# Documentation 1.8

ZABBIX

10.04.2025

## Contents

# Zabbix Manual

This is Zabbix manual for version 1.8.

All content is available by exploring the individual chapters on the left.



## 1 About



### 1 Overview of Zabbix

1.1 What is Zabbix?

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 front end. A web-based front end 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.

1.2 What does Zabbix offer?

Zabbix offers:

- auto-discovery of servers and network devices
- distributed monitoring with centralised WEB administration
- support for both polling and trapping mechanisms
- server software for Linux, Solaris, HP-UX, AIX, Free BSD, Open BSD, OS X
- native high performance agents (client software for Linux, Solaris, HP-UX, AIX, Free BSD, Open BSD, OS X, Tru64/OSF1, Windows NT4.0, Windows 2000, Windows 2003, Windows XP, Windows Vista)
- agent-less monitoring
- secure user authentication
- flexible user permissions
- web-based interface
- flexible e-mail notification of predefined events
- high-level (business) view of monitored resources
- audit log

1.3 Why use Zabbix?

- Open Source solution
- highly efficient agents for UNIX and WIN32 based platforms
- low learning curve
- high ROI. Downtimes are very expensive.
- low cost of ownership
- very simple configuration
- Centralised monitoring system. All information (configuration, performance data) is stored in relational database
- high-level service tree
- very easy setup
- support for SNMP (v1,v2). Both trapping and polling.
- visualisation capabilities
- built-in housekeeping procedure

1.4 Users of Zabbix

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

## 2 Goals and Principles

2.1 Main Goals of Zabbix Development

There are several goals Zabbix is trying to achieve:

- become recognized Open Source monitoring tool
- create Zabbix user group, which helps making the software even better
- provide high-quality commercial support

2.2 Main principles of Zabbix development

- be user friendly
- keep things simple
- use as few processing resources as possible
- react fast
- document every aspect of the software

## 3 Installation and Upgrade Notes

3.1 Installation

See the installation_from_source section for full details.

3.2 Version compatibility

Older agents from Zabbix 1.0, Zabbix 1.1.x, Zabbix 1.4.x and Zabbix 1.6.x can be used with Zabbix 1.8. It does not require any configuration changes on agent side.

> **Warning:**
> Older Zabbix proxies of version 1.6.x can't be used with Zabbix 1.8, they should be upgraded.

3.3 Important notes

3.3.1 For version 1.8

- All hosts now are required to belong to at least one group.
- CPU index for **system.cpu.util** key on Linux now starts with 0.
- Key **vfs.fs.size** returns data in bytes for all operating systems now.
- Key **vfs.fs.size** now takes into account reserved diskspace for root user.
- Comment at the end of a configuration file line is not allowed anymore (this worked for numeric parameters only before).

3.3.2 For version 1.8.3

- Parameter **service.ntp** for item keys **net.tcp.service** and **net.tcp.service.perf** renamed to **ntp**. Old syntax is still supported.
- Trying to run IPv6-enabled daemon on a system without IPv6 support fail:

`Listener failed with error: socket() for [[(null)]:10051] failed with error 97: Address family not support`

3.3.3 For version 1.8.5

- The method of external command invocation for Zabbix daemons has been changed to allow terminating runaway processes. Instead of using standard *popen* method as before, Zabbix now explicitly calls **/bin/sh** to execute desired command.
- Trying to run a Zabbix daemon, compiled on Linux kernel 2.6.27 or later on a system with kernel 2.6.26 or older will fail:`socket() for [[-]:10050] failed with error 22: Invalid argument`

3.3.4 For version 1.8.6

- Zabbix daemons now refuse to start up if configuration file contains incorrect parameters. If old parameters have accumulated in the configuration files, this will result in inoperable daemons after the upgrade until the parameters are fixed.

3.3.5 For version 1.8.8

- In some cases hosts and proxies with identical name might have appeared in the Zabbix database. In 1.8.8, Zabbix server will shut down if it detects such a situation. This check was removed in 1.8.9.

3.3.6 For version 1.8.9

The shutdown upon detection of duplicate hosts, introduced in 1.8.8, has been removed.

3.3.7 For version 1.8.16

Accepted data limit of 128MB was introduced when using Zabbix protocol. Any other data (including older Zabbix protocols) stays limited at 16MB.

3.3.8 For version 1.8.18

Zabbix server now correctly enables SSL host verification when using Ez Texting service to send alerts.

3.3.9 For version 1.8.20

If Zabbix is logging to syslog then after an upgrade to Zabbix 1.8.20 you will see changes in the application names appearing in syslog:

```
Zabbix Agent  → zabbix_agent
Zabbix Agent  → zabbix_agentd
Zabbix Proxy  → zabbix_proxy
Zabbix Server → zabbix_server
Zabbix Get    → zabbix_get
Zabbix Sender → zabbix_sender
```

The old, incorrect names (on the left) contained a space which is not allowed by RFC 5424 for APP-NAME. If you are using regular expressions in monitoring of syslog you may want to adjust them for the new application names.

3.4 System requirement changes

Additional or increased system requirements:

- Support for PHP 4 dropped.
- Maximal PHP memory size should be at least 128MB (option **memory_limit**).
- Maximal PHP POST size should be at least 16MB (option **post_max_size**).

Also see requirement changes for versions 1.8.2, 1.8.3 and 1.8.9.

3.5 Known problems

**Warning:**
Zabbix frontend in 1.8 does not work with SQLite backend. Please, use one of the other supported databases.

3.5.1 For version 1.8

- PHP mbstring check may fail with PHP < 5.2 in Zabbix 1.8. To avoid this issue, copy **zabbix.conf.php.example** file to **zabbix.conf.php** and modify parameters, including database access parameters.
- For IPMI support you need a recent OpenIPMI version - 2.0.14 and later is known to work.
- Sorting in frontend is not performed for entities with positional variables (like item names with $1 etc).
- XML export includes SNMP and other information for all items.
- Hostnames with spaces do not work when sending data from a file with **zabbix_sender** (fixed in 1.8.2).
- Uploading of images for network maps may fail if database is not configured properly. Make sure database, tables and fields are set to UTF-8 encoding.

- Precompiled binaries (agent, sender, get) might not work on 64bit systems with glibc versions older than 2.5. Common symptom is failing to start with the error message: *Floating point exception*. Use older versions, or compile from the scratch on the target system.

3.5.2 For version 1.8.2

Because of frontend changes, some installations might see incorrect older data appear in frontend. These include:

- Incorrect trigger appearing, with name ****ERROR****, usually in Monitoring → Triggers section, when showing all hosts from all groups. This trigger can be deleted by clicking on it, choosing **Configuration of triggers**, then clicking on **Delete** in the trigger editing form and confirming the deletion.

> **Attention:**
> You might have to remove *groupid=&hostid=&* part from the URL when attempting to delete the trigger.

- Depending on the installation time of your Zabbix server, default graphs might have incorrect configuration. This only affects you if those graphs are being used. Opening such a graph usually will swap working time and trigger showing with percentile values. If that is the case, simply fixing and saving the graph will solve the problem.

3.6 Upgrade procedure

The following steps have to be performed for successful upgrade from Zabbix 1.6.x to 1.8. The whole upgrade procedure may take several hours depending on size of Zabbix database.

3.6.1 Stop Zabbix server

Stop Zabbix server to make sure that no new data is inserted into database.

3.6.2 Backup existing Zabbix database

This is very important step. Make sure that you have backup of your database. It will help if upgrade procedure fails (lack of disk space, power off, any unexpected problem).

3.6.3 Backup configuration files, PHP files and Zabbix binaries

Make a backup copy of Zabbix binaries, configuration files and PHP files.

3.6.4 Install new server binaries

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

3.6.5 Review Server configuration parameters

Some parameters of zabbix_server.conf were changed in 1.8, new parameters added. You may want to review them.

- Configuration option **StartDBSyncers** has been removed from Zabbix server and proxy configuration files.

3.6.6 Upgrade database

> **Attention:**
> Database upgrade is a required step when upgrading from one major Zabbix version to another, such as from 1.6 to 1.8. It is not required for minor upgrades, such as from 1.8.x to 1.8.x, unless specifically stated so in the release notes of the version.

Before running upgrade scripts drop the following indexes:

**MySQL**

```
alter table dhosts drop index dhosts_1;
alter table dservices drop index dservices_1;
alter table httptest drop index httptest_2;
alter table httptest drop index httptest_3;
alter table history_log drop index history_log_2;
alter table history_text drop index history_text_2;
alter table actions drop index actions_1;
alter table escalations drop index escalations_2;
alter table graphs_items drop index graphs_items_1;
alter table graphs_items drop index graphs_items_2;
alter table services drop index services_1;
```

**Oracle or PostgreSQL**

```
drop index dhosts_1;
drop index dservices_1;
drop index httptest_2;
drop index httptest_3;
drop index history_log_2;
drop index history_text_2;
drop index actions_1;
drop index escalations_2;
drop index graphs_items_1;
drop index graphs_items_2;
drop index services_1;
```

**Ignore any warning messages about non-existent indexes!**

Database upgrade scripts are located in directory upgrades/dbpatches/1.8/<db engine>:

- MySQL: upgrades/dbpatches/1.8/mysql/patch.sql
- Oracle: upgrades/dbpatches/1.8/oracle/patch.sql
- PostgreSQL: upgrades/dbpatches/1.8/postgresql/patch.sql

Database upgrade should take around 10-15 minutes, for PostgreSQL it may take several hours or more because of conversion of existing historical data. It is recommended to test the upgrade procedure in a non-production environment.

> **Attention:**
> If you are converting the database to UTF-8, it can take many hours.

Make sure that you have enough permissions (create table, drop table, create index, drop index). Also make sure that you have enough free disk space.

These scripts are for upgrade from Zabbix 1.6.x to 1.8 only! For upgrade from earlier versions use also upgrade scripts from Zabbix 1.6.x.

3.6.7 Install new Zabbix GUI

Follow installation instructions.

3.6.8 Start new Zabbix binaries

Start new binaries. Check log files to see if the binaries have started successfully.

## 4. What's new in Zabbix 1.8

More than a year in making, Zabbix 1.8 has arrived with lots of new features, as well as improved old ones. You can introduce yourself to the changes for this new version of Zabbix in the following section.

**1 Notable improvements** With so many changes it is impossible to pick 3 most notable ones - which is attempted below. For this reason, it is suggested to read on, as some generally minor feature might be very important to you.

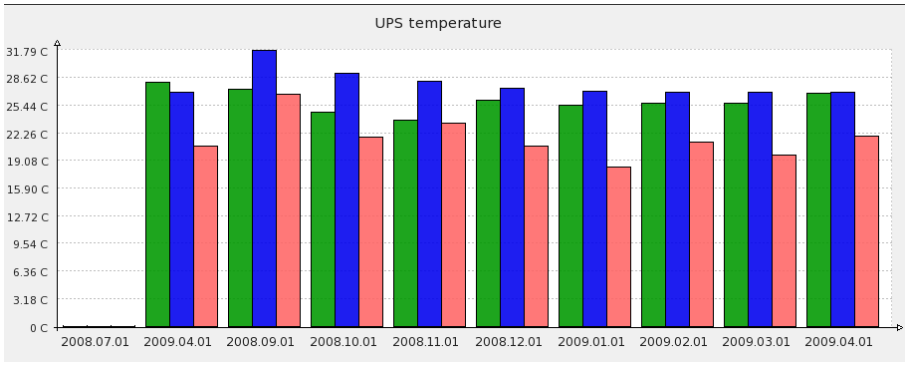
- Performance improvements
- Full Unicode support
- Bar reports
- API technical preview

**2 New features and improvements for the frontend** Zabbix web frontend is the feature that sets it apart from other solutions. Powerful, yet easy to use official GUI is shipped with the default package. It provides both non-intimidating access for novice users and large scale configuration features for sizable installations.

Being most user-visible part, we will start by looking at many new features and improvements in Zabbix 1.8 for the web frontend.

**2.1 Bar reports** Zabbix already has easy to use simple graphs that do not require any configuration - these are provided for every numeric item. Custom graphs, along with a couple simplistic reports, allow to look at the data in context. Zabbix 1.8 brings much more powerful built-in reporting.

New report category, bar reports, allows to look at the data from many different angles. Want to look at the weekly temperatures in the server room for the last two months? Have to compare webserver load for the first month of every quarter this year? All that and more is possible with this new feature.

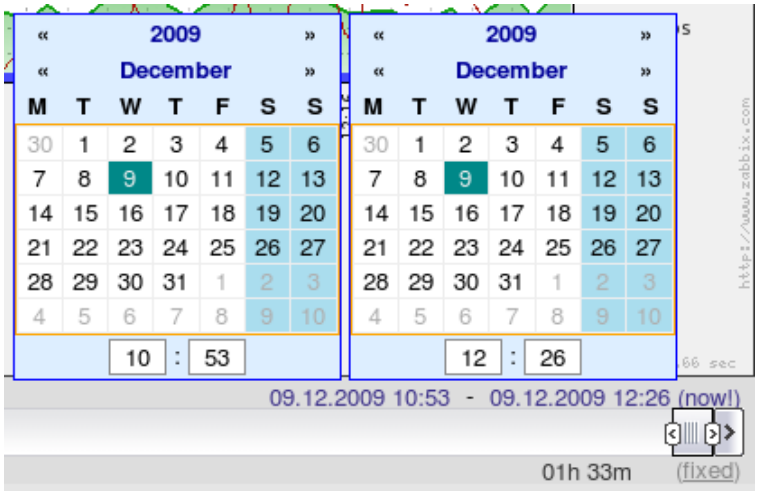



**2.2 Full Unicode support** While previous Zabbix versions were multi-language friendly, providing several frontend translations, it was not a truly global thing - the most popular encoding, Unicode, was not fully supported.

Zabbix 1.8 now fully supports Unicode/UTF-8, allowing for a true localised or multilanguage setup.

**2.3 Improved time period navigation** In Zabbix, single control is used to select time period displayed for many views, including simple and custom graphs, raw data, screens and web monitoring. Already improved in 1.6, time period selector has been improved in 1.8 further.

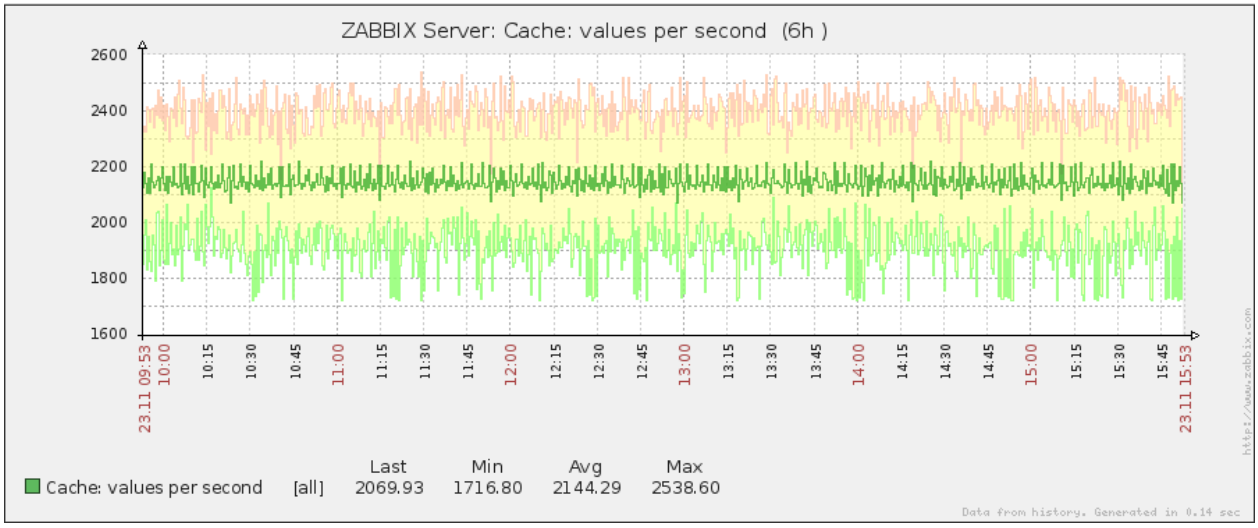

This scrollbar allows easy dragging and resizing of it. Additionally, links on the left hand side allow to choose some predefined, often used time periods and move displayed period forward/backward in time. And the dates on the right hand side actually work as links, popping up a calendar and allowing to set specific start/end time.




Notice the dynamic/fixed link in the lower right hand corner? It can be used to control whether time period is kept constant when you change start/end time.

In addition to the screenshots you can also view the video of using graph time period controls.

**2.4 Improved graphs** Zabbix graphs have been improved in many ways. This includes both visual and functional improvements, like the time period selector already mentioned. For example, information about max/min/avg values is presented clearly as a table.




2.4.1 Improved timeline in charts

Zabbix graphs - or charts - usually display time on x axis. And even this representation has been improved in the new version.

Comparing 1.6 and 1.8:

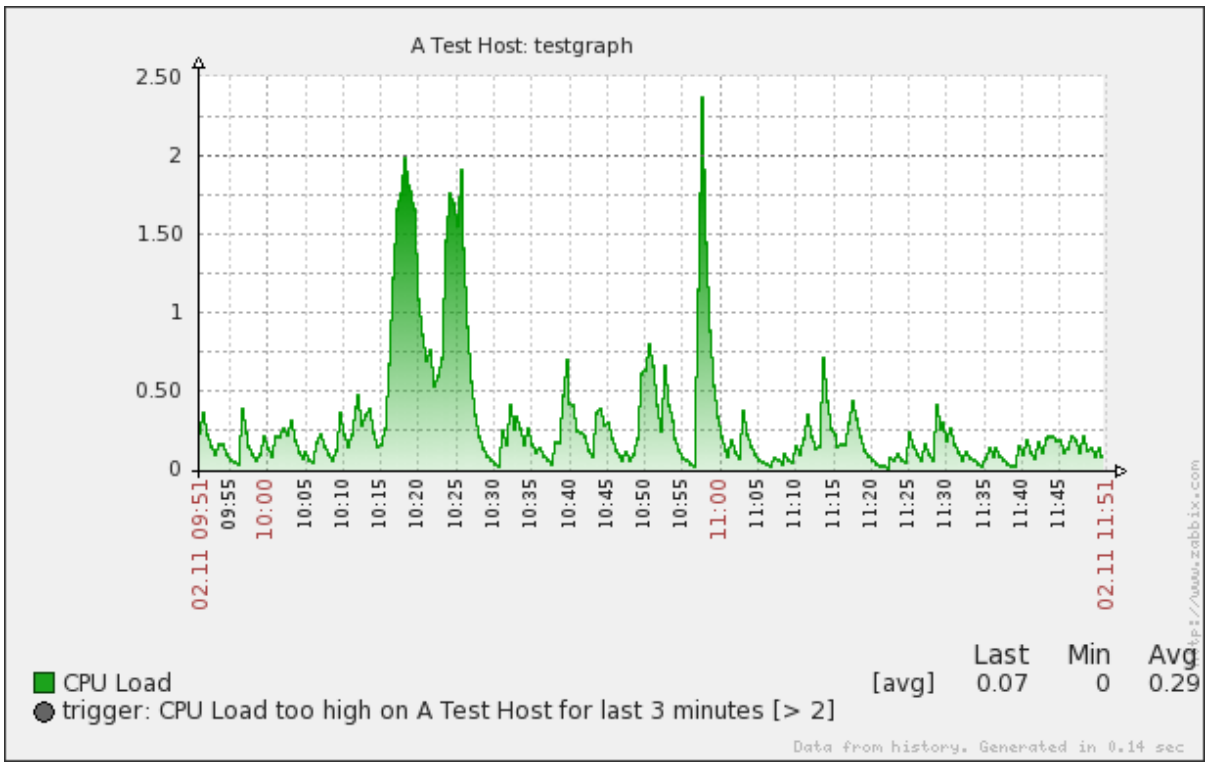
|<| |<| |-|

|<| |<| |-|

As can be seen, labels are now easier to read. Instead of prioritising some arbitrary point in time, depending on graph scale, actual points in time like change of a day are prioritised. Sometimes Zabbix will even use more ”human readable” labels:


2.4.2 Gradient line support in graphs

Zabbix graphs support several line styles, and 1.8 brings one more - gradient line. It’s easier to understand how that works by looking at an actual example.



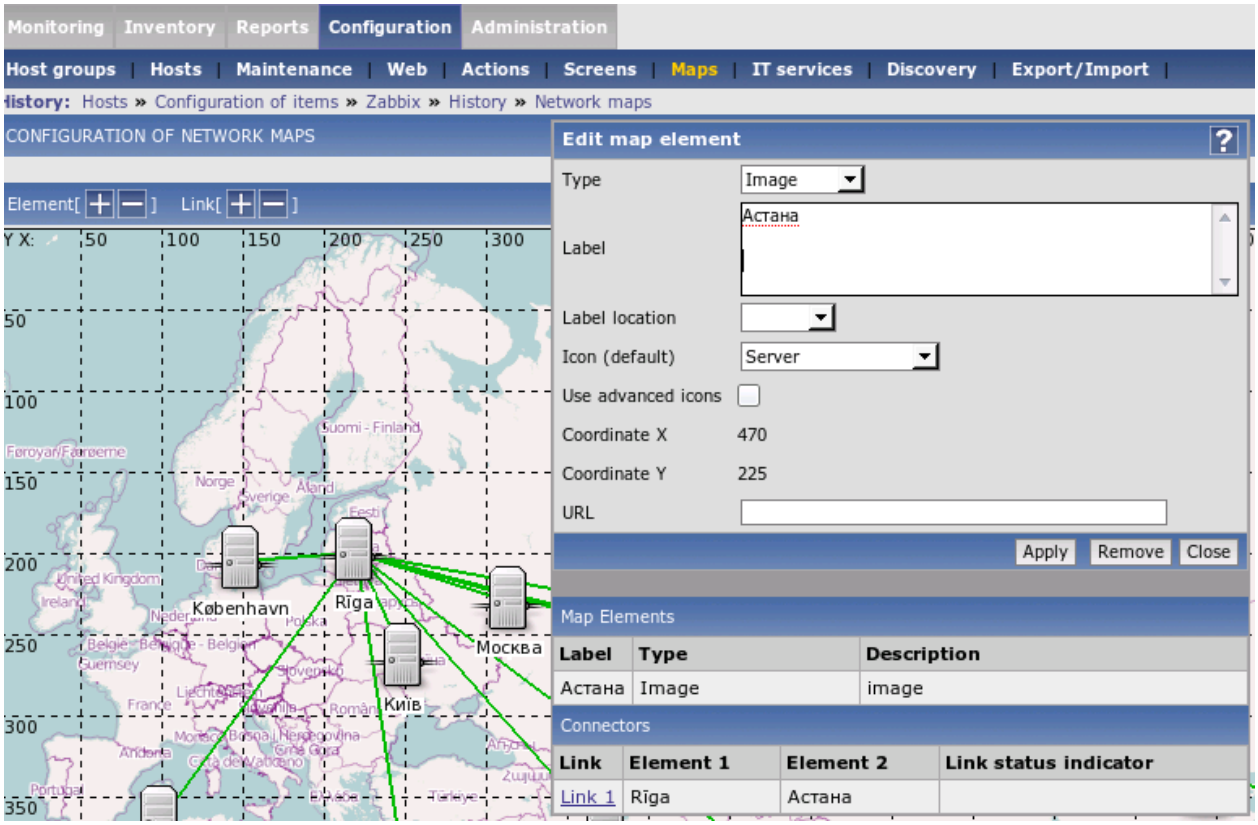
2.4.3 DejaVu font used for graphs

DejaVu font is now used for graphs for nice looking text - and for Unicode capabilities.

