT.X.ID}	X	X	X	X	X	*Nu ID of th ev th tri ge ar ac tio*

| | | |

{EVENT.RECOVERY.STATUS}	X	X	X	*Ve va of th re co er ev* Ca be us in re co er m sa or Su po si 2.
{EVENT.RECOVERY.TIME}	X	X	X	*Ti of th re co er ev* Ca be us in re co er m sa or Su si 2.

{EVENT.RECOVERY.VALUE}	X	X	X		*Numeric value of the recovery event. Can be used in recovery messages or Supported since 2.*
{EVENT.STATUS}	X	X	X	X	*Verbose value of the event that triggered an action. Supported since 2.*
{EVENT.TIME}	X	X	X	X	*Time of the event that triggered an action. Supported since 2.*
{EVENT.VALUE}	X	X	X	X	*Numeric value of the event that triggered an action. Supported since 2.*

{HOST.CONN<1-9>}		X	X	X	X	X[2]	X		X		X	X[5]	X		X[8]	X[8]	*IP or ho DN na de pe in or ho se ti* [3]. Su in tri ge na si 2.
{HOST.DESCRIPTION<1-9>}		X	X	X			X										*Ho de sc tio* Su si 2.
{HOST.DNS<1-9>}		X	X	X	X	X[2]	X		X		X	X[5]	X		X[8]	X[8]	*Ho DN na* [3]. Su in tri ge na si 2.
{HOST.HOST<1-9>}	X	X	X	X	X	X	X		X		X	X[5]	X		X[8]	X[8]	*Ho na* {H is de re ca
{HOST.ID}								X								X[8]	Ho ID
{HOST.IP<1-9>}	X	X	X	X	X	X[2]	X		X		X	X[5]	X		X[8]	X[8]	*Ho IP ad dr* [3]. Su po si 2. {I is de re ca

{HOST.METADATA}	X														*Ho m da* Us or fo ac tiv ag au re Su po si 2.
{HOST.NAME<1-9>}	X	X	X	X	X	X		X		X	X[5]	X		X[8]	X[8]	*Vi ho na* Su si 2.
{HOST.PORT<1-9>}	X	X	X	X				X								*Ho (a po* [3]. Su in au re si 2. Su in tri ge na tri ge de sc tio in te na ar tri ba no tif ca tio si 2.
{HOSTGROUP.ID}							X									*Ho gr ID*

{INVENTORY.ALIAS<1-9>}	X	X	X		*Al fie in ho in ve to*
{INVENTORY.ASSET.TAG<1-9>}	X	X	X		*As ta fie in ho in ve to*
{INVENTORY.CHASSIS<1-9>}	X	X	X		*Ch fie in ho in ve to*
{INVENTORY.CONTACT<1-9>}	X	X	X		*Co fie in ho in ve to {I is de re ca*

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

{INVENTORY.CONTRACT.NUMBER<1-9>}	X	X	X		*Co nu be fie in ho in ve to*
{INVENTORY.DEPLOYMENT.STATUS<1-9>}	X	X	X		*De st tu fie in ho in ve to*

{INVENTORY.HARDWARE<1-9>}	X	X	X	*Ha fie in ho in ve to* {I is de re ca
{INVENTORY.HARDWARE.FULL<1-9>}	X	X	X	*Ha (F de ta fie in ho in ve to*
{INVENTORY.HOST.NETMASK<1-9>}	X	X	X	*Ho su ne m fie in ho in ve to*
{INVENTORY.HOST.NETWORKS<1-9>}	X	X	X	*Ho ne wo fie in ho in ve to*
{INVENTORY.HOST.ROUTER<1-9>}	X	X	X	*Ho ro fie in ho in ve to*
{INVENTORY.HW.ARCH<1-9>}	X	X	X	*Ha ar ch te tu fie in ho in ve to*

{INVENTORY.HW.DATE.DECOMM<1-9>}	X	X	*Date when hardware was decommissioned*
{INVENTORY.HW.DATE.EXPIRY<1-9>}	X	X	*Date when maintenance expiry field in hardware inventory*
{INVENTORY.HW.DATE.INSTALL<1-9>}	X	X	*Date when installation field in hardware inventory*
{INVENTORY.HW.DATE.PURCHASE<1-9>}	X	X	*Date when purchase field in hardware inventory*
{INVENTORY.INSTALLER.NAME<1-9>}	X	X	*Installer name field in hardware inventory*

{INVENTORY.LOCATION<1-9>}	X	X	X	*Lo fie in ho in ve to* {I is de re ca
{INVENTORY.LOCATION.LAT<1-9>}	X	X	X	*Lo la tu fie in ho in ve to*
{INVENTORY.LOCATION.LON<1-9>}	X	X	X	*Lo lo gi tu fie in ho in ve to*
{INVENTORY.MACADDRESS.A<1-9>}	X	X	X	*MA ad dr A fie in ho in ve to* {I is de re ca
{INVENTORY.MACADDRESS.B<1-9>}	X	X	X	*MA ad dr B fie in ho in ve to*

{INVENTORY.MODEL<1-9>}				X		X		X												*Mo fie in ho in ve to*
{INVENTORY.NAME<1-9>}				X		X		X												*Na fie in ho in ve to {I is de re ca*
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19		
{INVENTORY.NOTES<1-9>}				X		X		X												*No fie in ho in ve to {I is de re ca*
{INVENTORY.OOB.IP<1-9>}				X		X		X												*OO IP ad dr fie in ho in ve to*
{INVENTORY.OOB.NETMASK<1-9>}				X		X														*OO su ne m fie in ho in ve to*
{INVENTORY.OOB.ROUTER<1-9>}				X		X														*OO ro fie in ho in ve to*