**2.5 Improved map editing** Zabbix supports network maps where monitored environment can be laid out on a background image to provide user friendly overview.

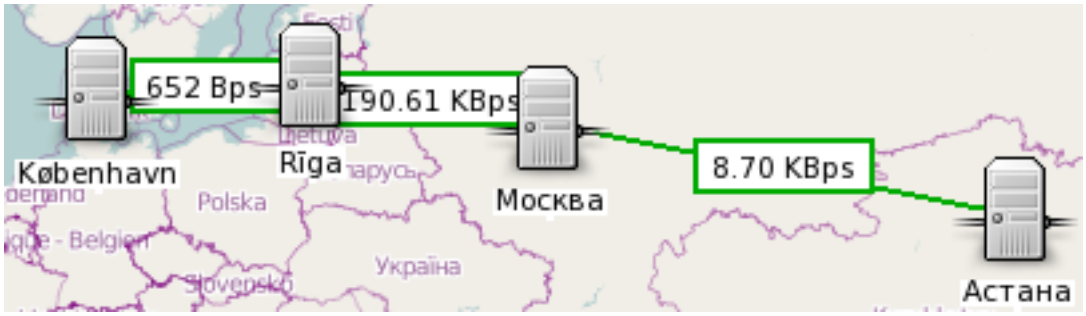
In previous versions, editing such network maps was not easy - coordinates of each element on the map had to be specified manually.

Map editing in Zabbix 1.8 has been greatly improved by adding drag and drop support, as well as selected element detail displaying in a popup.




You can even watch a video of map editing.

In addition to that map links may have optional text for displaying arbitrary information, for example, bandwidth data.

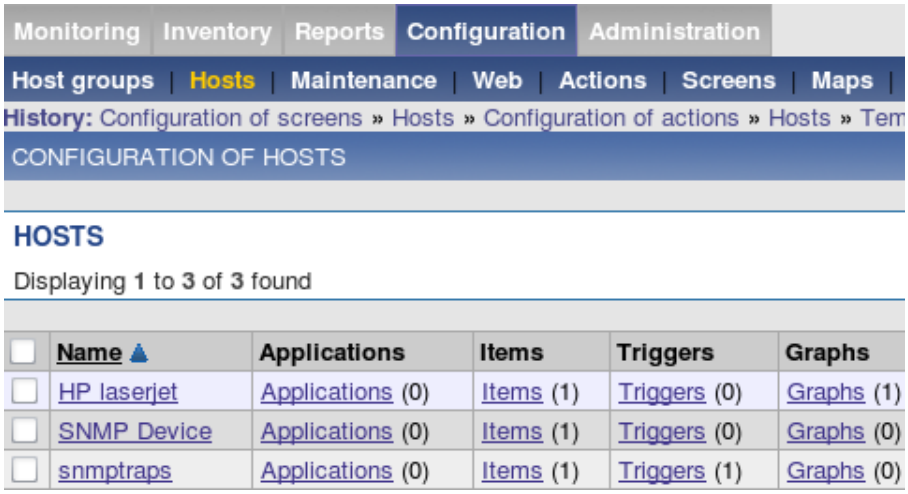



Background map CC-BY-SA Openstreetmap.

**2.6 Changed configuration layout** Zabbix web frontend provides convenient way to display and visualise received data, as well as configure all aspects of monitoring.

Layout of this configuration has been redone in 1.8. Instead of separate sections for items, graphs, triggers and so on, they are folded into host configuration section, where convenient linking allows for easy access to all of these entities and more.

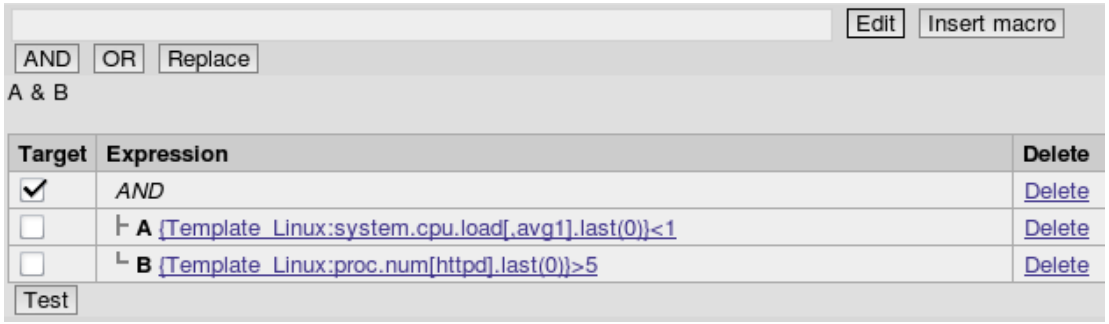
On the other hand, host group configuration has been brought out on the configuration menu.




Another change is general configuration being moved to administration section to avoid Zabbix administrator level users from having access to global configuration parameters.

**2.7 Visual trigger editing frontend** Usage thresholds and any other problem conditions are freely configurable by user. These definitions are called triggers, and complex expressions can be used for each trigger to define what is considered a problem.

In addition to ability to edit trigger expressions directly, a frontend to create triggers visually has been added.




There is a special mode for creating log related triggers.

It also incorporates ability to provide test data and try out trigger behaviour.

(screenshot)

**2.8 New and improved filters** As Zabbix frontend provides means to access all the information, it can be a daunting task to find the desired one. Previous versions offered ways to filter this information, and 1.8 improves situation in this regard by adding new filters and making existing ones more powerful.

2.8.1 Items filter

Item configuration section is the one where all aspects regarding data gathering are configured, thus it is displaying quite a lot of information. Being able to quickly find desired data gathering entries is crucial to efficient configuration, and in Zabbix 1.8 there's an improved filter for items that allows for much more detailed searching.

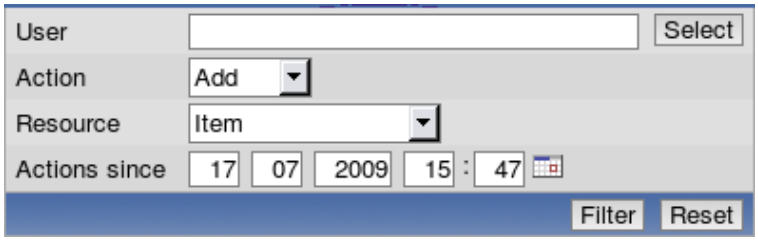
(screenshot)

After performing initial filtering, subfilter becomes available. It presents found values and results can be filtered further.


2.8.2 Audit filter

Accountability is important on any system with more than one user (and on many systems with single user as well). Zabbix frontend records all operations in an audit log.

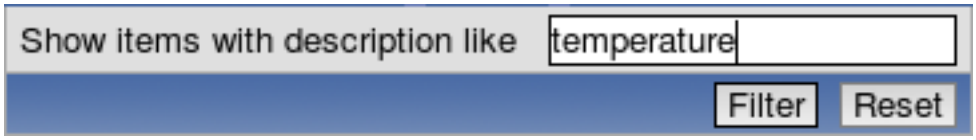
In version 1.8 audit logs now can be filtered quite extensively to find exactly the changes you are looking for.




2.8.3 Latest data filter

Looking at shiny graphs is tempting, but sometimes you need the real data. Latest data section in Zabbix frontend allows to see exact values for all monitored metrics.

It is now possible to filter this screen by freeform search against item descriptions.




2.8.4 Reworked ”Status of triggers” view

Trigger view is widely used to display list of current problems, and it was possible to display recent events for all the problems, limited by day count.

In 1.8, this screen gained has been changed, providing new features like expanding individual triggers to show their events and confirming all events for a trigger.

| Severity | Status | Last change | Age | Duration | Acknowledged | Host | Name | Comments |
|---|---|---|---|---|---|---|---|---|
| Average | PROBLEM | 30 Nov 2009 17:32:39 | 3d 6h 6m | | Acknowledge (119) | Another Host | SMTP service is down | Add |
| | PROBLEM | 30 Nov 2009 17:32:39 | 3d 6h 6m | 3d 6h 6m | Not acknowledged | | | |
| | UNKNOWN | 30 Nov 2009 17:32:31 | 3d 6h 6m | 8s | | | | |
| | PROBLEM | 30 Nov 2009 09:21:52 | 3d 14h 17m | 8h 10m 39s | Not acknowledged | | | |
| | UNKNOWN | 30 Nov 2009 09:21:31 | 3d 14h 17m | 21s | | | | |
| | PROBLEM | 27 Nov 2009 13:55:35 | 6d 9h 43m | 2d 19h 25m | Not acknowledged | | | |

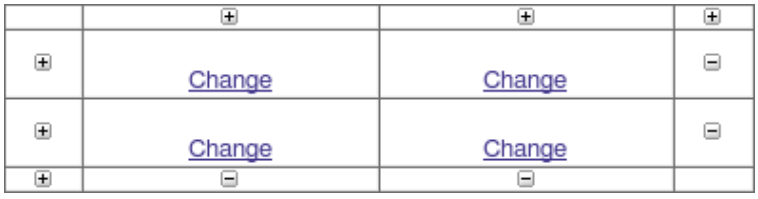
2.8.5 Other filters improved

Filters in other sections of the frontend have been improved as well, allowing to get to the data easier and more quickly.

**2.9 Improved screen editing** Zabbix screens is a feature that allows to group many frontend elements, including graphs, network maps, raw data and many others. Configuring them initially was not very hard, but making any significant changes was nearly impossible in some cases.

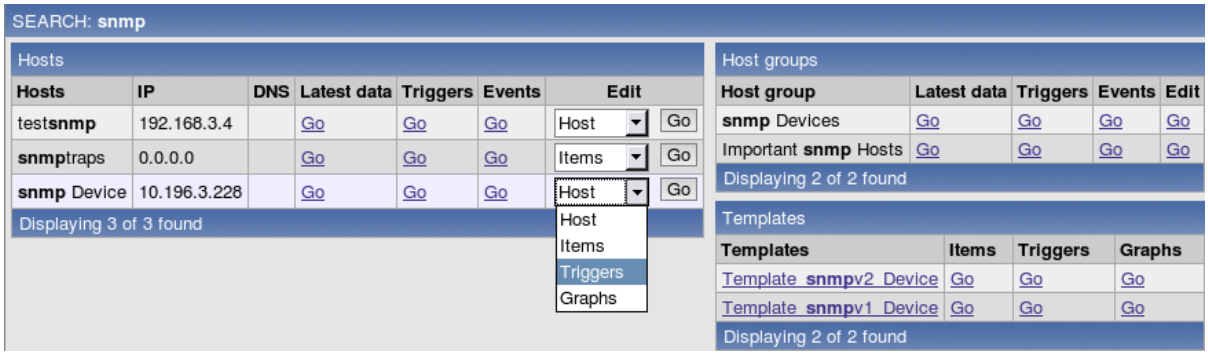
Screen editing has been greatly improved in 1.8. This includes:

- Drag and drop support. Dragging an element to empty cell will move it there, dragging an element on occupied cell will switch these elements. You can watch a video of this feature (site also allows to download original .ogg video).
- Using icons on the screen edges, rows now can be inserted and removed from arbitrary locations.




**2.10 Global search** There’s now a search box in the upper right corner, which allows searching in hosts, host groups and templates.

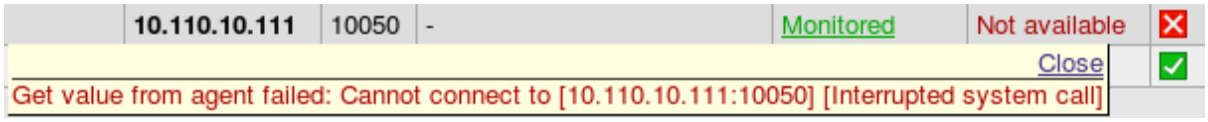

Results allow for a quick access to found entities and their elements:




**2.11 Minor frontend improvements** For a GUI minor visual change can bring large benefits to the user. Zabbix 1.8 has many minor improvements and features that should make working more productive and pleasant.

2.11.1 Cleaner error displaying

Error messages are now shown as icons and error text is available in a popup. Clicking the icon opens the popup to allow copying of the message.




2.11.2 History strings saved by reference

At the top of the frontend, there's a breadcrumbs type history, showing recently accessed pages.


When a language is switched in frontend, in previous versions existing history entries would not switch language, only new entries would be added in the correct language. Now history strings change appropriately.

2.11.3 Paging for entity lists added

Many locations of Zabbix frontend present information as lists - whether it's a list of hosts, items or triggers. These lists can get quite long on large installations of Zabbix, and that slows down frontend considerably.

Zabbix 1.8 supports splitting long lists in multiple pages. Entry count per page is configurable in user's profile.

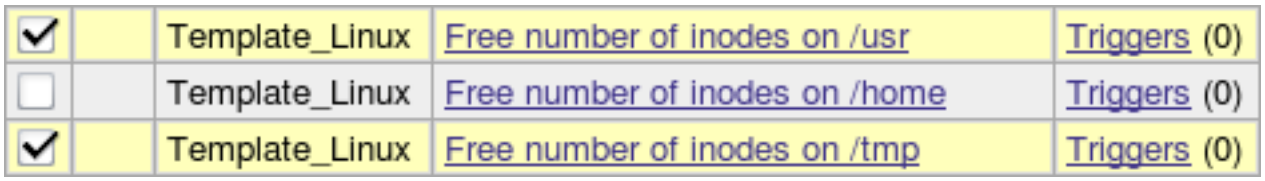
|<| |<| |-|

|<| |<| |-|

2.11.4 Selected rows are highlighted now

Most of the entries in these lists can be selected for performing some operation on them.

A minor but welcome improvement in 1.8 - selected rows now are hilighted.




2.11.5 Ability to display server name

Setting variable ZBX_SERVER_NAME allows Zabbix server name to be displayed in the frontend upper right corner.

It is also used in page title.

2.11.6 More flexible linked items

Zabbix supports very powerful templating that makes large scale configuration management easy. Templates can be linked to monitored hosts and they determine what and how is monitored.

Downstream linked items in Zabbix 1.8 are more flexible now - for example, it is possible to edit SNMP parameters like community string, or allowed hosts for trapper items, in items that are linked in from templates.

2.11.7 IP address becomes default option

In host creation form, IP address is now the default choice.

2.11.8 Debug mode added for frontend

Mostly useful for developers, but can be handy when trying to determine source of a problem for others as well.

```
Time:0.000403 SQL: SELECT title1, url1, title2, url2, title3, url3, ti
Time:0.00066 SQL: UPDATE profiles SET value_id=0, value_int=
Time:0.00055 SQL: UPDATE profiles SET value_id=0, value_int=
Time:0.000563 SQL: UPDATE profiles SET value_id=0, value_int
Time:0.010292 LONG SQL: UPDATE profiles SET value_id=0,
```

Debug mode can be enabled on user group basis.

2.11.9 Help icons lead to online manual

Oldtime Zabbix users might remember the days when help icons from Zabbix frontend linked to the online manual. With the conversion to online documentation that again is possible, and in Zabbix 1.8 most of these icons open Zabbix manual in a new browser window or tab.

**3 API** In version 1.8 first, Zabbix provides JSON-RPC API. It already allows to perform most of the configuration changes, thus enabling powerful means for automated or complex setup management.

While API itself might not be that exciting for casual users, it enables creation of various tools. One such tool already has been created - Zabbix commandline utility or Zabcon.

```
 +> get user show=userid,alias,refresh limit=2
User result set
+--------+-------+---------+
| userid | alias | refresh |
+--------+-------+---------+
| 1      | Admin | 0       |
| 2      | guest | 30      |
+--------+-------+---------+
2 rows total
 +>
```

Zabcon is especially exciting for users who would like to perform uncommon, large scale changes, as it allows easy scripting without programming skills.

Zabbix management from servers without GUI installed also is expected to be possible, and surely users will come up with innovative and impressing uses for this tool.

> **Attention:**
> Note: API is currently considered to be in a technical preview state and can change in next versions.

**4 Improvements for larger installations** Zabbix is being used in larger and larger environments every day. 1.8 release introduces several changes that are specifically useful in average and above setups.

**4.1 Performance improvements** When monitoring hundreds and thousands of devices, load on the monitoring server hardware can become a serious issue. Zabbix 1.8 brings many different improvements to the performance in several key areas.

4.1.1 Increased Zabbix server and proxy performance

Doing the main work behind the scenes, Zabbix server has been improved greatly to allow gathering more data on the same hardware. As Zabbix proxy shares some code with the server, it has benefited from these changes as well.

4.1.1.1 Much more efficient polling

Hundreds of pollers can be executed for greater parallelism and performance. The pollers do not communicate directly with the database and use very little of server CPU and memory resources.

4.1.1.2 Added configuration data cache module

Database cache is enabled by default and can not be disabled. Configuration option **StartDBSyncers** has been removed from Zabbix server and proxy configuration files.

A special internal check has been introduced, **zabbix[wcache,*]** to monitor health of this cache.

4.1.2 Frontend images recompressed with pngcrush

To reduce amount of data that users have to retrieve from webserver when using Zabbix frontend, all PNG images have been recompressed for optimal size.

4.1.3 Items with SNMP dynamic index use one connection

Zabbix supports monitoring SNMP metrics that have dynamic identifiers.

In version 1.8 index resolving and data retrieval is performed in a single connection now, reducing network load and load on the monitored devices.

**4.2 Automated host management improvements** For larger or constantly changing environments replicating these changes in Zabbix configuration can be a challenge. Zabbix already supported network discovery, and 1.8 now brings both improvements to the network discovery, as well as new methods to automate adding of new devices to monitor.

4.2.1 Network discovery improvements

Network discovery, available in previous versions, has received multiple improvements.

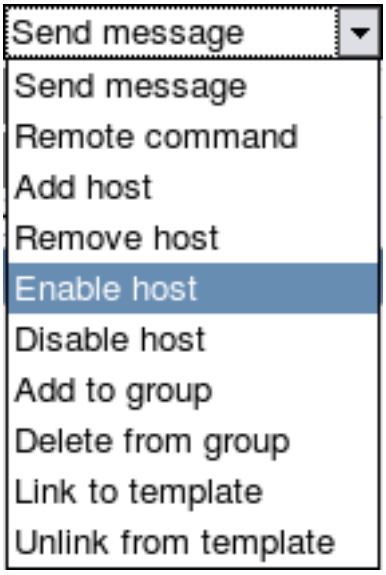
4.2.1.1 Port interval support

For services where it's appropriate, port interval support has been added.

4.2.1.2 New action operations

Based on discovery events, two new operations have been added, thus increasing available operation count for network discovery to 10.

- Enable host
- Disable host




4.2.1.3 IP mask support

Industry standard network mask notation can now be used in network discovery, for example, **192.168.1.0/24**.

4.2.1.4 Support for multihomed devices

If a host provides some service on multiple IP addresses, it would be discovered as multiple hosts in Zabbix versions before 1.8. Starting with 1.8, it is possible to use different uniqueness criteria for some services, for example **system.uname** returned by Zabbix agent or any OID returned by SNMP capable host.

4.2.1.5 SNMPv3 support

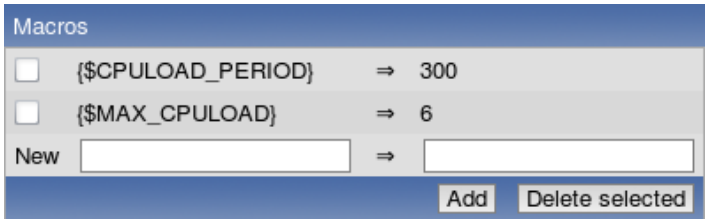
Network discovery in Zabbix 1.8 supports SNMPv3 with all the corresponding functionality.

4.2.2 Auto registration for active agents

Completely new in Zabbix 1.8, it is possible to allow active Zabbix agent auto-registration, after which server can start monitoring them. This allows to add new hosts for monitoring without any manual server configuration for each individual host.

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 collection of performance and availability data of the host.

**4.3 Support for global, template and host level macros** Support for user definable macros (or variables) has been added. These can be defined globally for Zabbix installation, on template and host level. For example, defining the following macros on a host level allows to set custom thresholds per host, even if all hosts are linked against single template:




In this case, templated trigger expression would be:

```
{ca_001:system.cpu.load[,avg1].min({$CPULOAD_PERIOD})}>{$MAX_CPULOAD}
```

**5 Various**

**5.1 Host maintenance** Host and host group maintenance has been added to Zabbix.

Hosts that are in maintenance are indicated as such in Monitoring → Triggers view.

| | | | | |
|---|---|---|---|---|
| Warning | PROBLEM | 16 Nov 2009 17:19:36 | A Test Host [Normal maintenance] | Severity 1 missing on A Test Host |
| Average | PROBLEM | 16 Nov 2009 13:21:57 | Another Host [No data maintenance] | SMTP service is down |

If a problem happens during the maintenance and is not resolved, notification is generated after maintenance period ends.

If a log item is added while host is in maintenance and maintenance ends, only new logfile entries since the end of the maintenance will be gathered.

**5.2 Improved audit log** Zabbix provides accontability by recording all user logins and changes to the Zabbix configuration in the audit log.

Audit log in 1.8 has been improved, and instead of simply seeing that something has changed, many entities will have exact changes recorded.

|<| |<| |-|

|<| |<| |-|

**5.3 New macros** Zabbix provides very useful variables - called macros - to be used in item names, notifications and elsewhere. Zabbix 1.8 increases the amount of available macros, as well as making some macros usable in more locations.

5.3.1 In notifications

Along with existing macros new ones can be used in notifications that are sent out.

- {NODE.*[1..9]}
- {ITEM.LOG.*[1..9]}
- {ITEM.VALUE} and {ITEM.VALUE[1..9]}
- {ITEM.LASTVALUE[1..9]}
- {HOST.CONN[1..9]} {HOST.DNS[1..9]} {IPADDRESS[1..9]}
- {TRIGGER.KEY[1..9]}
- {HOSTNAME[1..9]}
- {ITEM.NAME[1..9]}
- {PROFILE.*[1..9]}
- {EVENT.ACK.STATUS}
- {EVENT.ACK.HISTORY}
- {TRIGGER.EVENTS.UNACK}

5.3.2 In map labels

Map labels allow using handy macros like the current value of some item.

- {TRIGGERS.UNACK}

**5.4 Advanced regular expression editor** Advanced regular expression editor was added to Zabbix with ability to test regular expressions. It is now possible to define complex regular expression with easy to use interface and reuse them with simple reference.

**5.5 IPv6 support for SNMP monitoring** In addition to the SNMP related improvements for network discovery, IPv6 support has been implemented for SNMP monitoring.

**5.6 Supported PHP version changes** Zabbix frontend is based on PHP. Since the last stable Zabbix release there have been major changes in PHP versions, and Zabbix frontend has been changed accordingly.

5.6.1 Support for PHP 5.3 added

Released in 2009.06.30, PHP 5.3.0 was out for some time to require support of Zabbix frontend.

5.6.2 Support for PHP 4 dropped

Last bugfix release in 2008.01.03 and last release with security fixes in 2008.08.07, PHP4 was not receiving bugfixes anymore.

Zabbix 1.8 requires PHP 5.0 or later.

**6 Minor improvements** We call these minor, but for somebody one of these might be the biggest change in Zabbix 1.8.

**6.1 Basic authentication support in web monitoring** Web monitoring now supports basic HTTP authentication. It can be configured per scenario.

**6.2 New and improved monitored metrics** While Zabbix can be extended, built-in checks require less resources and are easier to use. Zabbix 1.8 introduces several new checks and improves old ones.

6.2.1 New items

Several completely new items have appeared.

- icmppingloss
- net.tcp.dns.query

6.2.1.1 CPU switches support on Linux

Key **system.cpu.switches** can be used for Linux hosts.

6.2.1.2 Added Windows services key

Added services key for Windows which can return services in a particular state.

6.2.2 ICMP items have new parameters

Zabbix ICMP items now are much more flexible. Item **icmpping** has gained the following parameters:

- target - host IP or DNS name;
- count - number of packets;
- interval - time between successive packets in milliseconds;
- size - packet size in bytes;
- timeout - timeout in milliseconds.

Now it is possible to use a key like this:

`icmpping[10.10.10.10,5,300,128,100]`

This would send five 128 byte packets to host with IP *10.10.10.10* with 300 ms interval between them, and use 100 ms timeout.

Item **icmppingsec** has gained all the above parameters, and one additional:

- mode - one of min, max, avg.

Default mode is *avg*.

6.2.3 'maxlines' parameter for log items

Item keys **log** and **eventlog** now have new parameter - **maxlines**. It specifies maximum number of new lines per second the agent will send to Zabbix server or Proxy.

By default, Zabbix agent does not send more than 100 log lines per second per log file. For fast growing file the number can be increased using the new parameter.

6.2.4 New Windows eventlog filters

Windows eventlog entries now can be filtered by type, source and event ID on the agent side.

6.2.5 SSH and telnet checks

Now SSH and telnet can be used for direct, agent-less monitoring. SSH supports both password and key authentication methods.

It makes possible very effective remote monitoring of network devices, appliances and servers without use of Zabbix Agent.

Currently SSH and telnet cannot be used in actions, this functionality will be available in future releases.

6.2.6 LVM swap devices support

LVM devices are now supported as swap devices on Linux.

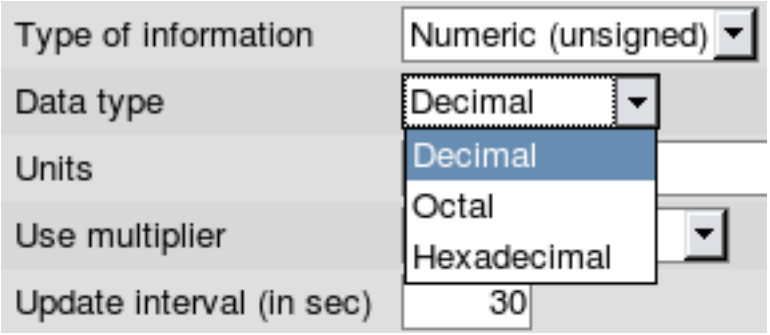
6.2.7 First CPU number changed on Linux

First CPU on Linux is now referred to as 0, which is consistent with other operating systems.

6.2.8 Positive sign for decimal values supported

If incoming decimal (float) value is preceded by a **+** sign, it is supported as a positive number now.

**6.3 New input data types** While different base values could be monitored before with user parameters, that was not easy enough. Zabbix 1.8 natively supports two new input data types, sometimes found on devices like printers.




- Octal
- Hexadecimal

**6.4 Client utilities moved to bin** Zabbix client utilities **zabbix_get** and **zabbix_sender** were moved from *sbin* to *bin*.

**6.5 Improved sample configuration files** Sample configuration file layout was changed. Now all parameters are included, and their default values, as well as allowed ranges, are documented.

```
### Option: ListenPort
#       Listen port for trapper.
#
# Mandatory: no
# Range: 1024-32767
# Default:
# ListenPort=10051
```

**6.6 Added manpages** Manpages for all Zabbix processes have been added.

**6.7 Notification media can be chosen in action operations** It means that it’s possible to define messages that will be sent to one or several media only. For example, all critical messages can be delivered by using SMS messaging, while other messages using both email and SMS without creating multiple actions.

**6.8 Timestamp support for zabbix_sender** Support for **-T** parameter in **zabbix_sender** has been added, which allows to set timestamp for each value. The option can be used to migrate older historical data from other monitoring tools to Zabbix database for graphing and long-term analysis.

**6.9 Manual user unblocking** Ability to manually unblock users who have been locked out by bruteforce protection was added.

| | | | | | |
|---|---|---|---|---|---|
| Jānis | Bērziņš | Zabbix User | Database administrators | No | Blocked |

Clicking on the **Blocked** link will unblock this user.

**6.10 Native support of Oracle** Previous version of Zabbix had a dependency on a third party library called sqlora8. The library is not actively developed any more. Oracle support is now implemented using native Oracle Call Interface, which greatly improves performance and stability of Zabbix setups using Oracle as a back-end database.

**6.11 Host status propagation from proxies** If a host is monitored by a proxy, status of the host will be correctly displayed and updated in Zabbix front-end.

**6.12 Rotated logfile monitoring** Zabbix supports logfile monitoring, and version 1.8 improves it further. If an application is writing to a new logfile with varying name - for example, if logfile name includes date - it is much easier to monitor with Zabbix 1.8, as it is now possible to specify regular expressions in logfile monitoring.

**6.13 Online documentation** Documentation from an inflexible PDF file has been converted to online format where anybody can comment on individual pages. Offline documentation can still be obtained with ODT export functionality.

**6.14 Detailed availability information displaying** Instead of simply displaying generic host status, in 1.8 frontend three different monitoring methods have their status displayed - Zabbix agent, SNMP and IPMI.


Errors related to each method are stored separately and can be obtained by moving mouse cursor over the error icon.

> **Note:**
> Default theme uses green to denote availability, while Black&blue theme uses blue colour.

**What's new in further releases** See detailed information on new features and significant changes in other 1.8 series releases:

- 1.8.1
- 1.8.2
- 1.8.3
- 1.8.4
- 1.8.5
- 1.8.6

**Installation and upgrade** See the installation section for new installations.

See upgrading section if upgrading from an older Zabbix release.

Before upgrading, read important notes.

### 5 What's new in Zabbix 1.8.1

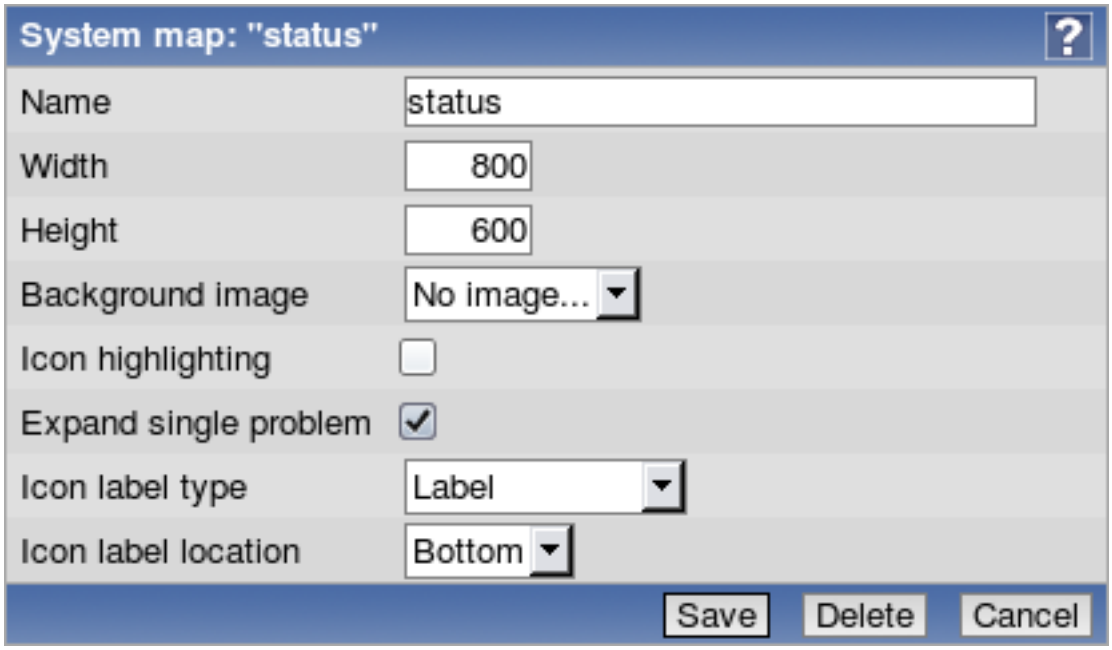
**5.1 Calculated items** Zabbix 1.8.1 adds support for a new item type - calculated items. These allow to reuse data from other items, making all kinds of calculations in the process.

**5.2 New and changed items**

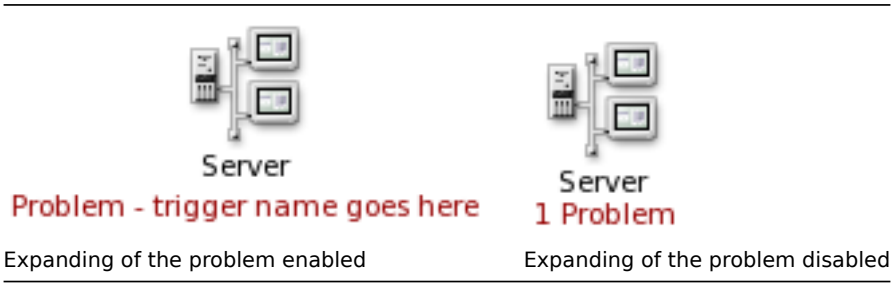
- added support of **system.stat[]** under AIX;
- added support of **net.if.*** under Windows;
- added support of **net.if.list** on Windows;
- added support of **kernel.maxproc[]** under Linux 2.6;
- added possibility to exclude some services from the result of Windows key **services[]**.

**5.3 Frontend improvements** 5.3.1 Single problem handling in maps

There's now an option for each map, controlling single problem displaying. If it is marked, previous behaviour is used - single problem has trigger name displayed. If it is disabled, single problem is listed as "1 Problem"



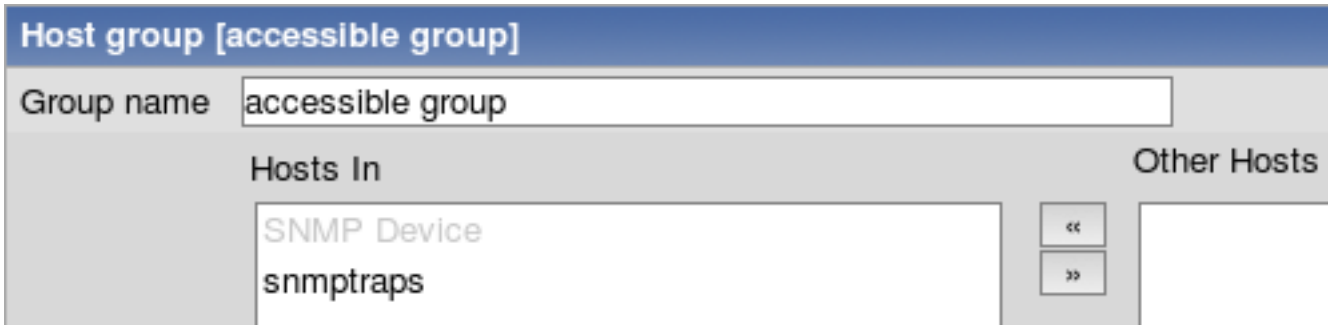

And this is the effect it has on maps:




Expanding of the problem enabled | Expanding of the problem disabled

5.3.2 Read-only hosts better represented

Hosts that user does not have write permissions to (but has read permissions) are disabled in hostgroup properties and can not be operated by the user. Previously these hosts were not visible at all.




5.3.3 Host status widget in dashboard

A new widget has been added to dashboard - host status. It shows host groups and how many hosts in each group have at least one problem. For those that have, field is coloured according to the trigger with highest severity.

Host status

| Host group | Without problems | With problems | Total |
|---|---|---|---|
| accessible group | 1 | 0 | 1 |
| Important SNMP Hosts | 1 | 0 | 1 |
| Linux servers | 4 | 1 | 5 |
| none | 4 | 0 | 4 |
| SNMP Devices | 1 | 0 | 1 |
| Test Group | 1 | 1 | 2 |
| Web pages | 1 | 0 | 1 |

Updated: 19:05:47

5.3.4 Changes to Zabbix status reporting

As of version 1.8.1, ”Status of Zabbix” dashboard widget and report is only available to users of Zabbix Superadmin type. Additionally, this report/widget shows any PHP installation or configuration problems found.

Status of Zabbix

| Parameter | Value | Details |
|---|---|---|
| Zabbix server is running | Yes | - |
| Number of hosts (monitored/not monitored/templates) | 96 | 25 / 2 / 69 |
| Number of items (monitored/disabled/not supported) | 223 | 89 / 126 / 8 |
| Number of triggers (enabled/disabled)[true/unknown/false] | 32 | 27 / 5 [0 / 17 / 10] |
| Number of users (online) | 9 | 1 |
| Required server performance, new values per second | 3.1127 | - |
| PHP memory limit | 64M | 128M is a minimal PHP memory limitation |
| PHP max execution time | 250 | 300 sec is a minimal limitation on execution time of PHP scripts |
| PHP MB string overload | no | MB String overload PHP is not set. Please set "mbstring.func_overload" to 2 in php.ini. |

Updated: 18:51:48

5.3.5 Item colouring in trigger editing

Trigger editing now colours items according to their status - green for enabled, red for disabled and grey for unsupported. This simple change should make identifying any problems with triggers much easier.

| Name | Expression |
|---|---|
| Coloured trigger | {A Test Host 2:agent.ping.**last**(0)}=0 & {A Test Host 2:as.**last**(0)}=1 & {A Test Host 2:sds.**last**(0)}=2 |

5.3.6 Unacknowledged event filter

New filter option added for Monitoring → Triggers view - ”Show triggers with unacknowledged events”. This option hides triggers that have all their events acknowledged.

5.3.7 Updated translations

Translations for the following languages have been updated:

- Russian;
- Japanese;
- French.

### 5.4 Other changes

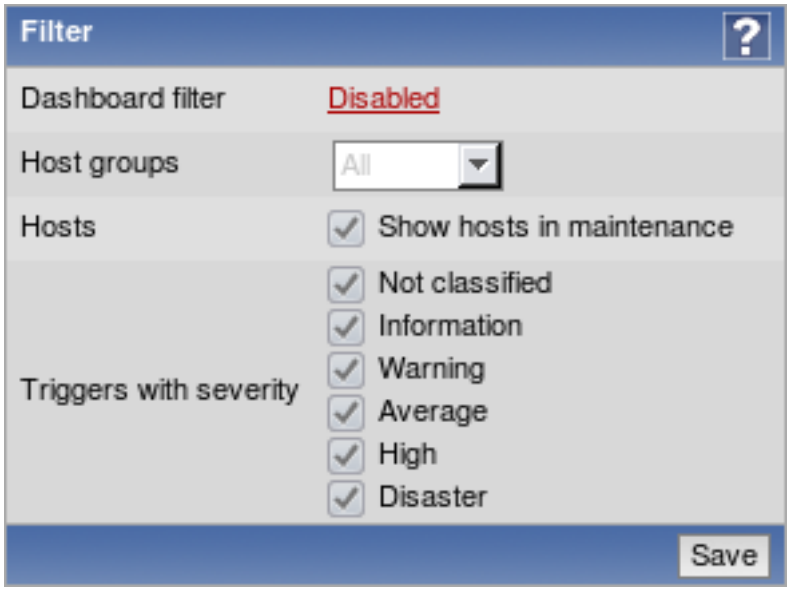
- Database index fixed, improving node synchronisation a lot. See release notes for upgrade instruction. Fixed index is used by default in new installations.
- API version changed to 1.1.

## 6 What’s new in Zabbix 1.8.2

**6.1 Frontend changes** 6.1.1 Dashboard filtering

Zabbix dashboard now can be filtered. Positioned at the upper right corner, just next to the fullscreen button, is the configuration button - .

After pressing this button, filter configuration is revealed. By default filtering is disabled and none of the options is available.

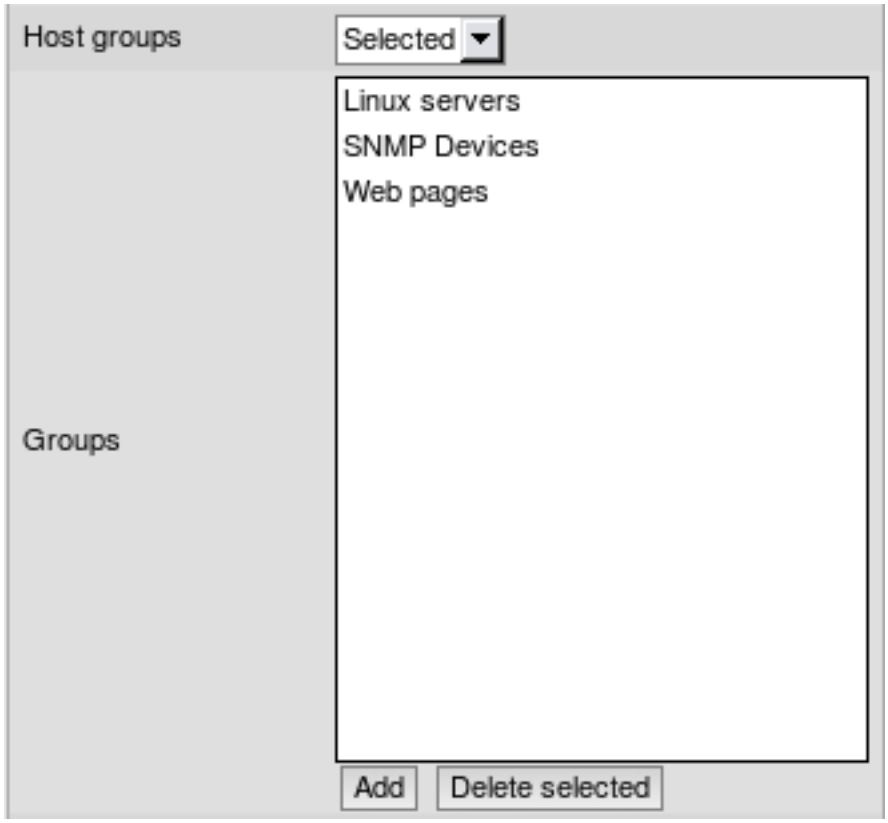



**Note:**
Filter can be customised per user account.

Clicking the **Disabled** control will enable the filter and allow to configure it. There are three available filtering categories.

6.1.1.1 Host group filter

This filter allows to choose which groups should be shown on the dashboard. By default, all host groups are shown. Choosing **Selected** in the dropdown allows to choose individual hostgroups to show.




This filtering affects **System status**, **Host status**, **Last 20 issues** and **Web monitoring** widgets.

**Attention:**
If host group filter is enabled, but no groups are selected, no data will be shown in the affected dashboard widgets.

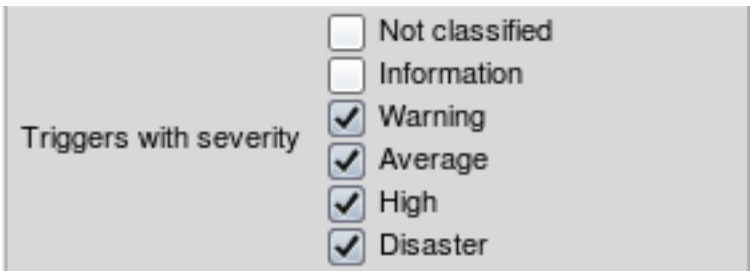
6.1.1.2 Maintenance filter

By default, all data for hosts that are in a maintenance is shown on the dashboard. Unchecking Show hosts in maintenance option will hide this information.


This filtering affects **System status**, **Host status**, **Last 20 issues** and **Web monitoring** widgets.

6.1.1.3 Trigger severity filter

Additionally, it is possible to filter shown data based on trigger severity.

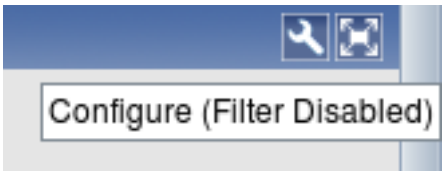



This filtering affects **System status**, **Host status** and **Last 20 issues** widgets. For **System status** widget, corresponding columns are hidden.

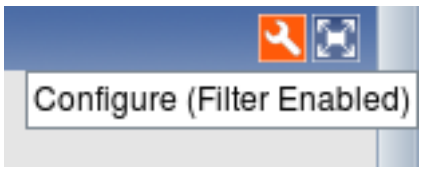
6.1.1.4 Filter indication

Clicking **Save** will return to the dashboard. To indicate an enabled filter, configuration button is highlighted -  .

Additionally, depending on filter state, button tooltip explains what it is indicating.