{INVENTORY.OS<1-9>}	X	X	X	*OS fie in ho in ve to {I is de re ca*
{INVENTORY.OS.FULL<1-9>}	X	X	X	*OS (F de ta fie in ho in ve to*
{INVENTORY.OS.SHORT<1-9>}	X	X	X	*OS (S fie in ho in ve to*
{INVENTORY.POC.PRIMARY.CELL<1-9>}	X	X	X	*Pr PO ce fie in ho in ve to*
{INVENTORY.POC.PRIMARY.EMAIL<1-9>}	X	X	X	*Pr PO er fie in ho in ve to*
{INVENTORY.POC.PRIMARY.NAME<1-9>}	X	X	X	*Pr PO na fie in ho in ve to*

{INVENTORY.POC.PRIMARY.NOTES<1-9>}	X	*Primary POC notes field in host inventory*
{INVENTORY.POC.PRIMARY.PHONE.A<1-9>}	X	*Primary POC phone A field in host inventory*
{INVENTORY.POC.PRIMARY.PHONE.B<1-9>}	X	*Primary POC phone B field in host inventory*
{INVENTORY.POC.PRIMARY.SCREEN<1-9>}	X	*Primary POC screen name field in host inventory*
{INVENTORY.POC.SECONDARY.CELL<1-9>}	X	*Secondary POC cell field in host inventory*
{INVENTORY.POC.SECONDARY.EMAIL<1-9>}	X	*Secondary POC email field in host inventory*

```
{INVENTORY.POC.SECONDARY.NAME<1-   X                                                
```

{INVENTORY.SERIALNO.A<1-9>}	X	X	X	*Se nu be A fie in ho in ve to* {I is de re ca
{INVENTORY.SERIALNO.B<1-9>}	X	X	X	*Se nu be B fie in ho in ve to*
{INVENTORY.SITE.ADDRESS.A<1-9>}	X	X	X	*Si ad dr A fie in ho in ve to*
{INVENTORY.SITE.ADDRESS.B<1-9>}	X	X	X	*Si ad dr B fie in ho in ve to*
{INVENTORY.SITE.ADDRESS.C<1-9>}	X	X	X	*Si ad dr C fie in ho in ve to*

{INVENTORY.SITE.CITY<1-9>}	X	X	X	Si ci fie in ho in ve to
{INVENTORY.SITE.COUNTRY<1-9>}	X	X	X	Si co try fie in ho in ve to
{INVENTORY.SITE.NOTES<1-9>}	X	X	X	Si no fie in ho in ve to
{INVENTORY.SITE.RACK<1-9>}	X	X	X	Si ra lo ca tio fie in ho in ve to
{INVENTORY.SITE.STATE<1-9>}	X	X	X	Si st fie in ho in ve to
{INVENTORY.SITE.ZIP<1-9>}	X	X	X	Si ZI fie in ho in ve to

{INVENTORY.SOFTWARE<1-9>}	X	X	X	*Sc fie in ho in ve to* {I is de re ca
{INVENTORY.SOFTWARE.APPX<1-9>}	X	X	X	*Sc ap pl ca tic A fie in ho in ve to*
{INVENTORY.SOFTWARE.APPX<1-9>}	X	X	X	*Sc ap pl ca tic B fie in ho in ve to*
{INVENTORY.SOFTWARE.APPX<1-9>}	X	X	X	*Sc ap pl ca tic C fie in ho in ve to*
{INVENTORY.SOFTWARE.APPX<1-9>}	X	X	X	*Sc ap pl ca tic D fie in ho in ve to*

{INVENTORY.SOFTWARE.APPLE<1-9>}	X	X	X																*Sc ap pl ca tio E fie in ho in ve to*
{INVENTORY.SOFTWARE.FULL<1-9>}	X	X	X																*Sc (F de ta fie in ho in ve to*
{INVENTORY.TAG<1-9>}	X	X	X																*Ta fie in ho in ve to* {I is de re ca
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
{INVENTORY.TYPE<1-9>}	X	X	X																*Ty fie in ho in ve to* {I is de re ca
{INVENTORY.TYPE.FULL<1-9>}	X	X	X																*Ty (F de ta fie in ho in ve to*

{INVENTORY.URL.A<1-9>}	X	X	X	*UR A fie in ho in ve to*
{INVENTORY.URL.B<1-9>}	X	X	X	*UR B fie in ho in ve to*
{INVENTORY.URL.C<1-9>}	X	X	X	*UR C fie in ho in ve to*
{INVENTORY.VENDOR<1-9>}	X	X	X	*Ve fie in ho in ve to*
{ITEM.DESCRIPTION<1-9>}	X	X	X	*De of th Nt ite in th tr ge ex pr si th ca a no tif ca tio* Su po si 2.