Dashboard filter disabled



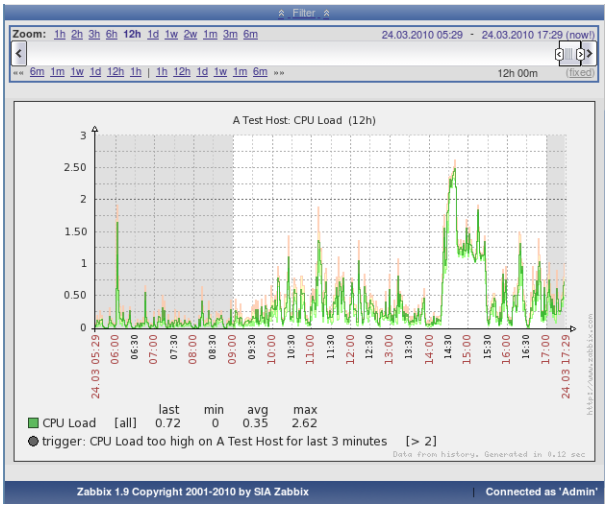

Dashboard filter enabled

6.1.2 Time period selector changes

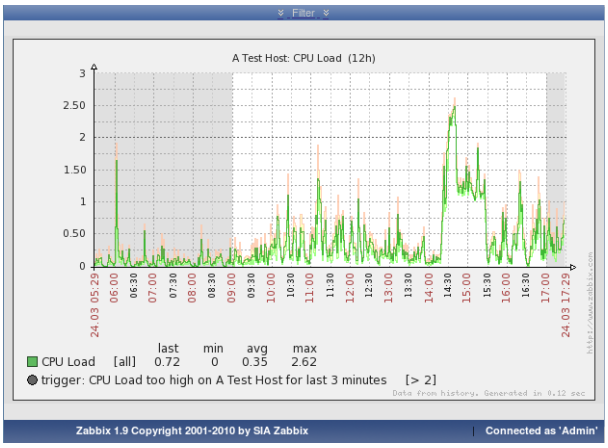
Starting with 1.8.2, there are two changes:

6.1.2.1 Time period selector moved to the top

Now time period scrollbar is part of the filter at the top of the page. In addition to being on top for all pages, it can be hidden by collapsing the filter.



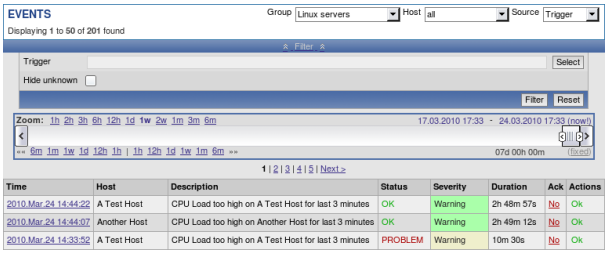

Expanded period scrollbar



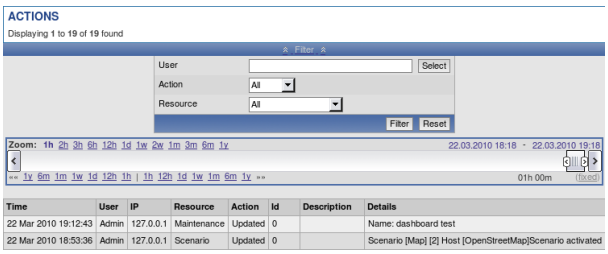

Collapsed period scrollbar

6.1.2.2 Improved events and auditlog time period selection

Instead of providing very limited ”Since” time filter, both event view and audit/action log now use the same standard time period scrollbar.




Period selector in events




Period selector in audit

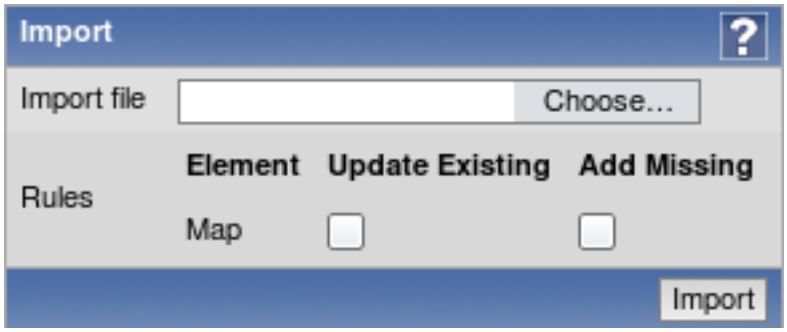
6.1.3 Map and screen exporting and importing

6.1.3.1 Map exporting and importing

Support for network map exporting and importing was added. Maps can now be exported from their configuration screen:

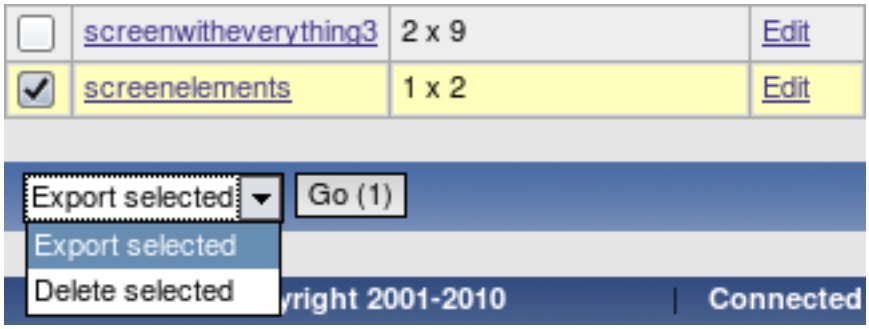

And on the same screen the import button can be found:


When importing, it is possible to choose whether overwrite existing maps and add new ones:

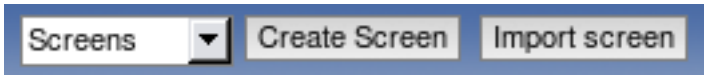



6.1.3.2 Screen exporting and importing

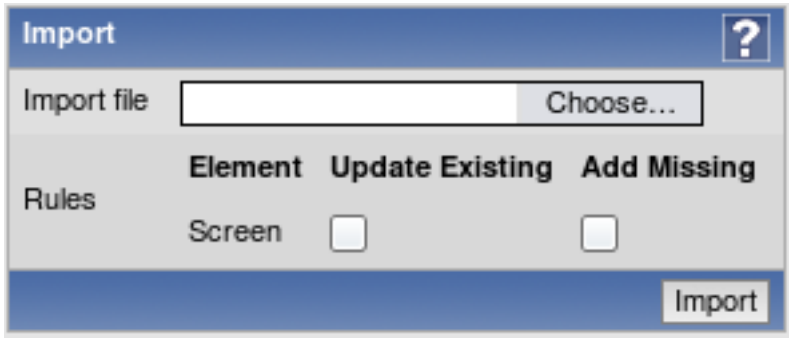
Support for screen exporting and importing was added. Screens can now be exported from their configuration screen:




And on the same screen the import button can be found:




When importing, it is possible to choose whether overwrite existing maps and add new ones:




6.1.4 More configurability for ”Status of triggers” screen element

Screen element ”Status of triggers” now is split in two new elements:

- Status of host triggers;
- Status of hostgroup triggers.

They allow to choose a host or host group to filter element contents on, respectively. If any is chosen, there are no host and hostgroup dropdowns. If none is chosen, element works as before - interactive hostgroup and host dropdowns allow to choose filter options in the fly.

6.1.5 Translation updates

The following new translations have been added:

- Ukrainian.

The following translations have been updated:

- Brazilian Portuguese;
- French;
- Russian.

6.1.6 Frontend requirement changes

- PHP parameter **max_input_time** is now required to be at least 300;

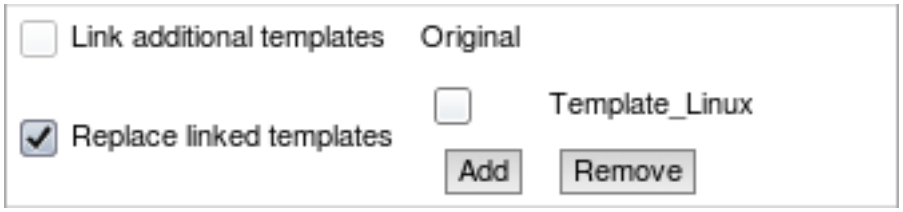
- PHP parameter **upload_max_filesize** must be at least 2MB;
- Parameter **mbstring.func_overload** not required anymore.

6.1.7 Miscellaneous frontend changes

- Host maintenance is now displayed in dashboard by colouring host name:


- When viewing host list in configuration, one more level of linked templates is now displayed in addition to the directly linked templates. For each directly linked template, a list in parenthesis shows templates that template is linked to itself.


- Frontend now always uses PHP timezone. It will not use browser time anymore. This should solve issues with mangled time, graphs working erroneously if browser timezone differs from PHP timezone and other issues.
- Added an ability to replace already linked templates when performing host mass update.




- API version increased to 1.2.

**6.2 Improved triggers** In Zabbix 1.8.2, some trigger functions and expressions have gained new features.

6.2.1 Time-shifting in triggers

Trigger functions **min**, **max**, **avg**, **last** and **count** now support additional parameter - **time_shift**. This parameter allows to match data from a specific period of time in past.

For example, **avg(3600,86400)** will return the average value for an hour one day ago. This way it is possible to compare average load today with average load of the same time yesterday:

```
{host:system.cpu.load.avg(3600)}/{host:system.cpu.load.avg(3600,86400)}>2
```

This expression would fire if average load for the last hour today exceeds average load of the same hour yesterday more than two times.

6.2.2 Additional suffixes in trigger expressions

Support for new suffixes has been added. New numeric suffixes:

- **T** - tera;

New time-related suffixes:

- **s** - seconds; when used, works the same as raw value;
- **m** - minutes;
- **h** - hours;
- **d** - days
- **w** - weeks.

These improvements allow to write expressions that are easier to understand and maintain, for example the following expressions:

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>120
{host:system.uptime[].last()}<86400
```

could be changed to:

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>2m
{host:system.uptime.last()}<1d
```

### 6.3 Improved items

- Telnet items now additionally support **%** as prompt termination character;
- Aggregate items now support:
  - Comma delimited list of host groups;
  - Item keys with doublequotes.
- Item key parameters now support arrays.

### 6.4 Other changes

- In previous versions, Zabbix server was able to reconnect to MySQL database if it temporarily became unavailable. With 1.8.2, this is also supported for PostgreSQL, Oracle and SQLite.
- Zabbix administrators now can only choose groups they are members of and users of those groups in action operations. Additionally, they do not have access to actions that have users or groups in their operations that administrator does not share a group with, or is not a member of, respectively.

6.4.1 New configuration parameters

- Zabbix server and proxy daemons now have new option - **LogSlowQueries**. Mostly helpful with performance debugging.
- Zabbix agent daemon has a new configuration parameter - **UnsafeUserParameters**. This allows to override security checks and pass all characters in user parameter arguments.
- Zabbix server daemon has a new configuration paramater - **MaxHousekeeperDelete**. Previously, housekeeper always removed 500 entries in one go. This amount can be customised now. If set to 0, no limit is applied.

6.4.2 Performance improvements

- Node synchronisation received further performance improvements;
- Various frontend sections use less memory and perform better.

6.4.3 More robust escalation module

Since version 1.8.2 any old escalations are removed for a particular trigger when adding new one. This should prevent multiple recovery messages (as they also use escalation module), neverending escalations because of misconfiguration and other problems.

6.4.4 Improved zabbix_sender

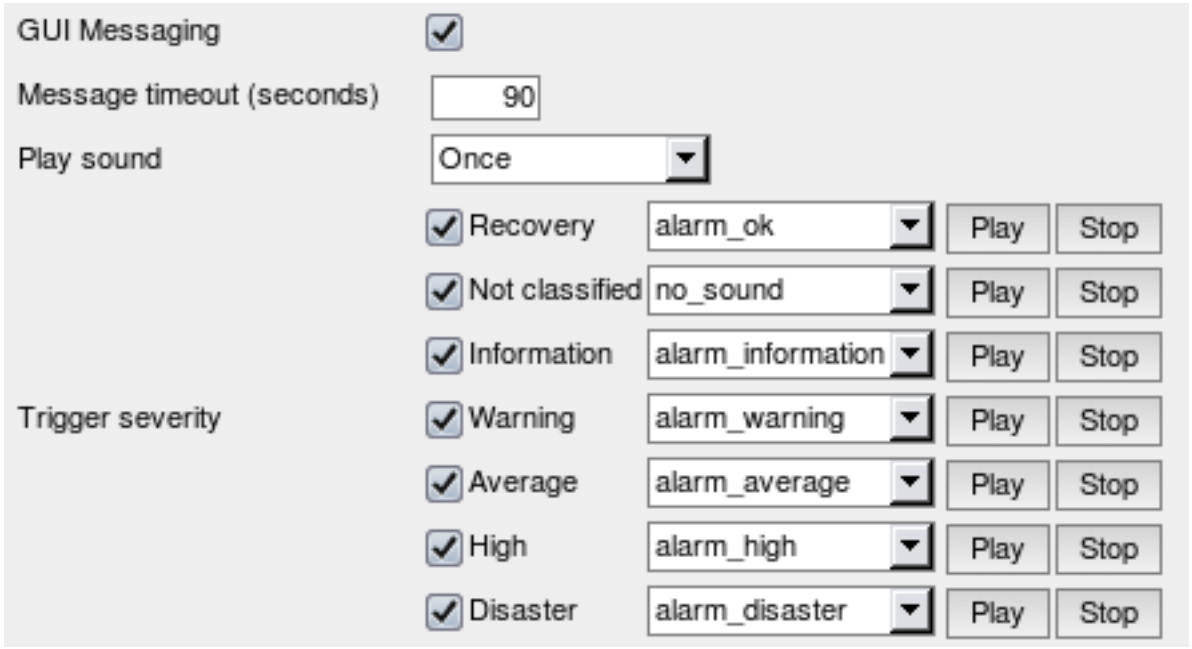
Zabbix sender has gained ability to get its input from standard input when specifying **-** as the input file. Additionally, **-r** flag makes it send new values as soon as they arrive, thus allowing to follow a file that gets information appended, or open a pipe and transmit its data as soon as it arrives.

## 7 What's new in Zabbix 1.8.3

### 7.1 Frontend changes 7.1.1 Global notifications

Previously, working in some other location than *Status of triggers* or *Dashboard* pages would not show any information regarding issues that are currently happening. Starting with 1.8.3, support for global frontend notifications is added. Global notifications involve both showing a message and playing a sound.

Global notifications can be enabled per user in profile configuration. If enabled, global message timeout can be changed. By default, messages will stay on screen for 90 seconds.

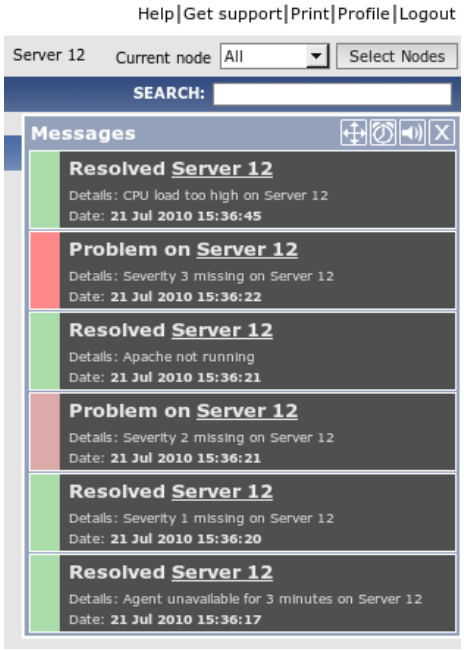



Additionally, it is possible to configure details of sound notifications. There are three possible options for playing a sound:

- 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.

It is possible to receive messages for problems and for resolutions. Messages can be filtered based on trigger severity as well. For each trigger severity and recovery message sound to be played can be customised.

As the messages arrive, they are displayed in a floating section on the right hand side. This section can be repositioned vertically.

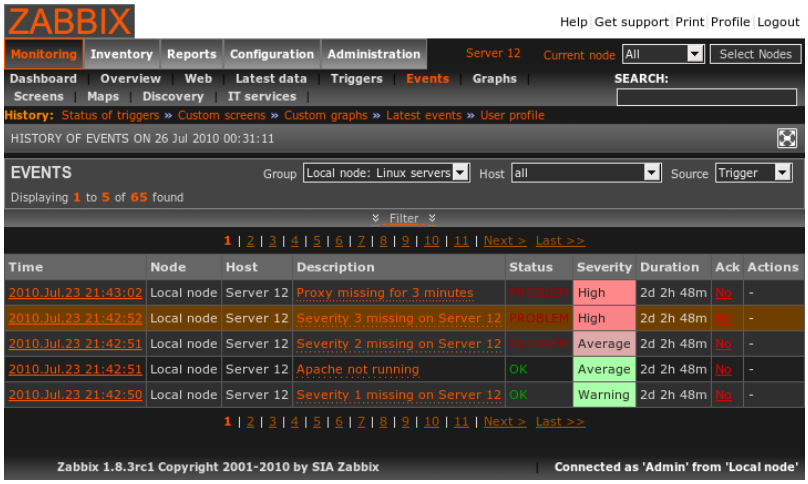



For this section, several controls are available:

-  **Move** button allows to reposition the section vertically. This can also be done by dragging section header;
-  **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.

7.1.2 New frontend theme

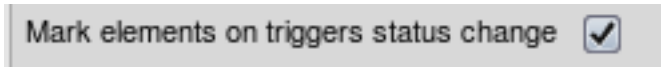
A new frontend theme has been added - *Dark orange*.




7.1.3 Network map improvements

7.1.3.1 Change marking in maps

A new per-map option has been introduced - ability to mark map elements that have their state changed recently.



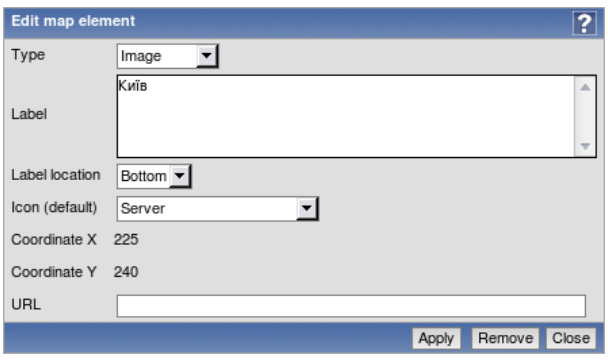

When this option is enabled, any map elements that recently had state changed (trigger firing or resolving) are marked by red triangles.


These triangles are placed on top, bottom, left and right of an object. If the object has a label, no triangle is displayed at the label location.

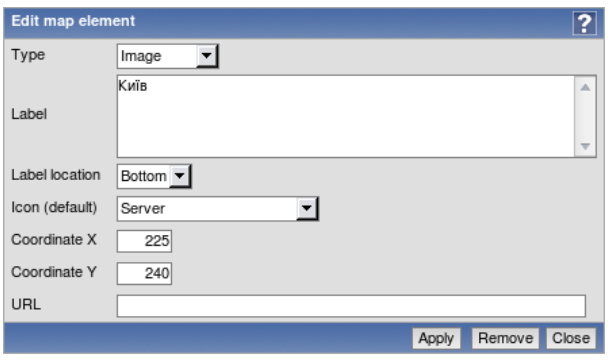
The period of showing these marks is the same as for trigger state flashing - 30 minutes.

7.1.3.2 Icon aligning in maps

In previous 1.8 branch versions of Zabbix it was not easy enough to place icons in exact locations and align them nicely - with the gain of drag and drop ability to enter element position manually was lost. Now this ability is restored.

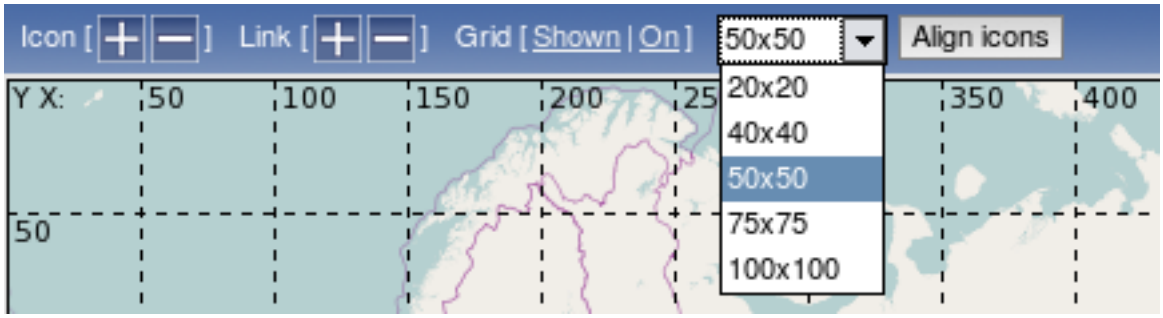



1.8.2 and before




1.8.3

An even bigger improvement - map icon aligning was added.



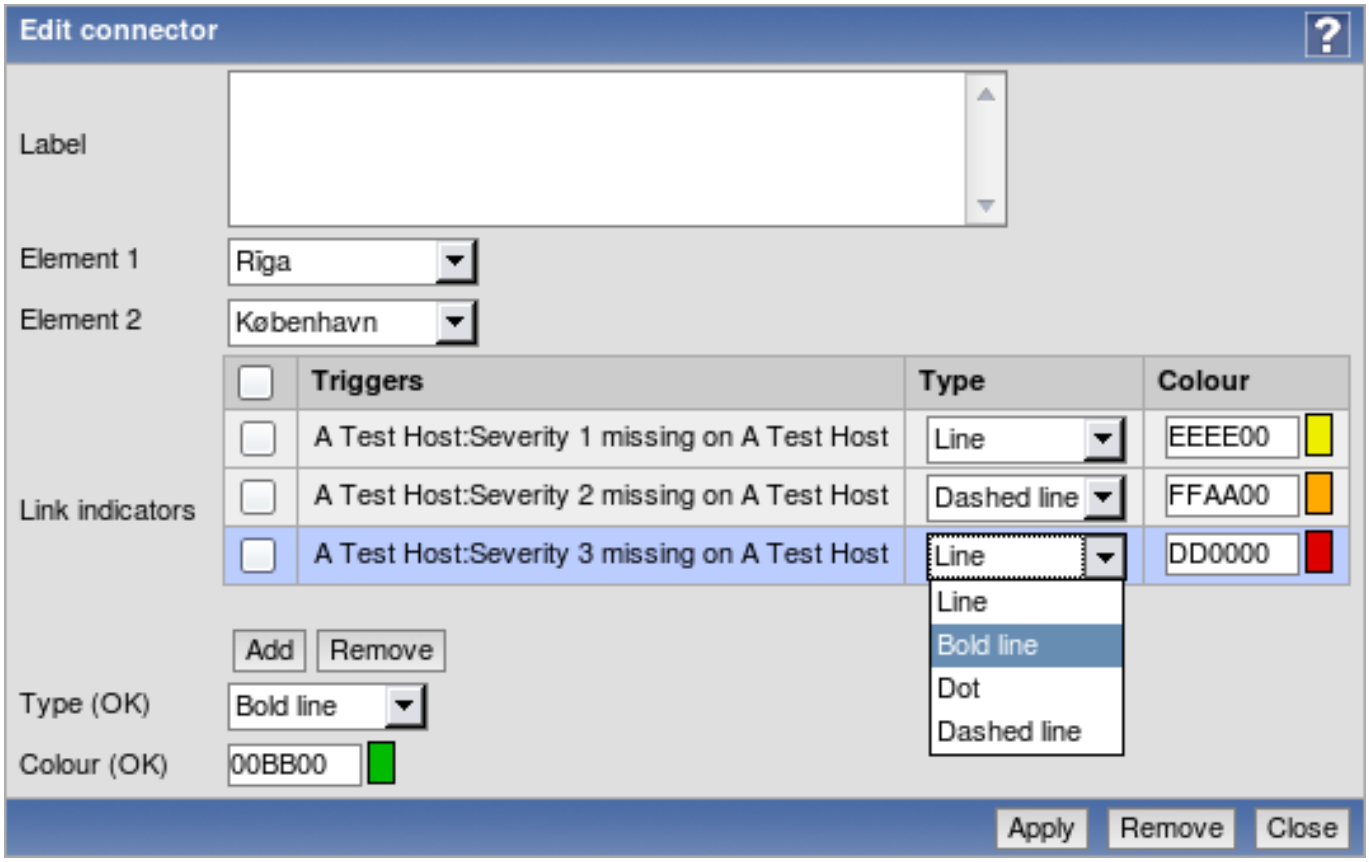

By default, grid for aligning is 50x50. Available grids:

- 20x20
- 40x40
- 50x50
- 75x75
- 100x100

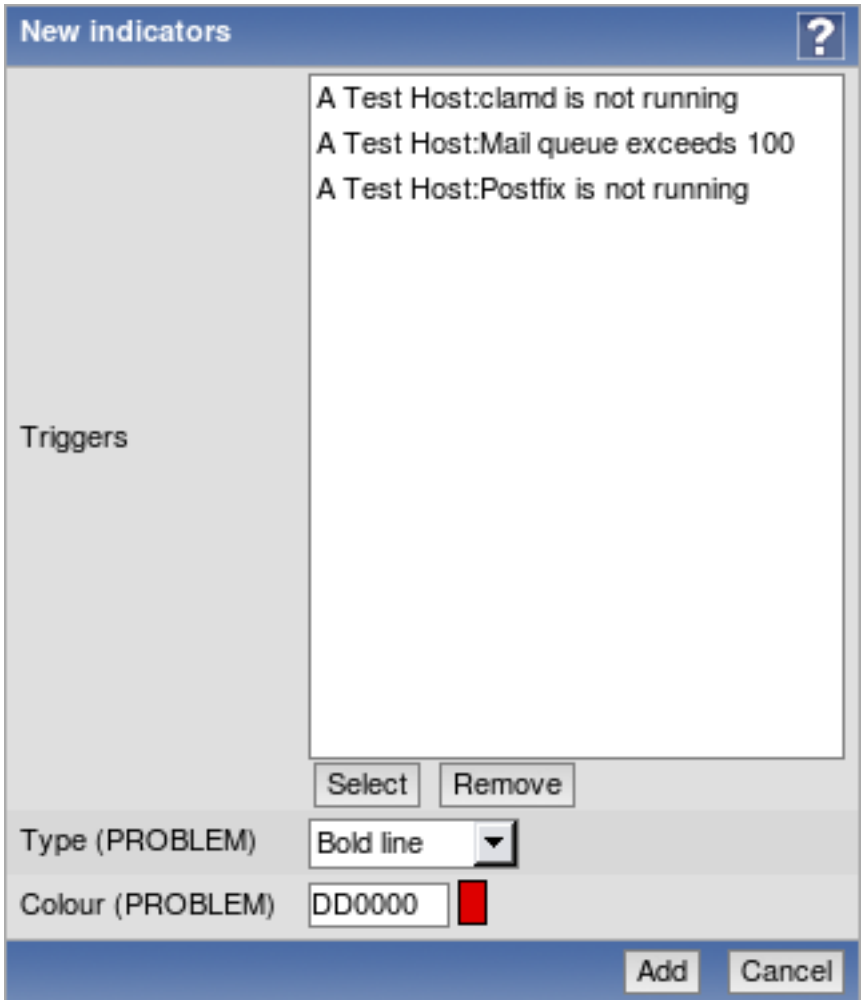
If aligning is on, moving an icon will align it at the center of the grid cell. Clicking button *Align icons* will align all icons to the nearest grid cell. It is now also possible to hide grid while editing the map.

7.1.3.3 Improved link status indicator editing in maps

It is now easier to edit status indicators for map links - line style and colour can be changed directly from the list:



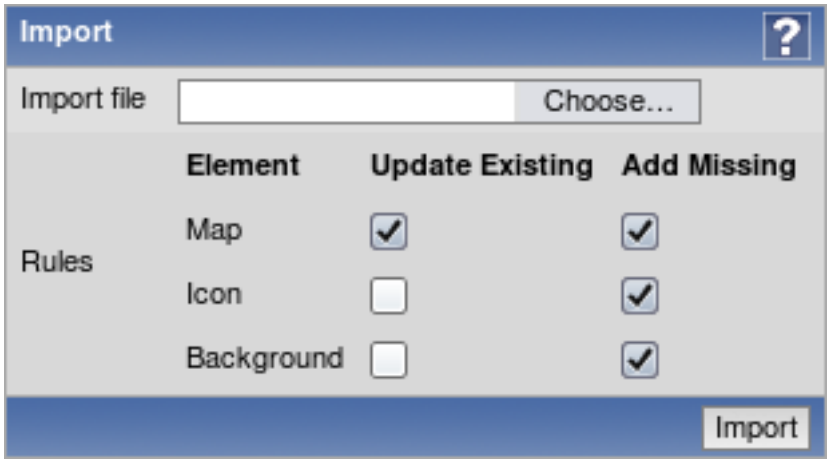

Additionally, it is now possible to add multiple indicators with same line style and colour in one go:




For the previous form, multiple triggers can be selected with checkboxes in the trigger list.

7.1.3.4 Image import and export

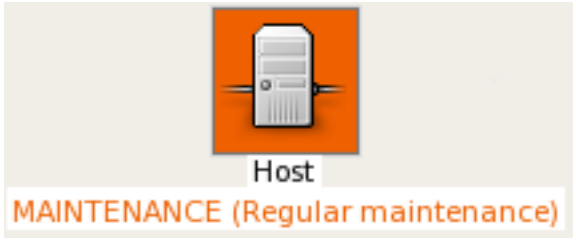
Available as a part of network map export and import, image export and import has been implemented. In the map import dialogue, there are checkboxes for icon or map background importing. This option is only available to Zabbix Super Administrators.




7.1.3.5 Map element highlighting changes

- Map element highlighting has been changed. Disabled elements in 1.8.3 have light grey background instead of dark red, and maintenance status background is slightly lighter.

| 1.8.2 and before | 1.8.3 |
|---|---|

| | | |
|---|---|---|
| Disabled |  |  |
| In maintenance |  |  |

- Now map element ”map” has same icon highlighting as the map element ”host group”.

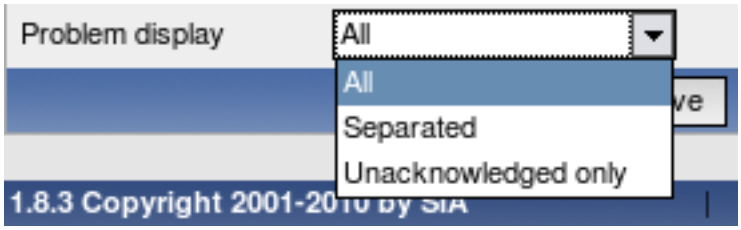
7.1.3.6 Other map improvements

- If multiple triggers with the same severity are assigned to the same map link, the one with the lowest ID takes precedence.

7.1.4 Unacknowledged trigger displaying in the dashboard

It is now possible to customise dashboard display depending on trigger acknowledgement state.

In the dashboard filter options, there is a new dropdown available, *Problem display*.




It provides 3 options:

- **All** - the default setting, which works just like before - displays all problems without any difference between acknowledged and unacknowledged ones.
- **Separated** - shows unacknowledged and all problems separately in the format **<Unacknowledged> of <All>**. Unacknowledged problems are displayed in bold, red font.
- **Unacknowledged only** - shows unacknowledged problems only. In this case they are also displayed in bold, red font to clarify that this option is being used.

The difference between these 3 options can be seen here:

| High | Average |
|---|---|
| 0 | 0 |
| 0 | 0 |
| 5 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 5 | 2 |

| Disaster | High | Average |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 3 of 5 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 0 | 3 of 5 | 2 of 2 |

| High | Average |
|---|---|
| 0 | 0 |
| 0 | 0 |
| 3 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 3 | 2 |

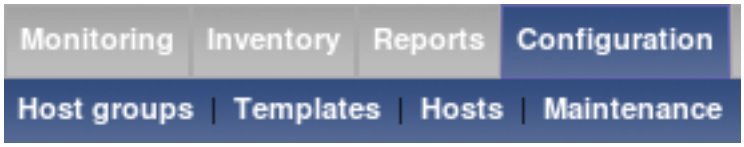
| | | |
|---|---|---|
| All | Separated | Unacknowledged only |

When in the *Separated* mode, tooltip for each number shows the corresponding issues - either the unacknowledged ones, or all of them.

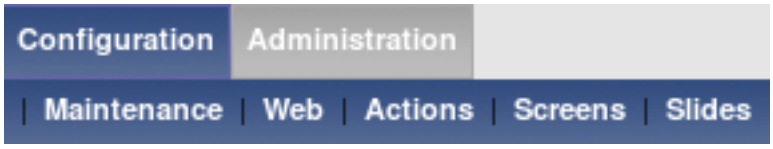
7.1.5 Reworked configuration section

Configuration section in 1.8.3 has been changed to improve usability.

- Template access has been brought out in the menu, as it was not that easy to find for new users. Additionally, last selected group is remembered separately for templates and for hosts now.




- Slideshow access has been brought out in the menu as a **Slides** entry.



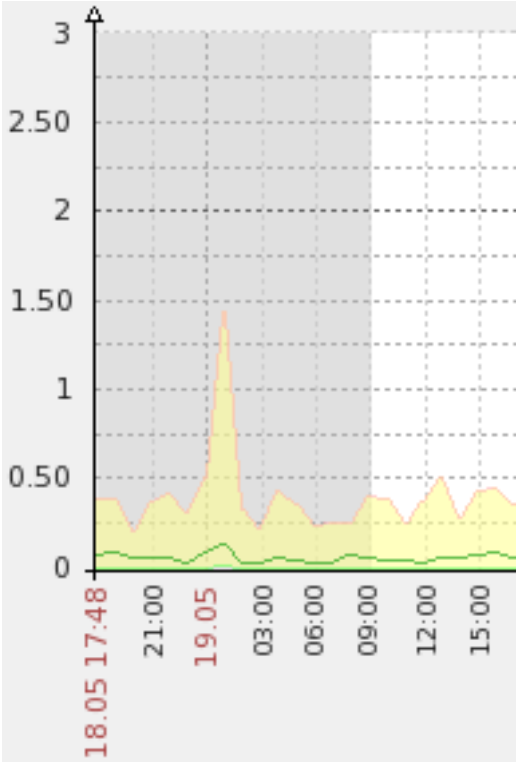

- Because of template availability from the main menu it was possible to remove dropdown in Configuration → Hosts section. Host elements like items, triggers etc can be accessed from host configuration section, and template section provides same access for templates.
- Configuration → Export/import section was removed, instead moving import and export controls in corresponding host and template pages in the same way as for network maps and screens. Export can be accessed by marking templates or hosts and choosing an action at the bottom of the list, and for import a button has been added at the top of the page.


7.1.6 More visibility for the trigger line

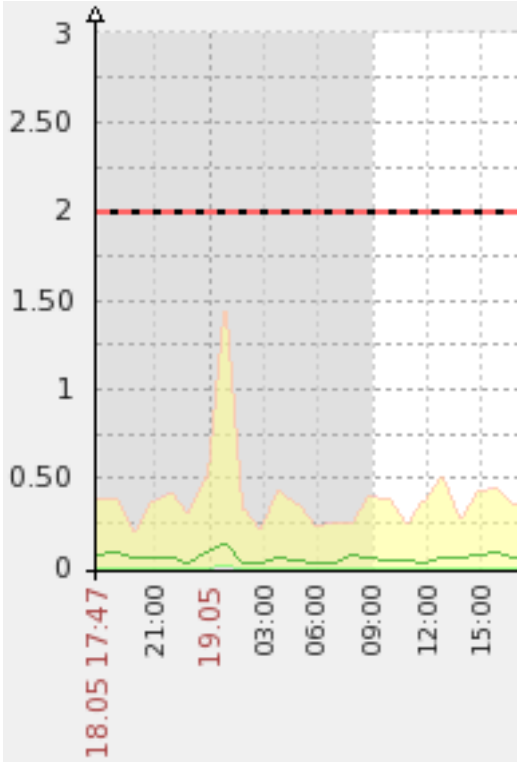
Trigger line in Zabbix graphs was extremely hard to spot in latest Zabbix versions. In 1.8.3, the line has been changed from 1 pixel height to 2 pixels. It also adds two additional changes:

- a 3 pixel long dashed black line is drawn on top of the trigger line;
- trigger line itself is coloured, according to the trigger severity (so it's light yellow for *Warning* severity and heavy red for *Disaster*).

These changes should change the trigger line from nearly invisible to very clearly visible - let's compare the 1.8.2 version and 1.8.3 one. In 1.8.2 trigger line is invisible unless the observer knows it's there.



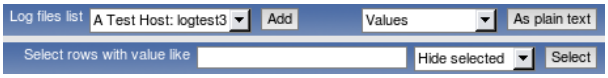

Trigger line in 1.8.2



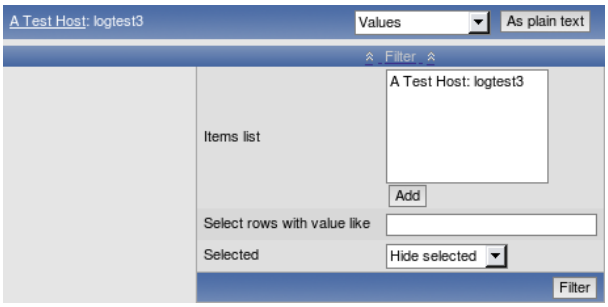

Trigger line in 1.8.3

7.1.7 Improved log viewing filter

Log viewing filter was brought in line with other filters in regard to being hideable. It was also redesigned, especially to make handling of multiple logfiles easier.




Log filter in 1.8.2 and before




Log filter in 1.8.3

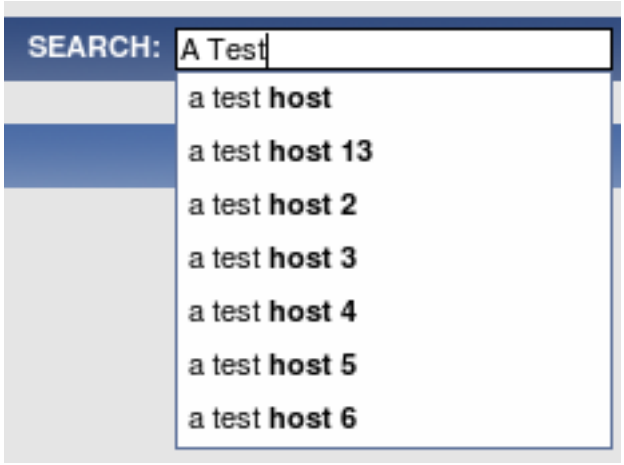
Additionally, this filter is also available now for text items, not for log items only.

7.1.8 Search improvements

Zabbix 1.8.3 has received two improvements to the global search functionality.

7.1.8.1 Search suggestions

Global search now provides suggestions, based on the entered string. This string is matched against hosts only and all suggestions are lowercase independent on the actual case of the hostname.

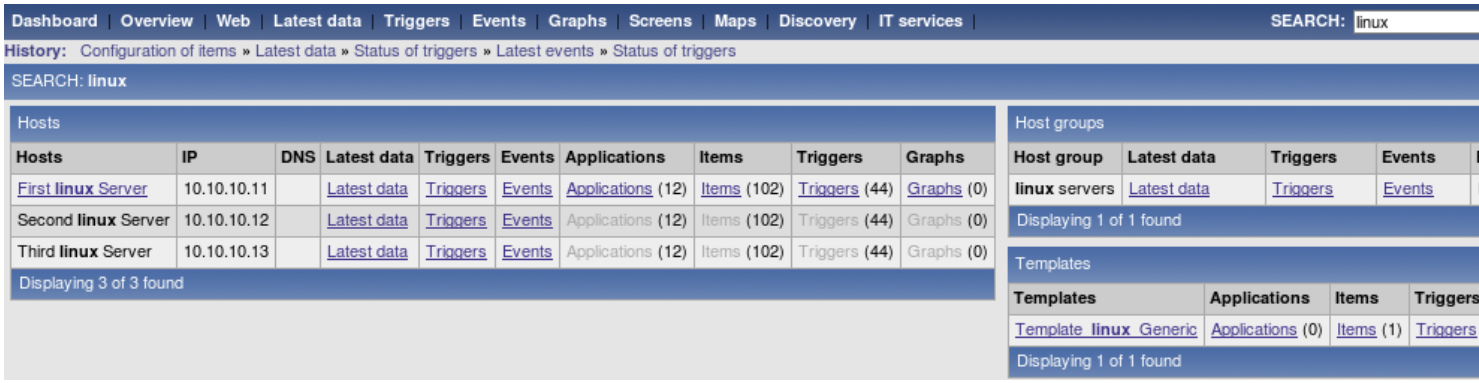



7.1.8.2 Improved search results

Search results form has been greatly improved.

- Instead of showing many ”Go” links, all links now list the actual target. That makes using them much easier.
- Instead of using a dropdown and ”Go” button, editing options for hosts now are shown in a table form, requiring less clicks to achieve the desired result.
- Count for all elements in host and template result boxes is shown.
- Links to application configuration have been added.
- Host and template names can be clicked now to access host or template properties directly.
- Disabled hosts are shown in red to provide more information.

While the new search results form is wider, it should be much more functional now:




In this screenshot, user has write access only to host *First Linux Server*, so editing options for other hosts are not available.

7.1.9 Unit blacklist created for items

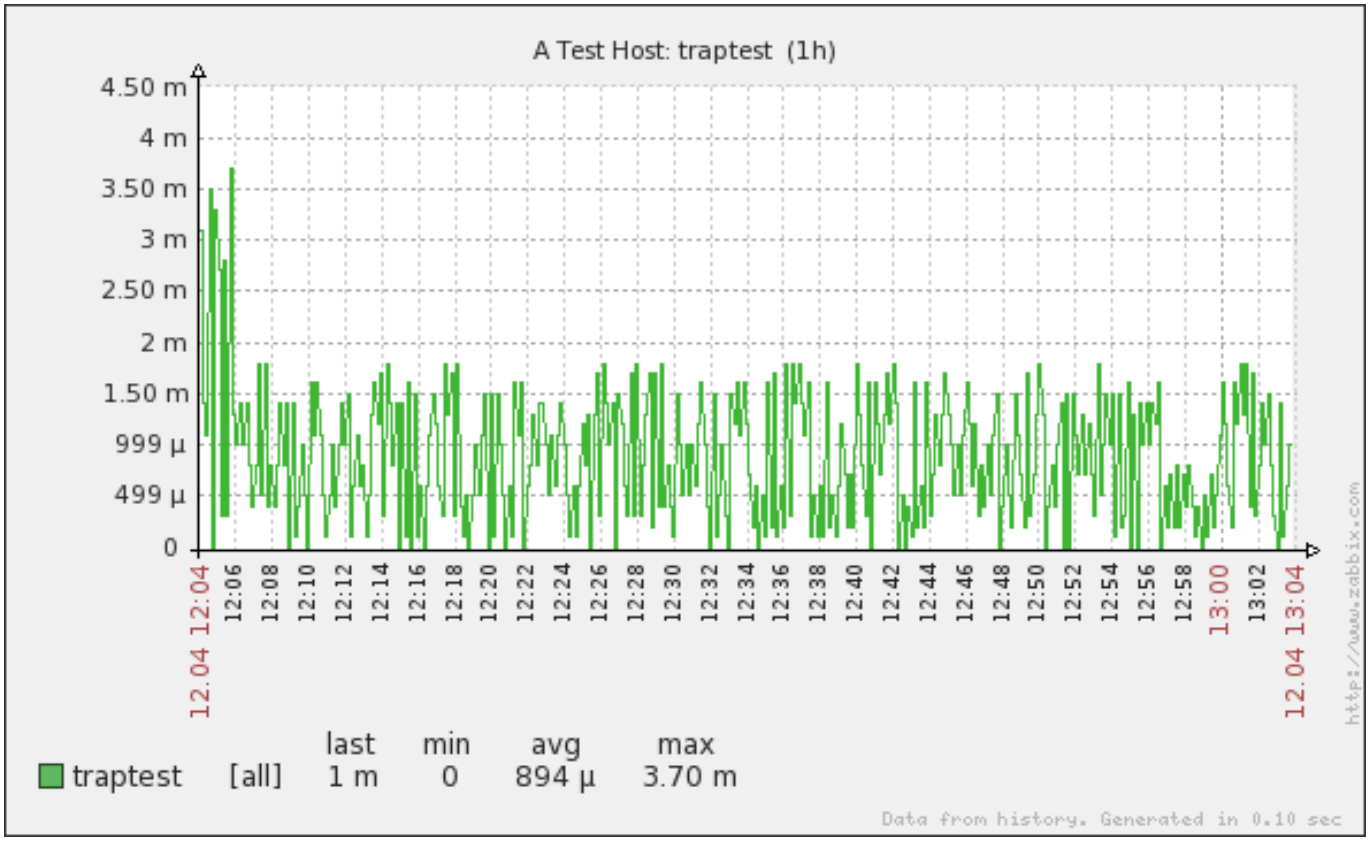
By default, specifying a unit for an item results in the multiplier prefix being added - for example, value ’2048’ with unit B would be displayed as 2KB. For a pre-defined, hard-coded list of units this is now prevented:

- %
- ms
- rpm

7.1.10 Support for more units

Support for the following units has been added for graph y axis:

- milli (m);
- micro (µ);
- nano (n).