{ITEM.ID<1-9>}	X	X	X	*Nu ID of th Nt ite in th tr ge ex pr si th ca a no tif ca tio* Su po si 1.
{ITEM.KEY<1-9>}	X	X	X	*Ke of th Nt ite in th tr ge ex pr si th ca a no tif ca tio* Su po si 2. {T is de re ca

{ITEM.KEY.ORIG<1-9>}	X	X	X		*Or ke (w m no ex pa of th Nt ite in th tri ge ex pr si th ca a no tif ca tio* Su po si 2.
{ITEM.LASTVALUE<1-9>}				X	*Th la es va of th Nt ite in th tri ge ex pr si th ca a no tif ca tio* Su po si 1.

It al to {+

{ITEM.LOG.AGE<1-9>}	*Ag of th lo ite ev*
{ITEM.LOG.DATE<1-9>}	*Da of th lo ite ev*
{ITEM.LOG.EVENTID<1-9>}	*ID of th ev in th ev lo* Fo W do ev lo m ito in or
{ITEM.LOG.NSEVERITY<1-9>}	*Nu se ity of th ev in th ev lo* Fo W do ev lo m ito in or

{ITEM.LOG.SEVERITY<1-9>}				*Ve se ity of th ev in th ev lo* Fo W do ev lo m ito in or
{ITEM.LOG.SOURCE<1-9>}				*So of th ev in th ev lo* Fo W do ev lo m ito in or
{ITEM.LOG.TIME<1-9>}				*Ti of th lo ite ev*
{ITEM.NAME<1-9>}	X	X	X	*Na of th Nt ite in th tri ge ex pr si th ca a no tif ca tio*

{ITEM.NAME.ORIG<1-9>}	X			X	X	X													*Or na (w m no ex pa of th Nt ite in th tri ge ex pr si th ca a no tif ca tio* Su po si 2.
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

{ITEM.STATE<1-9>}	X	*Th la es st of th Nt ite in th tr ge ex pr si th ca a no tif ca tic* Po si- bl va ue **N su po** ar **N m** Su si 2.

{ITEM.VALUE<1-9>}	X	Re to ei th 1) th la es va of th Nt ite in th tri ge ex pr si if us fo di pl in tri ge In th ca wo th sa as {IT 2) th hi to ca (a th ti of ev va of th Nt ite in th tri ge ex pr si if us fo di pl in ev ar

{LLDRULE.DESCRIPTION}	X	*De of th lo le di co er ru wh ca a no tif ca tio* Su si 2.
{LLDRULE.ID}	X	*Nu ID of th lo le di co er ru wh ca a no tif ca tio* Su si 2.
{LLDRULE.KEY}	X	*Ke of th lo le di co er ru wh ca a no tif ca tio* Su si 2.

{LLDRULE.KEY.ORIG}	X	*Or ke (w m no ex pa of th lo le di co er ru wh ca a no tif ca tio* Su si 2.
{LLDRULE.NAME}	X	*Na of th lo le di co er ru wh ca a no tif ca tio* Su si 2.

{LLDRULE.NAME.ORIG}	X		*Or na (w m no ex pa of th lo le di co er ru wh ca a no tif ca tio* Su si 2.
{LLDRULE.STATE}	X		*Th la es st of th lo le di co er ru* Po si- bl va ue **No su po** an **No m** Su si 2.
{MAP.ID}		X	*Ne m ID*

{PROXY	X	DESCRIPTION	X	<1-9>}	X	X	X
{PROXY	X	NAME	X	<1-9>}	X	X	X
{TIME}	X	X	X	X	X	X	X

{TRIGGER.DESCRIPTION}	X	*Tr* *de* *sc* *tio* Su po si 2. St wi 2. al m su po in tri ge de sc tio wi be ex pa if {T is us in no tif ca tio te {T is de re ca

{TRIGGER.EVENTS.ACK}	X		*Nu of ac kr ec ev fo a m el m in m or fo th tr ge wh ge er at cu re ev in nc tif ca tic* Su po si 1.
{TRIGGER.EVENTS.PROBLEM.ACK}	X		*Nu of ac kr ec PR LE ev fo al tr ge di re ga in th st* Su po si 1.

{TRIGGER.EVENTS.PROBLEM.UNACK}	X	X	*Nu of ur ac kr ec PR LE ev fo al tri ge di re ga in th st* Su po si 1.

{TRIGGER.EVENTS.UNACK}	X	*Nu of ur ac kr ec ev fo a m el m in m or fo th tr ge wh ge er at cu re ev in no tif ca tio* Su po in m el m la be si 1.

{TRIGGER.HOSTGROUP.NAME<X>}	*A sc (b SC qu co sp se a- ra lis of ho gr in wh th tri ge is de fir* Su po si 2.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
{TRIGGER.PROBLEM.EVENTS.PROBLEM.ACK}								X											*Nu of ac kr ec PP LE ev fo tri ge in PP LE st* Su po si 1.

{TRIGGER.PROBLEM.EVENTS.PROBLEM.UNACK}		X		*Nu of un ac kn ec PR LE ev fo tri ge in PR LE st* Su po si 1.
{TRIGGER.EXPRESSION}	X			*Tr ex pr si* Su po si 1.
{TRIGGER.ID}	X		X	*Nu tri ge ID wh tri ge th ac tio* Su in tri ge UR si Za bi 1.
{TRIGGER.NAME}	X			*Na of th tri ge*

{TRIGGER.NAME.ORIG}	X	*Original name (without macro expansion) of the trigger*
{TRIGGER.NSEVERITY}	X	*Numeric trigger severity* Possible value 0 Not classified 1 Information 2 Warning 3 Average 4 High 5 Disaster

Supported since 2.

{TRIGGER.SEVERITY}	X	*Trigger severity name* Can be defined in *Administration → General → Trigger severities*
{TRIGGER.STATE}	X	*The latest evaluation state of the trigger* Possible values: **Unknown** and **Normal**
{TRIGGER.STATUS}	X	*Current trigger status value* Can be either PROBLEM or OK

{TRIGGER.TEMPLATE.NAME}	X		*A sc (b SC qu co sp se a- ra lis of te pl in wh th tri ge is de fir or *U KM if th tri ge is de fir in ho* Su po si
{TRIGGER.URL}	X		2. *Tr UR Cu tri ge nu m va*
{TRIGGER.VALUE}		X	0 tri ge is in OR st 1 tri ge is PR LE st

{TRIGGERS.UNACK}	X	*Nu of ur ac kr ec tri ge fo a m el m di re ga in tri ge st* A tri ge is co si er to be ur ac kr ec if le or of its PR LE ev is ur ac kr ec

{TRIGGERS.PROBLEM.UNACK}	X	*Nu of ur ac kr ed PR LE tri ge fo a m el m* A tri ge is co si en to be ur ac kr ed if le or of its PR LE ev is ur ac kr ed Su si 1.

{TRIGGERS.ACK}	X	*Nu of ac kn ed tri ge fo a m el m di re ga in tri ge st* A tri ge is co si en to be ac kn ed if al of it' PP LE ev ar ac kn ed Su si 1.

{TRIGGERS.PROBLEM.ACK}	X			*Nu of ac kn ec PR LE tri ge fo a m el m* A tri ge is co si er to be ac kn ec if al of it' PR LE ev ar ac kn ec Su si 1. *Si m as us in bu in tri ge ex pr si*
{host:key.func(param)}	X[4]	X[10]	X[7]	

{$MACRO}	X[8]	X[8]	X[8]	X[8]	X[8]	X	X			X[9]	X	X	X	X	X			X[8]	X[8]	*Us de m* Su in ite ar tri ge na si 1. Su in gl sc co m ar co fir m tic te si Za bi 2.
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	

Footnotes

[1] Macros for map labels are supported since 1.8.

[2] The {HOST.*} macros supported in item key parameters will resolve to the interface that is selected for the item. They will not resolve when used in items without interfaces e.g. ”Zabbix agent (active)”, ”Calculated” etc.

[3] 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.

[4] This macro is supported in icon labels and link labels in maps. Only the **avg**, **last**, **max** and **min** functions, with seconds as parameter are supported in this macro.

[5] Supported since 2.0.3.

[6] Supported since Zabbix 2.2.0. {HOST.*} macros and user-defined macros {$MACRO} are supported in web scenario *Name*, *Variables*, *Headers*, *SSL certificate file* and *SSL key file* fields and in scenario step *Name*, *URL*, *Post*, *Headers* and *Required string* fields. {$MACRO} is also supported in web scenario *Agent*, *HTTP proxy*, *Authentication* (user and password) and *SSL key password* fields and in the scenario step *Required status codes* field.

[7] 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()}
```

[8] Supported since 2.4.0.

[9] Only in trigger expression constants and function parameters.

[10] While supported to build trigger expressions, simple macros may not be used inside each other.

Additional support for user macros

In addition to the locations listed, user-definable macros since Zabbix 2.0 are supported in numerous other locations:

- Hosts

    - Interface IP/DNS
    - Interface port
- Passive proxy
    - Interface port
- Items and item prototypes
    - SNMPv3 context name
    - SNMPv3 security name
    - SNMPv3 auth pass
    - SNMPv3 priv pass
    - SNMPv1/v2 community
    - SNMP OID
    - SNMP port
    - SSH username
    - SSH public key
    - SSH private key
    - SSH password
    - Telnet username
    - Telnet password
    - Calculated item formula
    - Trapper item "Allowed hosts" field (*since Zabbix 2.2*)
- Discovery

```
* SNMPv3 context name
* SNMPv3 security name
* SNMPv3 auth pass
* SNMPv3 priv pass
* SNMPv1/v2 community
* SNMP OID
```

Macros used in low-level discovery

There is a type of macro used within the low-level discovery function - **{#MACRO}**. It is a macro that is used in an LLD rule and returns real values of file system names, network interfaces and SNMP OIDs.

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.

LLD macros can be used:

- for item prototypes in
    - names
    - key parameters
    - SNMP OIDs
    - calculated item formulas
    - SSH and Telnet scripts
    - database monitoring SQL queries
    - descriptions (supported since 2.2.0)
- for trigger prototypes in
    - names
    - expressions (insofar as when referencing an item key prototype and as standalone constants)
    - descriptions (supported since 2.2.0)
- for graph prototypes in
    - names
- for host prototypes (supported since 2.2.0) in
    - names
    - visible names
    - host group prototype names
    - (see the full list)

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.

7 Setting time periods

1 Format

To set a time period, the following format has to be used:

`d-d,hh:mm-hh:mm`

You can specify more than one time period using a semicolon (;) separator:

`d-d,hh:mm-hh:mm;d-d,hh:mm-hh:mm...`

2 Description

Symbol	Description
d	Day of the week: 1 - Monday, 2 - Tuesday ,... , 7 - Sunday
hh	Hours: 00-24
mm	Minutes: 00-59

3 Default

Empty time period specification 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.

4 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`

8 Command execution

Zabbix uses common functionality to execute user parameters, remote commands, system.run[] items without the "nowait" flag, scripts (alert, external and global) and some internal commands.

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. 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:
Steps 5-7 do not refer to remote commands as they are executed with a "nowait" flag.

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.

Read more about user parameters, remote commands, alert scripts.

General

Monitoring server availability

At least three methods (or combination of all methods) may be used in order to monitor availability of a server.

- ICMP ping (”icmpping” key)
- ”zabbix[host,agent,available]” item
- trigger function nodata() for monitoring the availability of hosts that use active checks only

Sending alerts via WinPopUps

WinPopUps maybe very useful if you’re running Windows OS and want to get quick notification from Zabbix. It could be good addition for email-based alert messages. Details about enabling of WinPopUps can be found at http://www.zabbix.com/forum/showthread.php?t=2147.

Monitoring specific applications

AS/400

IBM AS/400 platform can be monitored using SNMP. More information is available at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg244504.html?Open.

MySQL

Several user parameters can be used for the monitoring of MySQL in the agent configuration file: /usr/local/etc/zabbix_agentd.conf