**Attention:**
Note that smallest available units might limited by available database field precision.

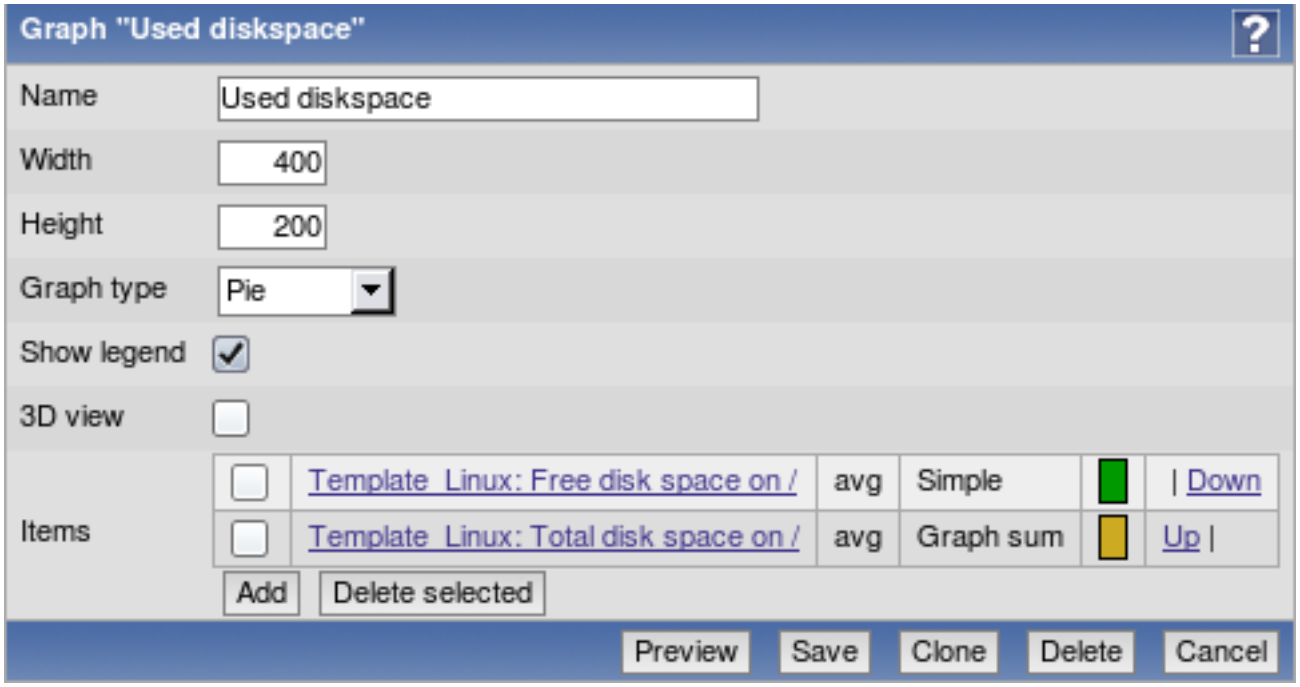
7.1.11 Ability to choose 'All' as a period

It is now possible to choose **All** as a period in graphs and elsewhere.


**Note:**
Maximal period currently is limited to two years.

7.1.12 Preview button in graph configuration

Previously, changing any parameter in the graph configuration would regenerate the graph. Now it only happens when really necessary, and button **Preview** has been added to force regenerating.




7.1.13 Supported macro changes

Macro **{TRIGGER.EVENTS.UNACK}** now is supported in map element labels. Additionally, these macros now expand to the following values:

| Macro | Value |
|---|---|
| {TRIGGER.EVENTS.UNACK} | Number of unacknowledged events for a map element in maps, or for the trigger which generated current event in notifications. |
| {TRIGGER.EVENTS.PROBLEM.UNACK} | Number of unacknowledged PROBLEM events for all triggers disregarding their state. |
| {TRIGGERS.UNACK} | Number of unacknowledged triggers for a map element, disregarding trigger state. |
| {TRIGGERS.PROBLEM.UNACK} | Number of unacknowledged PROBLEM triggers for a map element. |
| {TRIGGER.PROBLEM.EVENTS.PROBLEM.UNACK} | Number of unacknowledged PROBLEM events for triggers in PROBLEM state. |

The following new macros have been added:

| Macro | Value |
|---|---|
| {TRIGGER.EVENTS.ACK} | Number of acknowledged events for a map element in maps, or for the trigger which generated current event in notifications. |
| {TRIGGER.EVENTS.PROBLEM.ACK} | Number of acknowledged PROBLEM events for all triggers disregarding their state. |
| {TRIGGER.PROBLEM.EVENTS.PROBLEM.ACK} | Number of acknowledged PROBLEM events for triggers in PROBLEM state. |
| {TRIGGERS.ACK} | Number of acknowledged triggers for a map element, disregarding trigger state. |
| {TRIGGERS.PROBLEM.ACK} | Number of acknowledged PROBLEM triggers for a map element. |

7.1.14 Improved proxy view

Proxy view in Administration → DM → Proxies has been improved by adding two new columns - item count and required performance (showing same information as in the server status report, new values per second). That should help with proxy hardware requirement estimates.

| Name ▲ | Last seen (age) | Host count | Item count | Required performance (vps) | Hosts |
|---|---|---|---|---|---|
| a proxy | - | 4 | 7 | 0.23 | A Test Host 3, A Test Host 4, A Test Host 5, A Test Host 6 |
| b proxy | - | 1 | 1 | 0.03 | snmptraps |
| proxy | 4s | 1 | 30 | 0.67 | Another Host |

7.1.15 Item and trigger inheritance chain

If an item or a trigger is coming from a nested template chain, it isn't always obvious how exactly is it attached to the host. Since 1.8.3, in item or trigger editing form header full inheritance chain is shown, and it is possible to click on any template to see item editing form for that template.

|<| |<| |-|

|<| |<| |-|

7.1.16 Improved text item and logfile viewing

In 1.8.2 and before, text item history had lots of vertical whitespace. Because of this less lines could fit in a screen. In 1.8.3, vertical whitespace has been reduced considerably.

| Timestamp | Value |
|---|---|
| 2010.Jun.10 10:32:03 | before. |
| 2010.Jun.10 10:31:59 | too good |
| 2010.Jun.10 10:31:56 | look |
| 2010.Jun.10 10:31:52 | did not |
| 2010.Jun.10 10:31:47 | data |
| 2010.Jun.10 10:31:43 | Text |

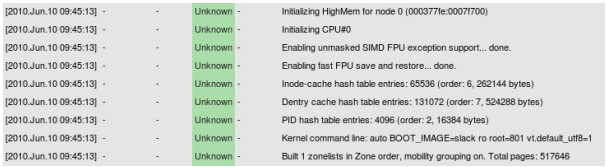
Text data in 1.8.2 and before

| Timestamp | Value |
|---|---|
| [2010.Jun.10 10:33:06] | now. |
| [2010.Jun.10 10:33:02] | It's better |
| [2010.Jun.10 10:32:03] | before. |
| [2010.Jun.10 10:31:59] | too good |
| [2010.Jun.10 10:31:56] | look |
| [2010.Jun.10 10:31:52] | did not |
| [2010.Jun.10 10:31:47] | data |
| [2010.Jun.10 10:31:43] | Text |

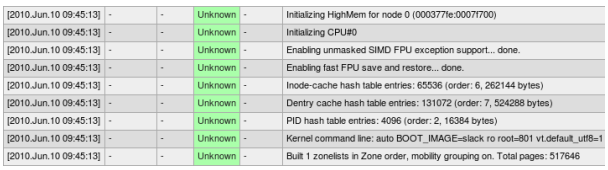
Text data in 1.8.3

Note that in Zabbix history viewing newer items are at the top, so you have to read lines from bottom up.

While logfile viewing did not have so much excessive vertical whitespace, it could still be slightly reduced even when adding table lines in 1.8.3.




Log data in 1.8.2 and before



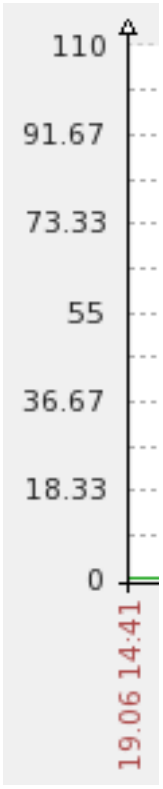
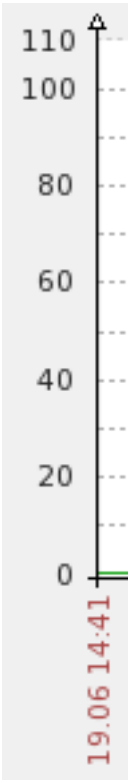
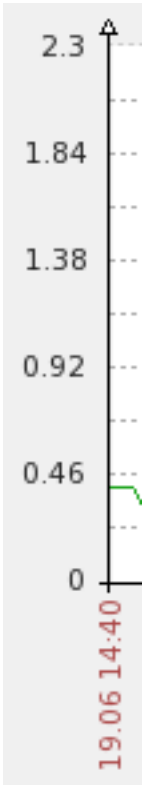
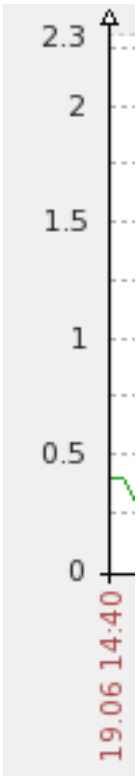

Log data in 1.8.3

7.1.17 Improved y axis labels

In previous versions, y axis labels for Zabbix graphs were displayed at even distances for full graph scale. In some cases, this resulted in label values that were hard to read. For example, graph going from 0 to 100 would have labels including 18.33, 36.67 and 91.67. In Zabbix 1.8.3, label calculation is improved. Zabbix will split the scale based on some round value, thus in our example values would become 20, 40 and 80. Additionally, if the graph has manual maximum scale entered, it is always printed.

If printing the maximum would collide with last rounded value, last rounded value is hidden. For example, on scale from 0 to 110 both of the labels 100 and 110 are printed. On scale from 0 to 101 only the label 101 is printed.

These benefits are visible both with large and small graph values.

| Large values | | Small values | |
|---|---|---|---|
|  |  |  |  |
| 1.8.2 and before | 1.8.3 | 1.8.2 and before | 1.8.3 |

7.1.18 Ability to unlink and clear templates upon mass update

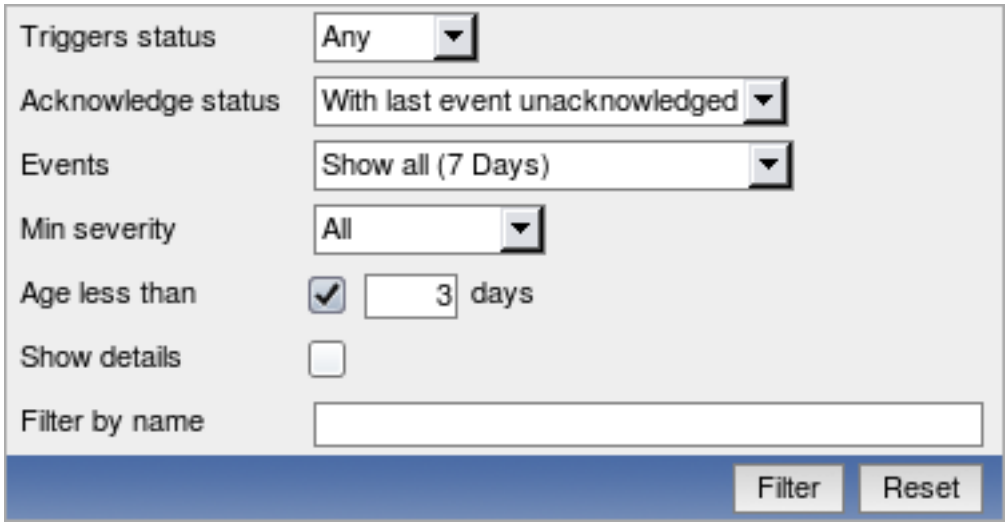
It is now possible to ”unlink and clear” (which not only unlinks the templates, but also deletes the linked elements) templates for host mass update. For this purpose, checkbox ”Clear when unlinking” must be marked in the ”Replace linked templates” section of the host mass update screen:

7.1.19 Improved trigger status filter

The filter in the ”Status of Triggers” page has been improved. Trigger acknowledge status has been brought out in a separate menu, and a new filtering option is available - ”With last event unacknowledged”. With these changes it is again possible to display all (including those in OK state) triggers for all hosts and host groups.

It is also now possible to filter triggers by age - option ”Age less than” allows to only show those triggers that have changed state in last N days.

Additionally, several options have been renamed to make it more clear what they are about - for example, *Select* became *Filter by name*.




7.1.20 Improved slideshow refresh rate selector

Instead of providing fixed periods, slideshow refresh can now be set to multiples of default settings. This allows to slow down or speed up slideshow displaying while still maintaining relative time of displaying for each slide.

| Refresh time multiplier |
|---|
| x0.25 |
| x0.5 |
| x1 |
| x1.5 |
| x2 |
| x3 |
| x4 |
| **x5** |

7.1.21 Translation updates

The following new translations have been added:

- Czech.

The following translations have been updated:

- Brazilian Portuguese;
- Chinese;
- French;
- German;
- Hungarian;
- Japanese;
- Latvian;
- Russian;
- Ukrainian.

7.1.22 Other frontend improvements

- Monitoring → Triggers section now shows the time it was last refreshed in the header.
- It is now possible to select multiple graphs, screens and maps when adding them to favourites on the dashboard.
- It is now possible to configure default axis side. Set in **include/defines.inc.php**, parameter GRAPH_YAXIS_SIDE_DEFAULT controls this. See frontend definition documentation for more information.
- Date formats can now be specified as a part of the locale, thus appropriate format for each region can be used now.
- In calculated item editing, field **Expression** was renamed to **Formula** to be less confusing (because triggers already had expression).

- Trigger option *Multiple TRUE events* was renamed to *Multiple PROBLEM events* to be consistent with other locations.
- Maximum rows per page and search limit increased to 99999.
- Screen name is shown when editing a screen.
- Row highlighting colour has been changed to be more visible.

---

Highlighting in 1.8.2 and before

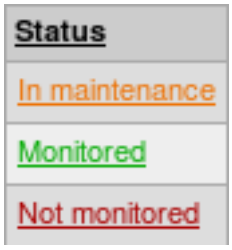
| | | |
|---|---|---|
| Template_Linux:Used disk space on /opt in % | Triggers (0) | vfs.fs.size[/opt,pused] |
| Template_Linux:Used disk space on / in % | Triggers (0) | vfs.fs.size[/,pused] |
| Template_Linux:Version of zabbix_agent(d) running | Triggers (1) | agent.version |
| Template_Linux:WEB (HTTP) server is running | Triggers (1) | net.tcp.service[http] |

Highlighting in 1.8.3

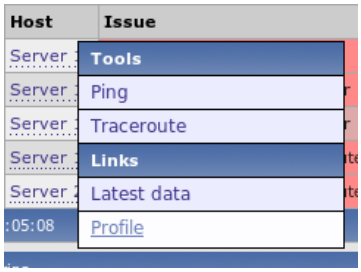
| | | |
|---|---|---|
| Template_Linux:Used disk space on /opt in % | Triggers (0) | vfs.fs.size[/opt,pused] |
| Template_Linux:Used disk space on / in % | Triggers (0) | vfs.fs.size[/,pused] |
| Template_Linux:Version of zabbix_agent(d) running | Triggers (1) | agent.version |
| Template_Linux:WEB (HTTP) server is running | Triggers (1) | net.tcp.service[http] |

---

- Being in maintenance is also displayed in the **Status** column in the host configuration page.




- If a URL is included in host profile information and it starts with *http* or *https*, it results in a clickable link in the inventory section.
- Host status now indicated by the text colour used for the host name in most locations in the frontend.
- Improved user macro handling - they can also be edited now.
- Switching item configuration filter options won't reload the item list immediately anymore, thus making filter easier to use on large installations. Also changing most options in item editing form won't reload it, just the appropriate fields will be updated - that should make item configuration easier.
- Improved performance by eliminating needless DISTINCT keyword usage. Some queries have improved in speed from 30 seconds to 0.1 second.
- Alternating row colours removed from all editing forms (they are still present in lists) and form background colour unified across the frontend.
- Long host, host group and graph names could push selection dropdowns past the right edge of the browser window. In 1.8.3, the situation is improved by limiting dropdown width if element names are too long. Full names are still used in the dropdown itself.
- Improved audit section names in dropdown and page header.
- Bundled DejaVu font upgraded from 2.30 to 2.31.
- Added link to host profile from dashboard, if host has profile information populated.




**7.2 Server and proxy changes** 7.2.1 Passive Zabbix proxy

Up to Zabbix version 1.8.3 Zabbix proxy was the one connecting to Zabbix server, and that was the only supported mode. But if one would like to place Zabbix server inside the local network which would monitor host in DMZ by proxy, that would not be possible - connection would have to go the other way.

Starting with version 1.8.3, Zabbix server and proxy communication can be in either direction. To control this, several new options have been introduced.

New options for the proxy:

- **ProxyMode** - controls the mode that proxy works in. By default, proxy works in active mode (connecting to sever). Setting this parameter to 1 will make proxy wait for incoming server connections.

New options for the server:

- **StartProxyPollers** - how many special pollers for passive proxies to start.
- **ProxyConfigFrequency** - how often Zabbix server sends configuration changes to passive proxies.
- **ProxyDataFrequency** - how often Zabbix server requests data from passive proxies.

7.2.2 Changed configuration parameters

- Performance improvements in database synchronisation by introducing new server configuration parameter - **StartDB-Syncers**. This allows to parallelise data writing to the database, resulting in notable improvement on powerful hardware. By default 4 DB syncers are started.
- Made poller balancing more intuitive by making them take items from a single item queue. As a consequence, unreachable server polling has also been improved and **StartPollersUnreachable** configuration parameter has been removed.

7.2.3 Server performance improvements

- Improved performance by not storing some runtime data in the database.
- Improved performance of updating trends in memory and their flushing to disk by caching required information.
- Faster configuration cache building both by improved SQL queries and in-memory optimisations.

7.2.4 Other server improvements

- Processes now provide more information on why they terminate. This should help greatly with Zabbix daemon debugging.
- Reduced amount of memory required for Zabbix server by storing repeated strings (like item keys) only once in the memory. On larger setups this can result in tenfold reduction in the memory usage.
- Zabbix daemons now support binding to multiple network interfaces, thus it is possible to specify comma-separated list to the ListenIP directive.
- Multirow inserts are used for PostgreSQL 8.2 and higher, which should improve the performance.
- If an e-mail subject would contain ASCII characters only, it won't be Base64 encoded anymore (originally implemented in 1.8.2).
- Hostname is now printed in server error messages about simple checks. That should ease debugging of failing items.
- Outdated environment variable setting for alert scripts has been removed.

**7.3 Other changes** 7.3.1 New supported and changed items

7.3.1.1 Table record count

In addition to previously supported **zabbix[history]** and **zabbix[trends]** items, Zabbix now supports additional items to monitor amount of values in corresponding tables. Note that on most database engines usage of these items can seriously degrade the performance.

- zabbix[history_log]
- zabbix[history_str]
- zabbix[history_text]
- zabbix[history_uint]
- zabbix[trends_uint]

7.3.1.2 New simple checks

The following simple checks have been added:

- ldap
- ldap_perf
- ntp
- ntp_perf

7.3.1.3 More advanced queue item

Zabbix internal item **zabbix[queue]** gained parameter support. It is now possible to specify for how long item must be missing data to be counted. For example, **zabbix[queue,6,59]** will count all items that are late by 6-59 seconds, inclusive. Time based suffixes are supported for these parameters, so the following syntax will check for all items that are haven't been refreshed between one minute and 6 hours: **zabbix[queue,1m,6h]**.

By default first value is 6, and second value is empty, which means infinity.

7.3.1.4 Item changes

- Parameter **service.ntp** for item keys **net.tcp.service** and **net.tcp.service.perf** renamed to **ntp**. Old syntax is still supported.

7.3.2 Changed acknowledge logic

This version changes how acknowledge works. In previous versions, if a trigger was acknowledged and changed state to UNKNOWN, then back, this new event would not be acknowledged (and in turn, trigger would become un-acknowledged). Since version 1.8.3, any acknowledge is copied to all events of the same type that are separated by UNKNOWN events. For example, if events have happened like this (newer events on top):

```
9. PROBLEM
8. OK
7. UNKNOWN
6. OK
5. PROBLEM
4. UNKNOWN
3. PROBLEM
2. OK
1. PROBLEM
```

Acknowledging event number 3 in the frontend would add the same acknowledge to event 5, and vice versa. If new events would now happen:

```
11. PROBLEM
10. UNKNOWN
...
```

In this case Zabbix server would add the acknowledge (if any) from event 9 to the new PROBLEM event number 11.

There is one exception - multiple TRUE (PROBLEM) triggers do not get the acknowledges copied over from other events.

7.3.3 Requirement changes

7.3.3.1 MySQL 5.5 supported

Upcoming MySQL version 5.5 drops support for table keyword *type*, and using *ENGINE* is required. Zabbix 1.8.3 changes to usage of keyword *ENGINE*.

## 8 What's new in Zabbix 1.8.4

### 8.1 Frontend improvements

8.1.1 Host configuration filter

Zabbix 1.8.4 introduces a filter in host configuration. In addition to the group selector, it is now possible to filter hosts on:

- Name
- IP
- DNS
- Port


Name, IP and DNS filter works as a substring, port is an exact match.

8.1.2 More control over rounding

Previously, value rounding in Zabbix was pretty much hardcoded. For example, version 1.8.3 displayed 2 decimal places for numbers above 1, and 6 decimal places for numbers below 1. 1.8.4 changes the threshold to 0.01 to reduce clutter, but additionally it provides more control over rounding. In the frontend definition file, three new parameters can be configured.

- ZBX_UNITS_ROUNDOFF_THRESHOLD

Threshold value for roundoff constants. Values less than this will be rounded to ZBX_UNITS_ROUNDOFF_LOWER_LIMIT number of digits after comma, greater to ZBX_UNITS_ROUNDOFF_UPPER_LIMIT. Default: 0.01

- ZBX_UNITS_ROUNDOFF_UPPER_LIMIT

Number of digits after comma, when value is greater than roundoff threshold. Default: 2

- ZBX_UNITS_ROUNDOFF_LOWER_LIMIT

Number of digits after comma, when value is less than roundoff threshold. Default: 6

The default value change alone should reduce clutter in graph legend for items like CPU load. A few examples of the added configurability (legend in all the examples is based on the same data)):

| | |
|---|---|
| last min avg max<br>0.01 0 0.007886 0.02 | Default roundoff values with 2 decimal places > 0.01 and 6 below |
| last min avg max<br>0.01 0 0.01 0.02 | All roundoff limited to two digits |
| last min avg max<br>0.011 0 0.007886 0.02 | Default roundoff values, but threshold changed to 0.02 |

8.1.3 XML validation

To catch any issues with XML as early as possible, Zabbix 1.8.4 introduces XML validation before importing it. This currently checks generic things like XML structure and data types, and is only available for screen import. Nevertheless, it should improve data quality, especially for users who generate screen configuration outside Zabbix.

8.1.4 Zapcat compatibility mode

Based on user feedback, this version introduces a new parameter that allows to enable Zapcat legacy support. When enabled, parameter ZAPCAT_COMPATIBILITY allows to use item key syntax that would otherwise be rejected as invalid. Please note that this is legacy syntax support which will be only available in 1.8 series.