```
### Set of parameters for monitoring MySQL server (v3.23.42 and later)
### Change -u and add -p if required
#UserParameter=mysql.ping,mysqladmin -uroot  ping|grep alive|wc -l
#UserParameter=mysql.uptime,mysqladmin -uroot status|cut -f2 -d":"|cut -f2 -d" "
#UserParameter=mysql.threads,mysqladmin -uroot status|cut -f3 -d":"|cut -f2 -d" "
#UserParameter=mysql.questions,mysqladmin -uroot status|cut -f4 -d":"|cut -f2 -d" "
#UserParameter=mysql.slowqueries,mysqladmin -uroot status|cut -f5 -d":"|cut -f2 -d" "
#UserParameter=mysql.qps,mysqladmin -uroot status|cut -f9 -d":"|cut -f2 -d" "
#UserParameter=mysql.version,mysql -V
```

- *mysql.ping*

Check whether MySQL is alive.

```
Result: 0 - not started 1 - alive
```

- *mysql.uptime*

Number of seconds MySQL is running.

- *mysql.threads*

Number of MySQL threads.

- *mysql.questions*

Number of processed queries.

- *mysql.slowqueries*

Number of slow queries.

- *mysql.qps*

Queries per second.

- *mysql.version*

Version of MySQL. For example: mysql Ver 14.14 Distrib 5.1.53, for pc-linux-gnu (i686)

For additional information see also the userparameter_mysql.conf file in conf/zabbix_agentd directory.

Mikrotik routers

Use SNMP agent provided by Mikrotik. See http://www.mikrotik.com for more information.

Windows

Use Zabbix Windows agent included (pre-compiled) into Zabbix distribution.

Tuxedo

Tuxedo command line utilities tmadmin and qmadmin can be used in definition of a UserParameter in order to return per server/service/queue performance counters and availability of Tuxedo resources.

Informix

Standard Informix utility **onstat** can be used for monitoring of virtually every aspect of Informix database. Also, Zabbix can retrieve information provided by Informix SNMP agent.

HP OpenView

Zabbix can be configured to send messages to OpenView server. The following steps must be performed:

Step 1

Define new media.

The media will execute a script which will send required information to OpenView.

Step 2

Define new user.

The user has to be linked with the media.

Step 3

Configure actions.

Configure actions to send all (or selected) trigger status changes to the user.

Step 4

Write media script.

The script will have the following logic. If trigger is ON, then execute OpenView command *opcmsg -id application=<application> msg_grp=<msg_grp> object=<object> msg_text=<text>*. The command will return unique message ID which has to be stored somewhere, preferrably in a new table of ZABBIX database. If trigger is OFF then *opcmack <message id>* has to be executed with message ID retrieved from the database.

Refer to OpenView official documentation for more details about opcmsg and opcmack. The media script is not given here.

10 Performance tuning

Attention:
This is a work in progress.

Overview

It is very important to have Zabbix system properly tuned for maximum performance.

Hardware

General advice on hardware:

- Use fastest processor available
- SCSI or SAS is better than IDE (performance of IDE disks may be significantly improved by using utility hdparm) and SATA
- 15K RPM is better than 10K RPM which is better than 7200 RPM
- Use fast RAID storage
- Use fast Ethernet adapter
- Having more memory is always better

Operating system

- Use latest (stable!) version of OS
- Exclude unnecessary functionality from kernel
- Tune kernel parameters

Zabbix configuration parameters

Many parameters may be tuned to get optimal performance.

zabbix_server

StartPollers

General rule - keep value of this parameter as low as possible. Every additional instance of zabbix_server adds known overhead, in the same time, parallelism is increased. Optimal number of instances is achieved when queue, on average, contains minimum number of parameters (ideally, 0 at any given moment). This value can be monitored by using internal check zabbix[queue].

> **Note:**
> See the "See also" section at the bottom of this page to find out how to configure optimal count of zabbix processes.

DebugLevel

Optimal value is 3.

DBSocket

MySQL only. It is recommended to use DBSocket for connection to the database. That is the fastest and the most secure way.

Database engine

This is probably the most important part of Zabbix tuning. Zabbix heavily depends on the availability and performance of database engine.

- use fastest database engine, i.e. MySQL or PostgreSQL
- use stable release of a database engine
- rebuild MySQL or PostgreSQL from sources to get maximum performance
- follow performance tuning instructions taken from MySQL or PostgreSQL documentation
- for MySQL, use InnoDB table structure
- ZABBIX works at least 1.5 times faster (comparing to MyISAM) if InnoDB is used. This is because of increased parallelism. However, InnoDB requires more CPU power.
- tuning the database server for the best performance is highly recommended.
- keep database tables on different hard disks
- 'history', 'history_str, 'items' 'functions', triggers', and 'trends' are most heavily used tables.
- for large installations keeping MySQL temporary files in tmpfs is:
  - MySQL >= 5.5: not recommended (MySQL bug #58421)
  - MySQL < 5.5: recommended

GUI debugging

Problems related to the frontend performance may be diagnosed using the frontend debug mode.

General advice

- monitor required parameters only
- tune 'Update interval' for all items. Keeping a small update interval may be good for nice graphs, however, this may overload Zabbix
- tune parameters for default templates
- tune housekeeping parameters
- do not monitor parameters which return the same information.
- avoid the use of triggers with long period given as function argument. For example, max(3600) will be calculated significantly slower than max(60).

Viewing Zabbix process performance with "ps" and "top"

Zabbix 2.2 introduces a new feature - processes change their commandlines to display current activity and meaningful statistics, like:

```
UID        PID  PPID  C STIME TTY      TIME CMD
zabbix22  4584     1  0 14:55 ?    00:00:00 zabbix_server -c /home/zabbix22/zabbix_server.conf
zabbix22  4587  4584  0 14:55 ?    00:00:00 zabbix_server: configuration syncer [synced configuration in 0
zabbix22  4588  4584  0 14:55 ?    00:00:00 zabbix_server: db watchdog [synced alerts config in 0.018748 s
zabbix22  4608  4584  0 14:55 ?    00:00:00 zabbix_server: timer #1 [processed 3 triggers, 0 events in 0.0
zabbix22  4609  4584  0 14:55 ?    00:00:00 zabbix_server: timer #2 [processed 2 triggers, 0 events in 0.0
zabbix22  4637  4584  0 14:55 ?    00:00:01 zabbix_server: history syncer #4 [synced 35 items in 0.166198
zabbix22  4657  4584  0 14:55 ?    00:00:00 zabbix_server: vmware collector #1 [updated 0, removed 0 VMwar
zabbix22  4670     1  0 14:55 ?    00:00:00 zabbix_proxy -c /home/zabbix22/zabbix_proxy.conf
```

```
zabbix22  4673  4670  0 14:55 ?    00:00:00 zabbix_proxy: configuration syncer [synced config 15251 bytes
zabbix22  4674  4670  0 14:55 ?    00:00:00 zabbix_proxy: heartbeat sender [sending heartbeat message succ
zabbix22  4688  4670  0 14:55 ?    00:00:00 zabbix_proxy: icmp pinger #1 [got 1 values in 1.811128 sec, id
zabbix22  4690  4670  0 14:55 ?    00:00:00 zabbix_proxy: housekeeper [deleted 9870 records in 0.233491 se
zabbix22  4701  4670  0 14:55 ?    00:00:08 zabbix_proxy: http poller #2 [got 1 values in 0.024105 sec, id
zabbix22  4707  4670  0 14:55 ?    00:00:00 zabbix_proxy: history syncer #4 [synced 22 items in 0.008565 s
zabbix22  4738     1  0 14:55 ?    00:00:00 zabbix_agentd -c /home/zabbix22/zabbix_agentd.conf
zabbix22  4739  4738  0 14:55 ?    00:00:00 zabbix_agentd: collector [idle 1 sec]
zabbix22  4740  4738  0 14:55 ?    00:00:00 zabbix_agentd: listener #1 [waiting for connection]
zabbix22  4741  4738  0 14:55 ?    00:00:00 zabbix_agentd: listener #2 [processing request]
```

The main process is an exception. Instead of current activity the original commandline is shown. This helps to distinguish processes on systems with multiple Zabbix instances.

This feature is not implemented for Microsoft Windows.

If logging level is set to **DebugLevel=4** these activity and statistics messages are also written into log file.

Linux

On Linux systems `ps` command can be used together with `watch` command for observing how Zabbix is doing. For example, to run `ps` command 5 times per second to see process activities:

```
watch -n 0.2 ps -fu zabbix
```

To show only Zabbix proxy and agent processes:

```
watch -tn 0.2 'ps -f -C zabbix_proxy -C zabbix_agentd'
```

To show only history syncer processes:

```
watch -tn 0.2 'ps -fC zabbix_server | grep history'
```

The `ps` command produces a wide output (approximately 190 columns) as some activity messages are long. If your terminal has less than 190 columns of text you can try

```
watch -tn 0.2 'ps -o cmd -C zabbix_server -C zabbix_proxy -C zabbix_agentd'
```

to display only commandlines without UID, PID, start time etc.

`top` command also can be used for observing Zabbix performance. Pressing 'c' key in `top` shows processes with their command-lines. In our tests on Linux `top` and `atop` correctly displayed changing activities of Zabbix processes, but `htop` was not displaying changing activities.

BSD systems

If `watch` command is not installed, a similar effect can be achieved with

```
while [ 1 ]; do ps x; sleep 0.2; clear; done
```

AIX, HP-UX

If `watch` command is not available, one can try

```
while [ 1 ]; do ps -fu zabbix; sleep 1; clear; done
```

Solaris

By default the `ps` command does not show changing activities. One option is to use `/usr/ucb/ps` instead. If `watch` command is not installed, a periodically updated list of processes can be shown with

```
while [ 1 ]; do /usr/ucb/ps gxww; sleep 1; clear; done
```

On Solaris 11:

- `/usr/ucb/ps` is not installed by default. You may need to install *ucb* package, e.g. `pkg install compatibility/ucb`,
- if Zabbix daemon has been started by privileged user its activities are not shown to non-privileged user.
- the `sleep` command accepts not only whole seconds but also fractions of second (e.g. `sleep 0.2`).

See also

1. How to configure optimal count of zabbix processes

11 Version compatibility

Supported agents

Older agents from Zabbix 1.x and previous versions of Zabbix 2.x can still be used with Zabbix 2.4. It does not require any configuration changes on agent side.

However, to take full advantage of new and improved items, improved performance and reduced memory usage, use the latest 2.4 agent.

Supported Zabbix proxies

Only Zabbix 2.4 proxies may be used with Zabbix server. Zabbix 1.6, 1.8, 2.0 and 2.2 proxies are not supported with Zabbix 2.4 server.

Zabbix 2.4 proxies may only be used with Zabbix 2.4 server. They will not work with 2.2 or older Zabbix server.

Supported XML files

XML files, exported with 1.8, 2.0 and 2.2 are supported for import in Zabbix 2.4.

> **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 recognised.

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;
```

Zabbix manpages

These are Zabbix manpages for Zabbix processes.

zabbix_agentd

Section: Maintenance Commands (8)
Updated: 10 November 2011
Index Return to Main Contents

NAME

zabbix_agentd - Zabbix agent daemon.

SYNOPSIS

zabbix_agentd [-hpV] [-c <*config-file*>**] [-t** <*item-key*>**]**

DESCRIPTION

zabbix_agentd is a daemon for monitoring of various server parameters.

Options -c, --config <*config-file*>
Use the alternate *config-file* instead of the default one. Absolute path should be specified.

-R, --runtime-control <*option*>
Perform administrative functions according to *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

<pid>
Process identifier

<process type>
All processes of specified type (e.g., listener)

<process type>,N
Process type and number (e.g., listener,3)

-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

zabbix_get(8), **zabbix_proxy**(8), **zabbix_sender**(8), **zabbix_server**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION

Options

FILES

SEE ALSO

AUTHOR

This document was created by man2html, using the manual pages.
Time: 23:32:57 GMT, September 15, 2014

zabbix_get

Section: Maintenance Commands (8)
Updated: 5 July 2011
Index Return to Main Contents

NAME

zabbix_get - Zabbix get utility.

SYNOPSIS

zabbix_get [-hV] [-s <*host name or IP*>**] [-p** <*port number*>**] [-I** <*IP address*>**] [-k** <*item key*>**]**

DESCRIPTION

zabbix_get is a command line utility for getting data from a remote 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.

-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]

SEE ALSO

zabbix_agentd(8), **zabbix_proxy**(8), **zabbix_sender**(8), **zabbix_server**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION
Options

EXAMPLES

SEE ALSO

AUTHOR

This document was created by man2html, using the manual pages.
Time: 14:47:43 GMT, July 23, 2012

zabbix_proxy

Section: Maintenance Commands (8)
Updated: 10 November 2011
Index Return to Main Contents

NAME

zabbix_proxy - Zabbix proxy daemon.

SYNOPSIS

zabbix_proxy [-hV] [-c <*config-file*>**] [-R** <*option*>**]**

DESCRIPTION

zabbix_proxy is a daemon used for remote data collection.

Options -c, --config <*config-file*>
Use the alternate *config-file* instead of the default one. Absolute path should be specified.

-R, --runtime-control <*option*>
Perform administrative functions according to *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.

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

<pid>
Process identifier

<process type>
All processes of specified type (e.g., poller)

<process type>,N
Process type and number (e.g., poller,3)

-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

zabbix_agentd(8), **zabbix_get**(8), **zabbix_sender**(8), **zabbix_server**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION

Options

FILES

SEE ALSO

AUTHOR

This document was created by man2html, using the manual pages.
Time: 23:35:33 GMT, September 15, 2014

zabbix_sender

Section: Maintenance Commands (8)
Updated: 16 October 2015
Index Return to Main Contents

NAME

zabbix_sender - Zabbix sender utility.

SYNOPSIS

zabbix_sender [-hpzvIV] {-kso | [-T] -i <*inputfile*>**} [-c** <*config-file*>**]**

DESCRIPTION

zabbix_sender is a command line utility for sending performance data to a remote Zabbix server. 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 agent configuration file. By default Zabbix sender does not read any configuration file. Absolute path should be specified. Only parameters **Hostname**, **ServerActive** and **SourceIP** are supported. First entry from the **ServerActive** parameter is used.

-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.

-p, --port <*port*>
Specify port number of server trapper running on the server. Default is 10051.

-s, --host <*host*>
Specify agent hostname as registered in Zabbix frontend. Host IP address and DNS name will not work.

-I, --source-address <*IP*>
Specify source IP address.

-k, --key <*key*>
Specify item key to send value to.

-o, --value <*value*>
Specify value.