8.1.5 Reversed steps in web monitoring graph legend

Built-in web monitoring graphs have had entries reversed in their legend to better match the order in web scenario configuration, so for steps like these:

| Name | Timeout | URL |
|---|---|---|
| First page | 15 sec | http://www.openstreetmap.org/ |
| Log in | 15 sec | https://www.openstreetmap.org/login |
| Traces | 15 sec | http://www.openstreetmap.org/traces/mine |

output has changed like this:

| | |
|---|---|
| Download speed for step 'Traces' of scenario 'Map'<br>Download speed for step 'Log in' of scenario 'Map'<br>Download speed for step 'First page' of scenario 'Map' | Download speed for step<br>Download speed for step<br>Download speed for step |
| Before 1.8.4 | Since 1.8.4 |

8.1.6 Trigger expression helper improvements

Two minor improvements are available for the trigger expression helper.

8.1.6.1 Entering function parameter time_shift

It is now possible to enter time_shift trigger function parameter in the trigger expression helper.


8.1.6.2 Filtered function dropdown

When choosing function to be used in constructed trigger expression, previously all choices were displayed always. Now dropdown list is filtered so that only functions that are valid for the chosen item type are available.

8.1.7 List of all hosts available in hostgroup properties

It should be easier to edit hostgroup membership on small Zabbix setups now - the view of other hosts gained ability to list all hosts that don't belong to the currently edited group.

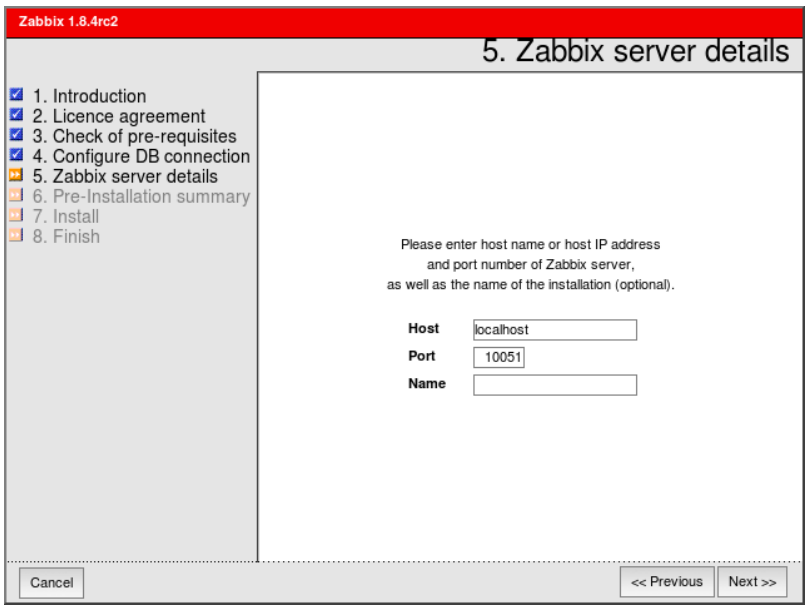

8.1.8 Zabbix server details visible in status report

In *Reports → Status of Zabbix* (available also as a dashboard widget for Zabbix superadmins), Zabbix server host and port are now displayed, as configured in the frontend.

| Parameter | Value | Details |
|---|---|---|
| Zabbix server is running | Yes | localhost:10051 |

8.1.9 Setting server name during the installation

The feature introduced in Zabbix 1.8, ability to display server name on the frontend, can now be configured using the frontend configuration wizard.




8.1.10 Frontend history override parameter

A new frontend definition has been added - **ZBX_HISTORY_DATA_UPKEEP**

This parameter allows to override frontend's choices when deciding whether to use history or trends table to display data, specified in days. Possible values:

- lower then zero - Zabbix uses the "Keep history" value for each item

- equal to zero - Zabbix only uses trend data
- greater then zero - Zabbix always uses this value instead of the ”Keep history” value

The default behaviour does not change and Zabbix still uses ”Keep history” value for each item.
This might be useful in setups with partitioned data storage.

8.1.11 Updated translations

- French
- Latvian
- Russian
- Ukrainian

8.1.12 Other frontend improvements

- Improved network map performance.
- Improved *Administration → Notifications* report performance.
- Bundled DejaVu font upgraded from 2.31 to 2.32.
- Screens can now be referenced on the frontend not only by id, but also by name. Adding GET parameter **screenname** will open the screen with that name, for example:

```
http://zabbix/zabbix/screens.php?screenname=Local%20servers
```

If both **elementid** (screen id) and **screenname** are specified, **screenname** has higher priority.
If **screenname** parameter is used, selected screen won't be saved in user profile - that is, revisiting screen section later will retrieve previously chosen screen, not the one referenced by name.

### 8.2 New and changed supported items

- On Linux, support for net.tcp.listen and net.udp.listen items has been added.
- Network traffic can now be monitored on OpenBSD without being root.
- Format of the sensors key has been changed. This item is now supported on Linux 2.4 and OpenBSD.

### 8.3 Zabbix daemon related improvements

8.3.1 Zabbix agent daemon improvements

On AIX, supported technology level is printed when executing agent with *--version* flag. Possible values:

- Supported technology levels: 5100
- Supported technology levels: 5200
- Supported technology levels: 5300-00,01,02,03,04,05
- Supported technology levels: 5300-06 and above
- Supported technology levels: 6100 and above

8.3.2 Zabbix sender improvements

Zabbix sender is a command line utility for sending custom data to Zabbix server.

8.3.2.1 Value pooling

Utility zabbix_sender has been improved in realtime sending scenarios by gathering multiple values that are passed to it in close succession, and sending them to the server in single connection. Value that is not further apart from previous value than 0.2 seconds can be put in the same stack, but maximum pooling time still is 1 second.

8.3.2.2 Using default hostname

It is now possible to use hostname ”-” in the input file. This will use the default hostname from the configuration file, thus allowing to send data in always using local hostname instead of hardcoding it.

8.3.3 Zabbix server improvements

Zabbix server performance has been improved for trigger functions **last** and **prev** by not retrieving redundant information from the database.

Memory usage has been reduced for web monitoring.

Reduced memory usage and fragmentation for the configuration cache.

Unreachable poller concept has been reintroduced along with resurrection of `StartPollersUnreachable` Zabbix server configuration parameter.

Zabbix server by default is not built with libcurl support anymore - it has to be specified explicitly.

8.3.3.1 New and improved macros

Support for several new macros has been added in notifications.

- {PROXY.NAME} macro is now available in trigger, network discovery and active agent auto-registration notifications
- {HOSTNAME} macro is available in active agent auto-registration notifications
- Value mapping is now available for {ITEM.VALUE} macro in trigger notifications

8.3.3.2 New trigger functions

New trigger function has been added - **strlen**, which returns length of the last value in characters.

8.3.3.3 Improved error messages

Error messages in the Zabbix server logfile regarding web monitoring have been improved - now they will also include information about scenario and step that produced the error message.

8.3.3.4 Improved fping handling with source IP specified

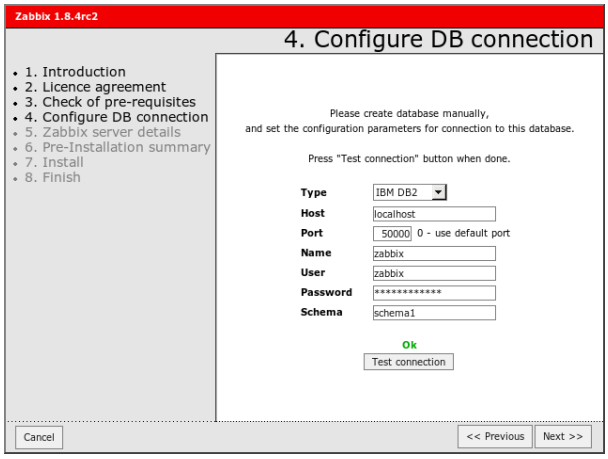
Utility used by Zabbix to perform ICMP pings, fping, is not very actively maintained, thus several features are only available as patchsets. One of those is an ability to specify source IP, which can either be unsupported, or supported with different flags (**-S** or **-I**). If **SourceIP** parameter is specified in Zabbix server configuration file, Zabbix attempts to specify source IP for fping as well. Before 1.8.4, Zabbix server always passed **-S**. If that was not supported, it just failed. Starting with 1.8.4, Zabbix server tries to detect whether **-S** or **-I** is supported. If neither is, fping is called without source IP parameter.

This detection happens by looking at the output of **fping -h**. In Zabbix 1.8.4, each pinger process detects fping capabilities individually when it is started.

**8.4 Misc improvements** User parameters in agent daemon files have been split out in several files that are included from the main configuration file. More examples have been added as well.

8.4.1 DB2 support

Support for additional database backend - IBM DB2 - has been added.



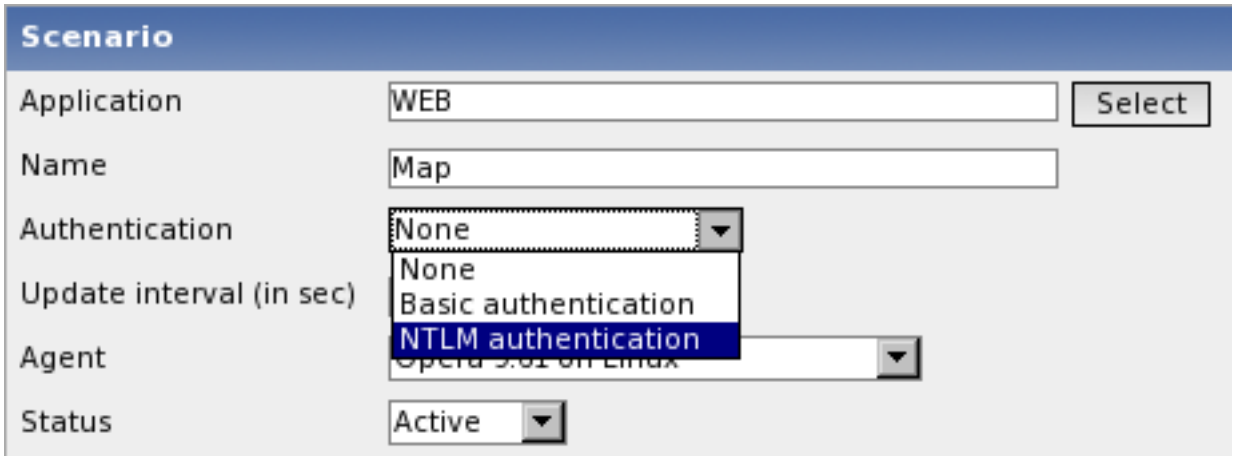

8.4.2 Extended user macro support

User macros can now be used in these additional locations:

- SNMP items and discovery
  - community
  - OID
  - security name
  - auth passphrase
  - priv passphrase
- parameters field of database item
- item descriptions and trigger names

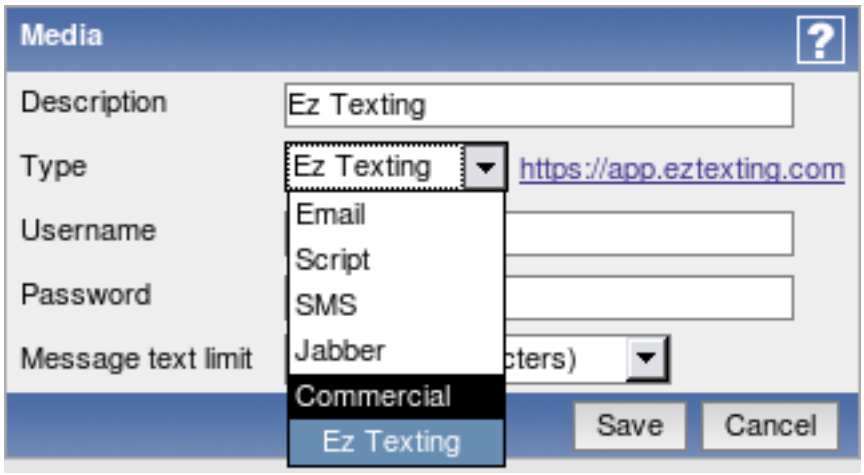
8.4.3 NTLM authentication for web monitoring

For built-in web monitoring, NTLM (Windows NT LAN Manager) authentication is now supported.




8.4.4 Direct support for Ez Texting

It is now possible to use Zabbix technological partner Ez Texting for message sending without custom media types - it can be selected as one of directly supported media types and access details can be provided in Zabbix media definition.




8.4.5 Improved problem reporting

If Zabbix server is compiled without web monitoring support, but web monitoring is attempted to be used, a helpful message will be visible in the frontend:


**9 What's new in Zabbix 1.8.5**

**9.1 Frontend improvements** 9.1.1 Network map improvements

Network maps can now be referenced on the frontend not only by id, but also by name, just like screens in 1.8.4. Adding GET parameter **mapname** will open the map with that name, for example:

`http://zabbix/zabbix/maps.php?mapname=Drag%20and%20drop%20map`

If both **sysmapid** (network map ID) and **mapname** are specified, **mapname** has higher priority.

If **mapname** parameter is used, selected network map won't be saved in user profile - that is, revisiting map section later will retrieve previously chosen map, not the one referenced by name.

9.1.2 Removed Zapcat compatibility switch

Zapcat compatibility switch, added in 1.8.4, has been removed. Instead, expression parser has been reworked to accept Zapcat syntax by default.

9.1.3 Reordered configuration menu

Sequence of configuration menu entries *Discovery* and *IT services* was changed to match the one in monitoring section.

9.1.4 Added translations

- Slovak

9.1.5 Updated translations

- Brazilian Portuguese
- Latvian
- Japanese
- Russian
- Ukrainian

**9.2 Zabbix daemon improvements** 9.2.1 Zabbix agent improvements

9.2.1.1 Improved performance

Zabbix agent performance has been improved, especially on systems with many cores.

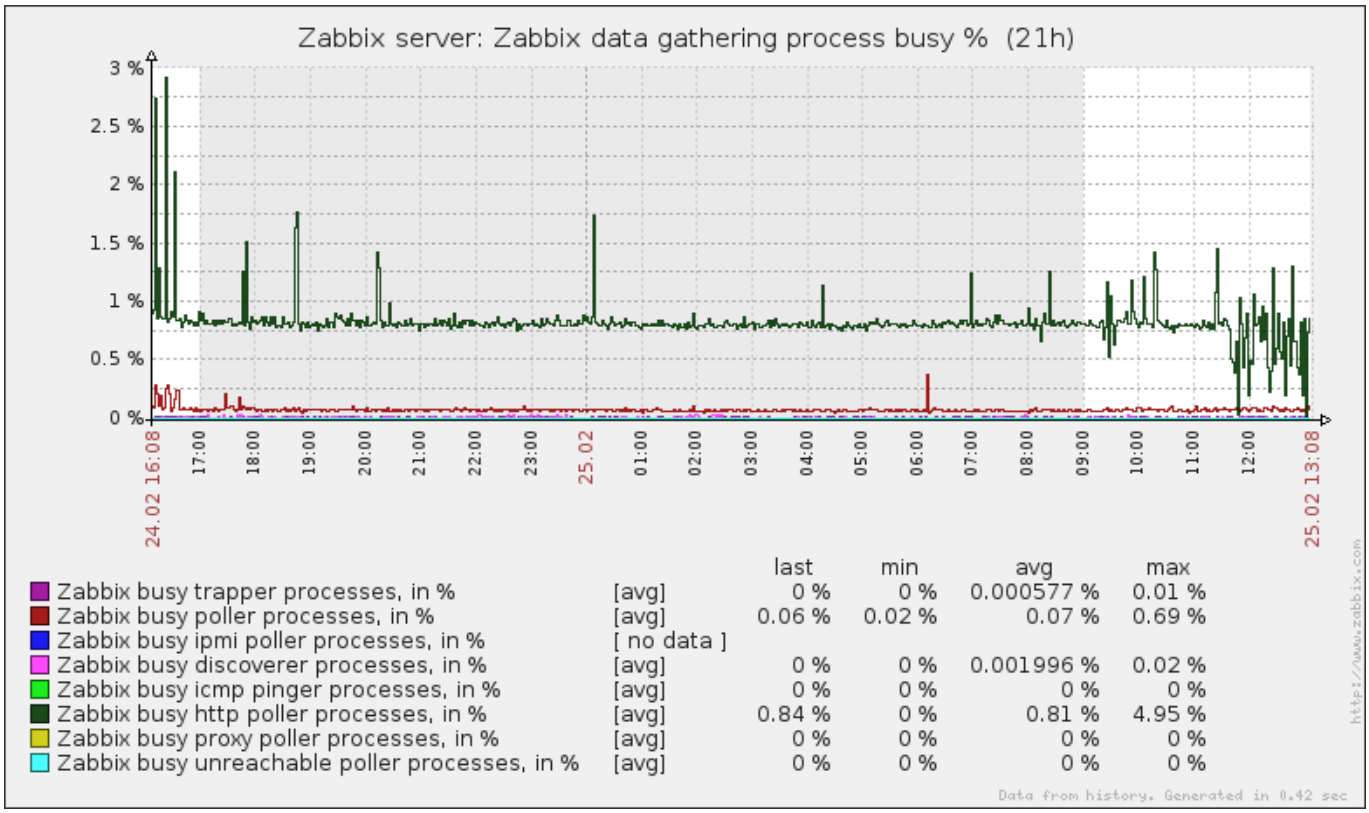
9.2.1.2 Unified internal process startup messages

Zabbix agent now prints internal process types upon startup just like the server does. Example Zabbix agent daemon startup log:

```
943:20110224:111750.848 agent #0 started [collector]
943:20110224:111750.848 agent #1 started [listener]
943:20110224:111750.850 agent #2 started [listener]
943:20110224:111750.851 agent #3 started [listener]
943:20110224:111750.851 agent #4 started [active checks]
```

9.2.2 New items supported

New internal item was added to monitor Zabbix process state. Item **zabbix[process,<type>,<mode>,<state>]** allows to monitor busy or idle percentage of different Zabbix server processes over the last minute.




It is now possible to determine how much each of Zabbix server internal processes spent in a busy state. This should help with evaluating performance, estimating how many poller processes to use and other Zabbix finetuning tasks.

9.2.3 Process limits changed

Zabbix daemon internal process limits have been changed. These changes affect Zabbix server and proxy daemons.

| Option | Previous limit | New limit |
|---|---|---|
| StartDBSyncers | 64 | 100 |
| StartDiscoverers | 255 | 250 |

| Option | Previous limit | New limit |
|---|---|---|
| StartHTTPPollers | 255 | 1000 |
| StartIPMIPollers | 255 | 1000 |
| StartPingers | 255 | 1000 |
| StartPollersUnreachable | 255 | 1000 |
| StartPollers | 255 | 1000 |
| StartProxyPollers | 255 | 250 |
| StartTrappers | 255 | 1000 |

For Zabbix agent daemon, maximum value of **StartAgents** has been increased from 16 to 100.

9.2.4 Listening on IPv6 and IPv4

Support for listening on all IPv4 and IPv6 addresses at the same time has been added.

9.2.5 Global regular expression support with proxies

Global (user definable) regular expression support has been added for use with Zabbix proxies.

**9.3 Misc improvements** Added Ubuntu upstart configuration files.

9.3.1 New trigger functions

Several new trigger functions have been added.

- **dayofmonth** returns current date
- **logeventid** checks whether Event ID of the last log entry matches a regular expression

## 10 What's new in Zabbix 1.8.6

### 10.1 Frontend improvements

- When switching from one host to another while having a custom graph open, Zabbix will try to select a graph with the same name on the target host.
- Previously, graph mode *dynamic* or *fixed* was reset back to *fixed* whenever the page was reloaded. Starting with Zabbix 1.8.6, this mode is saved for each location (graphs, screens etc) individually.

10.1.1 Updated translations

- Japanese

### 10.2 Zabbix daemon improvements

- Zabbix proxy now logs a message when it receives the configuration data from the server.
- All Zabbix daemons now refuse to start up if configuration file contains unrecognised parameters. This should help to discover incorrectly spelt parameters.
- Improved Zabbix server performance when gathering text data.
- Improved Zabbix server performance when processing unsupported items. Previously, information about unsupported parameters was stored directly in the database bypassing write cache. Starting with 1.8.6, it uses write cache, thus reducing database load. Internal item **zabbix[wcache,...]** has been extended to monitor amount of *not supported* values in the cache.
- It is now possible to customise automatic host name setting on the Zabbix agent and proxy daemons. Previously, it always returned system.hostname contents. A new configuration parameter, HostnameItem, allows to set another item that will be used if configuration parameter Hostname is not set. If both HostnameItem and Hostname are set, Hostname takes precedence.
- For Jabber/XMPP notifications, server now supports SRV record lookup.
- Improved housekeeper performance on PostgreSQL.
- Warnings while connecting to Oracle database are now logged.
- Zabbix agents and proxies now log their hostname in the log file when starting up. Additionally, Zabbix proxies log whether they are operating in the passive or active mode.

10.2.1 Configuration cache reloading

Zabbix server and proxy daemons now have an ability to reload in-memory configuration cache upon request. This can be done by passing parameter **-R** or **--runtime-control** and a runtime control option. Currently only one runtime option is supported - **config_cache_reload**.

Additional benefit when using active Zabbix proxies is that active proxy will request configuration from the Zabbix server upon receiving this request.

When a daemon receives such a request, it also logs it in the logfile. If the request is sent while configuration cache is being updated, it is ignored (but log file entry about signal received is still added).

### 10.3 Item changes

- Item **vfs.file.md5sum[]** previously was limited to files less than 64MB. This limit has been removed in 1.8.6 and item is only limited by the time it takes to compute the checksum.
- Item **system.hostname[]** on Windows now has an additional parameter to choose between NetBIOS and host name.
- Items **vfs.dev.read** and **vfs.dev.write** have gained support for LVM. Additionally, in previous versions of Zabbix only relative device name could be used (like *sda*). Now **/dev/** prefix may also optionally be used. A few examples of supported syntax:
  - vfs.dev.read[mapper/VolGroup01-LogVol00,sectors]
  - vfs.dev.read[VolGroup01/LogVol00,sectors]
  - vfs.dev.read[/dev/sda,operations]
  - vfs.dev.read[sdb,operations]
- Items **net.tcp.dns.query** gained support for SRV records.
- **net.if.*** checks under Windows now support multibyte NIC names.

### 10.4 General changes

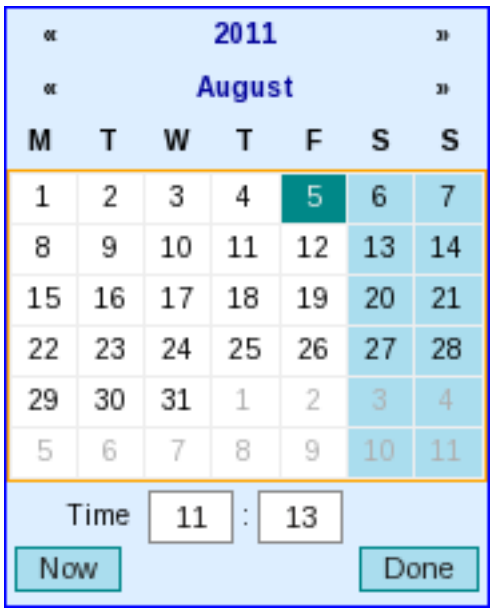
Support for PostgreSQL 9+ was added.