-i, --input-file <*inputfile*>
Load values from input file. Specify **-** as <inputfile> to read values from standard input.

Each value must be specified on its own line. Each line must contain 3 whitespace delimited entries: **<hostname> <key> <value>**, where ”hostname” is the name of monitored host as registered in Zabbix frontend, ”key” is target item key and ”value” - the value to send. Specify **-** as **<hostname>** to use hostname from agent configuration file or from **--host** argument.

An example of a line of an input file:

”Linux DB3” db.connections 43

The value type must be correctly set in item configuration of Zabbix frontend. Zabbix sender will send up to 250 values in one connection. Contents of the input file must be in the UTF-8 encoding. All values from the input file are sent in a sequential order top-down. Entries must be formatted using the following rules:

•
Quoted and non-quoted entries are supported.

•
Double-quote is the quoting character.

•
Entries with whitespace must be quoted.

•
Double-quote and backslash characters inside quoted entry must be escaped with a backslash.

•
Escaping is not supported in non-quoted entries.

•
Linefeed escape sequences (\n) are supported in quoted strings.

•
Linefeed escape sequences are trimmed from the end of an entry.

-T, --with-timestamps
This option can be only used with **--input-file** option.

Each line of the input file must contain 4 whitespace delimited entries: **<hostname> <key> <timestamp> <value>**. Timestamp should be specified in Unix timestamp format. If target item has triggers referencing it, all timestamps must be in an increasing order, otherwise event calculation will not be correct.

An example of a line of the input file:

"Linux DB3" db.connections 1429533600 43

For more details please see option **--input-file**.

-r, --real-time
Send values one by one as soon as they are received. This can be used when reading from standard input.

-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 -s Monitored Host -k mysql.queries -o 342.45

Send **342.45** as the value for **mysql.queries** key in **Monitored Host** host using Zabbix server defined in agent daemon configuration file.

zabbix_sender -z 192.168.1.113 -i data_values.txt

Send values from file **data_values.txt** to 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 daemon 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.

SEE ALSO

zabbix_agentd(8), **zabbix_get**(8), **zabbix_proxy**(8), **zabbix_server**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION
Options

EXIT STATUS

EXAMPLES

SEE ALSO

AUTHOR

This document was created by man2html, using the manual pages.
Time: 09:47:39 GMT, February 14, 2017

zabbix_server

Section: Maintenance Commands (8)
Updated: 10 November 2011
Index Return to Main Contents

NAME

zabbix_server - Zabbix server daemon.

SYNOPSIS

zabbix_server [-hV] [-c <*config-file*>] [-R <*option*>]

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. Absolute path should be specified.

-R, --runtime-control <*option*>
Perform administrative functions according to *option*.

Runtime control options

config_cache_reload
Reload configuration cache. Ignored if cache is being currently loaded. Default configuration file (unless **-c** option is specified) will be used to find PID file and signal will be sent to process, listed in PID file.

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

<pid>
Process identifier

<process type>
All processes of specified type (e.g., poller)

<process type>,N
Process type and number (e.g., poller,3)

-h, --help
Display this help and exit.

-V, --version
Output version information and exit.

FILES

/usr/local/etc/zabbix_server.conf
Default location of Zabbix server configuration file (if not modified during compile time).

SEE ALSO

zabbix_agentd(8), **zabbix_get**(8), **zabbix_proxy**(8), **zabbix_sender**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION

Options

FILES

SEE ALSO

AUTHOR

This document was created by man2html, using the manual pages.
Time: 23:38:26 GMT, September 15, 2014

Zabbix 2.4 API

api reference templatescreen create isreadable exists massremove getobjects delete get massadd massupdate iswritable update object get object httptest exists get object create isreadable exists getobjects delete get iswritable update deletedependencies object adddependencies create isreadable exists delete get iswritable update object create exists getobjects delete get update object hostgroup create isreadable updatebyposition delete get iswritable update object mediatype script host screen exists get object screenitem create exists getobjects delete get update object iconmap create execute delete get update getscriptsbyhosts object create exists delete get update object configuration apiinfo user event get object create isreadable exists getobjects delete get iswritable update object create isreadable exists getobjects delete get massadd massupdate iswritable update object export import hostinterface image create isreadable delete get iswritable update object proxy dservice create isreadable exists massremove getobjects delete get massadd massupdate iswritable update object item application alert create isreadable delete get iswritable update object usergroup create exists delete get massadd update object create delete get update object create isreadable exists delete get iswritable update object get object history dhost maintenance create exists replacehostinterfaces massremove delete get massadd update object create isreadable exists massremove getobjects delete get massadd massupdate iswritable update object discoveryrule triggerprototype graphitem create isreadable delete get iswritable update object service create isreadable exists delete copy get iswritable update object itemprototype create isreadable exists delete copy get iswritable update object create delete get update object graph get object logout create isreadable delete get login deletemedia iswritable updateprofile update addmedia object updatemedia create exists delete get update object create isreadable getsla delete get iswritable update deletedependencies deletetimes object addtimes adddependencies action acknowledge get object graphprototype create isreadable delete get iswritable update object drule template version usermacro trigger hostprototype map templatescreenitem create isreadable exists getobjects delete get iswritable update object get object create exists getobjects delete get update object dcheck get object create exists delete get update object create updateglobal delete get createglobal update deleteglobal object usermedia reference_commentary changes_2.4 changes_2.2_-_2.4