## 11 What's new in Zabbix 1.8.7

### 11.1 Frontend improvements

11.1.1 Improved time picker

When dates in the upper right corner of the time scrollbar are clicked, popup appears with a calendar and input fields for time. Previously, it could only be confirmed by clicking on a date, and that also immediately closed it. Thus, to select time and date, one had to enter time first, then click on a date. Zabbix 1.8.7 improves usability of this popup. A button *Done* was added. Clicking on a date now does not close the popup, thus user may choose desired date and time and click on *Done* when satisfied with the selection.




Additionally, *Now* button was added that sets the calendar date and time to current moment (but still keeps it open).

Clicking on the date/time string again will close the popup and discard the changes.

11.1.2 Default action conditions

When a new action for triggers is created, it now gets two additional conditions automatically:

- *Trigger value = PROBLEM*

- *Maintenance status = not in maintenance*

11.1.3 Updated translations

- Japanese
- French

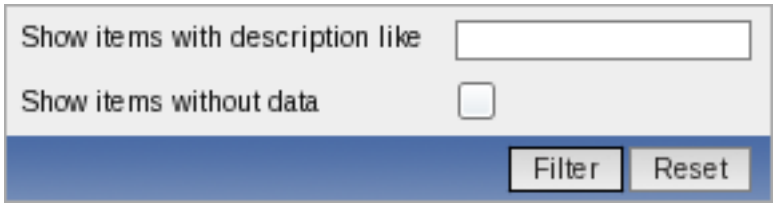
**11.2 API improvements** API flag `searchWildcardsEnabled` has been added. If it is set to **1**, database wildcards may be used in search patterns.

**11.3 Daemon improvements** Zabbix server has a capability to notify users in a specific user group in case of database unavailability. Previously, Zabbix server only loaded user information when starting up. Now watchdog process reloads this information every **CacheUpdateFrequency** seconds, which is every 60 seconds by default. This is a separate reloading than general configuration cache updating, thus it is not affected by the forced cache reloading.

**11 What's new in Zabbix 1.8.8**

**11.1 Frontend improvements** 11.1.1 Latest data improvements

- Starting with Zabbix 1.8.8, all hosts and hostgroups are shown in *Monitoring → Latest data*. Previously, hosts without data were not listed here.
- *Monitoring → Latest data* section gains a new filter to show items without data. Previously, they were not displayed at all. By default such items are still not visible.




11.1.2 API changes

- API search wildcard was changed from "%" to "*"

11.1.3 Updated translations

- French

**11.2 Daemon improvements** 11.2.1 Zabbix server improvements

Zabbix server performance has been improved when calculating trigger expressions.

The processes that take values from history cache and put them in the database - **history syncers** (configured by the server configuration parameter **StartDBSyncers**) take up to 1000 values on each iteration and calculate all triggers that reference corresponding items. Previously, each trigger function would result in a separate query to the database. Starting with Zabbix 1.8.8, a single SQL query is prepared for all of the functions in all triggers that history syncer has been tasked with updating.

Additionally, if there are multiple triggers in such a batch that check text or log items, last two values for items like those are cached for the current run.

In total, this results in a significantly decreased query count against the database and thus performance improvements.

The same improvements also affect the internal **timer** process (only one such process may run currently), which calculates triggers that include time based functions.

History syncers try to take a new batch of values from the history cache 0-5 seconds after they have finished the work on the previous one, based on how many values it got previously (for example, if it got 1000 values, it will make the next attempt immediately after finishing the work).

**11.3 General improvements** Macro **{TRIGGER.ID}** is now supported in frontend and notifications if used inside trigger URL field.

**11 What's new in Zabbix 1.8.9**

**11.1 Frontend improvements** Improved performance for event acknowledging.

11.1.1 Updated translations

- Brazilian Portuguese

**11.2 Supported PostgreSQL version changes**

- Support for PostgreSQL 9.1 was added
- PostgreSQL support changed from *7.0.2 and later* to *7.4 and later*.

**11.3 Zabbix daemon improvements** 11.3.1 Zabbix server improvements

11.3.1.1 Improved log messages

Zabbix server log messages about failed checks have been improved. Previously, if a check failed, Zabbix server would log messages, similar to these:

```
Zabbix host [monitored host]: first network error, wait for 15 seconds
Zabbix host [monitored host]: another network error, wait for 15 seconds
```

After 3 failures like these, host would be disabled and a message logged:

```
Disabling Zabbix host [monitored host]
```

If a host then became available, a log message would say:

```
Enabling Zabbix host [monitored host]
```

Starting with Zabbix 1.8.9, this has been improved. First, messages now tell which specific item key failed, thus allowing to see whether the problem happens with multiple items or just one:

```
Zabbix agent item [proc.num[sshd]] on host [monitored host] failed: another network error, wait for 15 sec
Zabbix agent item [system.cpu.load] on host [monitored host] failed: another network error, wait for 15 se
```

Additionally, for all failures item type is logged as well (in this case, *Zabbix agent*).

If a host is determined to be unavailable and then available, messages are a bit more verbose now and also include check type:

```
temporarily disabling Zabbix agent checks on host [monitored host]: host unavailable
enabling Zabbix agent checks on host [monitored host]: host became available
```

And a new message has been introduced if host responds after one or two failures (this was not logged before at all):

```
resuming Zabbix agent checks on host [monitored host]: connection restored
```

These changes should allow for much easier debugging of connectivity or configuration issues - for example, if all problems on a host would be associated with a user parameter, it would most likely be a performance problem with the executed command.

11.3.1.2 Acknowledge synchronisation in distributed mode

Starting with Zabbix 1.8.9 acknowledge status will be fully synchronised from child nodes to master node.

11.3.1.3 Performance improvements

Zabbix server performance was improved in some edge cases by skipping value updating for disabled or removed items.

11.3.1.4 Waiting for database to appear upon startup

Starting with Zabbix 1.8.9, upon startup Zabbix daemons will wait for database to be available. This will help with case when database is started by bootup process before Zabbix daemons, but takes long time to become ready.

11.3.2 Zabbix agent improvements

Zabbix agent daemon performance on AIX was improved by only collecting perfstat data if it is requested.

## 12 What's new in Zabbix 1.8.10

**12.1 Frontend improvements** 12.1.1 Updated translations

- French
- Japanese
- Russian

**12.2 Server improvements** Zabbix server housekeeper process previously only logged the amount of values that were removed from history and trends. Starting with Zabbix 1.8.10 it also logs the amount of values removed for deleted items, events, alerts and sessions.

## 13 What's new in Zabbix 1.8.11

**13.1 Frontend improvements** Global script error messages now suggest possible causes of the problem.

13.1.1 Updated translations

- Brazilian Portuguese
- French
- Japanese

**13.2 Daemon improvements**

- Added support for individual CPU statistics in **system.cpu.util** item on FreeBSD
- Support for all data types in SNMP dynamic indexes has been added
- Added possibility to remove a discovered host from "Discovered hosts" group by action
- Previously, Zabbix daemons used a hardcoded priority of 5. Starting with 1.8.11, default or user specified priority is used without any overrides on the Zabbix side
- Unused options TrendCacheSize and CacheUpdateFrequency have been removed from Zabbix proxies
- Error messages for missing SNMP instances are now more user friendly
- Unneeded information like user history for nodes and web monitoring items for proxies is not synchronised to them anymore
- Command line arguments that exceed 2KB are now supported in **proc.num** and **proc.mem** checks on Linux

## 14 What's new in Zabbix 1.8.12

**14.1 Multiple server support in active agent** Previously, Zabbix agent was able to communicate with one server or proxy in the active mode. In 1.8.12, a new parameter **ServerActive** has been added. If it is defined, only the hosts in this list will be used for active checks.

For each host in the list, a separate process will get list of items and process them - thus each server or proxy can send a totally different list of active checks to the agent and only get their values.

Zabbix sender does not use **ServerActive** parameter - it still takes the first value from **Server** parameter only.

> **Note:**
> See the "See also" section at the bottom of this page to read more details about these changes.

**14.2 New macros** Support for macros {ITEM.ID<1-9>} and {TRIGGER.EXPRESSION} has been added in notifications and commands.

### See also

1. Differences in the Zabbix agent configuration for active and passive checks starting from version 1.8.12

## 15 What's new in Zabbix 1.8.13

**15.1 Frontend improvements** Improved performance of 'Status of Zabbix' dashboard widget

15.1.1 Updated translations

- French

### 15.2 Daemon improvements

- Zabbix server performance when using a partitioned PostreSQL setup was improved.
- A new item has been added - **agent.hostname**. It returns the value agent would use for communicating with Zabbix server or proxy for active items. As an agent can obtain this value from 3 different sources (**Hostname** configuration parameter, item key, specified in **HostnameItem** parameter or from the default **system.hostname** key), the new item allows to determine what is the actual value for debugging or verification purposes.

## 15 What's new in Zabbix 1.8.14

**15.1 Frontend improvements** Improved screen performance for non-Superadmin users.

## 16 What's new in Zabbix 1.8.15

**16.1 Frontend improvements** Performance in graph related pages was improved.

## 17 What's new in Zabbix 1.8.16

### 17.1 Daemon improvements

- Suffixes "KMGTsmhdw" are now supported in the second parameter of count() trigger function
- Previously it was possible to send large amount of data to the Zabbix server, potentially exhausting memory. This is now limited to accept only 128MB when using Zabbix protocol. Any other data (including older Zabbix protocols) stays limited at 16MB.
- When encoding email subject from UTF-8 to Base64, lines longer than 75 bytes will be split up to conform with RFC-2047.

## 18 What's new in Zabbix 1.8.17

**18.1 Frontend improvements** 18.1.1 Updated translations

- Japanese

**18.2 Daemon improvements** 18.2.1 Improved SNMP performance

- Performance of the SNMP normalize function was improved.
- Previously, each dynamic index was obtained individually when requested. On systems with thousands of dynamic indexes Zabbix pollers were 100% busy for more than 10 minutes until the cache was built up. Now the first dynamic index request will cause all indexes in the same SNMP table to be looked up and processed in one go. This greatly speeds up the cache building and reduces the SNMP query count. Note that all entries are looked up even if only some would be used later.

**18.3 Miscellaneous improvements** Zabbix sender now supports *ServerActive* parameter from the agent daemon configuration file. The first IP address is used.

## 19 What's new in Zabbix 1.8.18

### 19.1 Daemon improvements

- Zabbix server now correctly enables SSL host verification when using Ez Texting service to send alerts.

#### 20 What's new in Zabbix 1.8.20

#### 20.1 Daemon improvements

- Zabbix application names in syslog fixed to meet RFC 5424 for APP-NAME. See Syslog application names change

#### 20.2 Frontend improvements

- LDAP authetication bind password, once stored in the database, was accessible to Zabbix Super Admin level users in clear text in HTML source code. Fixed for 1.8.20, by hiding the password from clear view.

#### 21 What's new in Zabbix 1.8.21

#### 21.1 Frontend improvements

- Improved SQL query performance and page execution in Maintenance configuration page.

#### 22 What's new in Zabbix 1.8.22

> **Warning:**
> Zabbix 1.8 is not supported anymore. See lifecycle and release policy page for more information.

Zabbix 1.8.22 was a bugfix release that received no functional improvements.

## 2 Installation

how_to_get_zabbix requirements components installation_from_source upgrading appliance

#### 1 How to Get Zabbix

Check the Zabbix Home Page at http://www.zabbix.com for information about the current version and for downloading instructions.

Zabbix is distributed as a source package, however it is also included into number of OS distributions pre-compiled.

#### 2 Requirements

2.1 Hardware requirements

2.1.1 Memory Requirements

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!

2.1.2 CPU Requirements

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

2.1.3 Other hardware

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

2.1.4 Examples of hardware configuration

The table provides several hardware configurations:

| Name | Platform | CPU/Memory | Database | Monitored hosts |
|---|---|---|---|---|
| Small | Ubuntu Linux | PII 350MHz 256MB | MySQL MyISAM | 20 |
| Medium | Ubuntu Linux 64 bit | AMD Athlon 3200+ 2GB | MySQL InnoDB | 500 |
| Large | Ubuntu Linux 64 bit | Intel Dual Core 6400 4GB | RAID10 MySQL InnoDB or PostgreSQL | >1000 |
| Very large | RedHat Enterprise | Intel Xeon 2xCPU 8GB | Fast RAID10 MySQL InnoDB or PostgreSQL | >10000 |

**Note:**
Actual configuration depends on number of active items and refresh rates very much. It is highly recommended to run the database on a separate box for large installations.

2.2 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:

- AIX
- FreeBSD
- HP-UX
- Linux
- Mac OS/X
- NetBSD
- OpenBSD
- SCO Open Server
- Solaris
- Windows 2000, 2003, XP, Vista (only Zabbix agent)

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

2.3 Software Requirements

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

The following software is required to run Zabbix:

| Software | Version | Comments |
|---|---|---|
| **Apache** | 1.3.12 or later | |
| **PHP** | 5.0 or later | |
| **PHP modules:<br>php-gd** | GD 2.0 or later | PHP GD module must support PNG images. |
| **PHP TrueType support** | | --with-ttf |
| **PHP bc support** | | php-bcmath, --enable-bcmath |
| **PHP XML support** | | php-xml or php5-dom, if provided as a separate package by the distributor |
| **PHP session support** | | php-session, if provided as a separate package by the distributor |

| Software | Version | Comments |
|---|---|---|
| **PHP socket support** | | php-net-socket, --enable-sockets.<br>Required for user script support. |
| **PHP multibyte support** | | php-mbstring, --enable-mbstring |
| **IBM DB2<br>ibm_db2** | | Required if IBM DB2 is used as Zabbix back end database. |
| **MySQL<br>php-mysql** | 3.22 or later | Required if MySQL is used as Zabbix back end database. |
| **Oracle<br>oci8** | | Required if Oracle is used as Zabbix back-end database. |
| **PostgreSQL<br>php-pgsql** | 7.0.2 or later if Zabbix 1.8.9<br>7.4 or later if Zabbix >= 1.8.9 | Required if PostgreSQL is used as Zabbix back-end database.<br>Consider using PostgreSQL 8.x or later for much better performance.<br>It is suggested to use at least PostgreSQL 8.3, which introduced much better VACUUM performance. |
| **SQLite<br>php-sqlite3** | 3.3.5 or later | Required if SQLite is used as Zabbix back-end 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.

2.3.1 WEB browser on client side

Support for HTML and PNG images is required. Cookies and Java Script must be enabled. Latest versions of Mozilla Firefox, Microsoft Internet Explorer, Opera and Konqueror are supported. Other browsers (Google Chrome, Apple Safari) may work with Zabbix as well.

2.4 Server requirements

| Requirement | Description |
|---|---|
| OpenIPMI | Required for IPMI support |
| libssh2 | Required for SSH support. Version 1.0 or higher. |
| fping | Required for ICMP ping items. |

2.5 Choice of database engine

Zabbix Server and Proxy support five database engines:

- IBM DB2
- MySQL
- Oracle
- PostgreSQL
- SQLite

**Note:**
IBM DB2 is supported starting from Zabbix 1.8.4.

2.6 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 50 bytes per value. In our case, it means that 130M of values will require 130M * 50 bytes = **6.5GB** of disk space.

- 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 128 bytes per each total. Suppose we would like to keep trend data for 5 years. Values for 3000 items will require (3000/3600)*(24*3600***365**)* **128** = **3.4GB** per year, or **16.8GB** for 5 years. The first value **3600** in the formula represents trend averaging period, one hour.

- Housekeeper settings for events

Each Zabbix event requires approximately 130 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* **130** = **12.3GB**

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<br>items : number of items<br>days : number of days to keep history<br>refresh rate : average refresh rate of items<br>bytes : number of bytes required to keep single value, depends on database engine, normally 50 bytes. |
| **Trends** | days*(items/3600)*24*3600*bytes<br>items : number of items<br>days : number of days to keep history<br>bytes : number of bytes required to keep single trend, depends on database engine, normally 128 bytes. |
| **Events** | days*events*24*3600*bytes<br>events : number of event per second. One (1) event per second in worst case scenario.<br>days : number of days to keep history<br>bytes : number of bytes required to keep single trend, depends on database engine, normally 130 bytes. |

So, the total required disk space can be calculated as:
**Configuration + History + Trends + Events**
The disk space will NOT be used immediately after Zabbix installation. Database size will grow then it will stop growing at some point, which depends on hosekeeper settings.

> **Note:**
> Disk space requirements for nodes in distributed setup are calculated in a similar way, but this also depends on a total number of child nodes linked to a node.

2.7 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.

## 3 Components

### 3.1 Zabbix Components

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

### 3.2 Zabbix Server

This is the centre of the Zabbix software. The Server can remotely check networked services (such as web servers and mail servers) using simple service checks, but it is also the central component to which the Agents will report availability and integrity information and statistics. The Server is the central repository in which all configuration, statistical and operational data are stored, and it is the entity in the Zabbix software that will actively alert administrators when problems arise in any of the monitored systems.

Zabbix can also perform agent-less monitoring and also monitor network devices using SNMP agents.

### 3.3 Zabbix Proxy

The Proxy is an optional part of Zabbix deployment. The Proxy collects performance and availability data on behalf of Zabbix Server. All collected data is buffered locally and transferred to Zabbix Server the Proxy belongs to.

Zabbix Proxy is an ideal solution for a centralized monitoring of remote locations, branches, networks having no local administrators.

Zabbix Proxies can also be used to distribute load of a single Zabbix Server. In this case, only Proxies collect data thus making processing on the Server less CPU and disk I/O hungry.

### 3.4 Zabbix Agent

In order to actively monitor local resources and applications (such as harddrives, memory, processor statistics etc.) on networked systems, those systems must run the Zabbix Agent. The Agent will gather operational information from the system on which it is running, and report these data to the Zabbix for further processing. In case of failures (such as a harddisk running full, or a crashed service process), the Zabbix Server can actively alert the administrators of the particular machine that reported the failure.

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

### 3.5 The WEB Interface

In order to allow easy access to the monitoring data and the configuration of Zabbix from anywhere and from any platform, the Web-based Interface is provided. The Interface is a part of the Zabbix Server, and is usually (but not necessarily) run on the same physical machine as the one running the Zabbix Server.

> **Note:**
> Zabbix front-end must run on the same physical machine if SQLite is used.

## 4 Installation from Source

### 4.1 Software requirements

Building of Zabbix server or agents from sources requires additional software.

The following software is required to compile Zabbix (required versions):

**One of the following database engines:**

- IBM DB2 Headers and Libraries - CLI headers and libraries are required.
- MySQL Headers and Libraries.
- Oracle Headers and Libraries - OCI headers and libraries are required.
- PostgreSQL Headers and Libraries.
- SQLite Headers and Libraries.

> **Note:**
> Usually provided as part of mysql-dev, postgresql-dev, sqlite3-dev packages.

**NET-SNMP** (or **UCD-SNMP**) library and header files. Required for SNMP support. Optional.

**Iksemel** library and header files. Required to enable Jabber messaging. Optional.

**Libcurl** library and header files. Required for WEB monitoring module. Optional.

**C Compiler**. GNU C compiler is the best choice for open platforms. Other (HP, IBM) C compilers may be used as well.

**GNU Make**. GNU Make is required to process Zabbix Makefiles.

4.2 Structure of Zabbix distribution

- src

The directory contains sources for all Zabbix processes except frontends.

- src/zabbix_server

The directory contains Makefile and sources for zabbix_server.

- src/zabbix_agent

The directory contains Makefile and sources for zabbix_agent and zabbix_agentd.

- src/zabbix_get

The directory contains Makefile and sources for zabbix_get.

- src/zabbix_sender

The directory contains Makefile and sources for zabbix_sender.

- include

The directory contains Zabbix include files.

- misc
  - misc/init.d

The directory contains start-up scripts for different platforms.

- frontends
  - frontends/php

The directory contains files of PHP frontend.

- create

The directory contains SQL script for initial database creation.

- create/schema

Database creation schemas.

- create/data

Data for initial database creation.

- upgrades

The directory contains upgrade procedures for different versions of Zabbix.

4.3 Zabbix Server

Server side

Step 1

Create the Zabbix superuser account

This is the user the server will run as. For production use you should create a dedicated unprivileged account ('zabbix' is commonly used). Running Zabbix as 'root','bin', or any other account with special rights is a security risk. Do not do it!

> **Note:**
> Zabbix server process (zabbix_server) is protected from being run under root account.

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.

Step 2

Extract Zabbix sources

```
shell> tar -zxf zabbix-1.8.tar.gz
```

Step 3

Create the Zabbix database

Zabbix comes with SQL scripts used to create the required database schema and also to insert a default configuration. There are separate scripts for IBM DB2, MySQL, Oracle, PostgreSQL and SQLite.

**For IBM DB2**:

```
shell> db2 "create database zabbix using codeset utf-8 territory us pagesize 32768"
shell> cd create/schema
shell> db2batch -d zabbix -f ibm_db2.sql
shell> cd ../data
shell> db2batch -d zabbix -f data.sql
shell> db2batch -d zabbix -f images_ibm_db2.sql
```

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
```

**For MySQL**:

```
shell> mysql -u<username> -p<password>
mysql> create database zabbix character set utf8;
mysql> quit;
shell> cd create/schema
shell> cat mysql.sql | mysql -u<username> -p<password> zabbix
shell> cd ../data
shell> cat data.sql | mysql -u<username> -p<password> zabbix
shell> cat images_mysql.sql | mysql -u<username> -p<password> zabbix
```

**For Oracle** (we assume that user *zabbix* with password *password* exists and has permissions to create database objects in service ORCL):

```
shell> cd create
```

Copy directory data/images somewhere on oracle server, e. g. /home/oracle:

```
shell> scp -r data/images user@host:/home/oracle
```

Edit file data/images_oracle.sql and set images_dir variable to "/home/oracle/images":

```
CREATE OR REPLACE DIRECTORY image_dir AS '/home/oracle/images'
```

Proceed with importing data:

```
shell> sqlplus zabbix/password@host/ORCL
sqlplus> set def off
sqlplus> @schema/oracle.sql
sqlplus> @data/data.sql
sqlplus> @data/images_oracle.sql
sqlplus> exit
```

> **Note:**
> Zabbix requires UTF8 database character set. If database is not UTF8 it can be converted by running: `ALTER DATABASE NATIONAL CHARACTER SET UTF8;`

**For PostgreSQL**:

```
shell> psql -U <username>
psql> create database zabbix;
psql> \q
shell> cd create/schema
shell> cat postgresql.sql | psql -U <username> zabbix
shell> cd ../data
shell> cat data.sql | psql -U <username> zabbix
shell> cat images_pgsql.sql | psql -U <username> zabbix
```

**For SQLite**:

```
shell> cd create/schema
shell> cat sqlite.sql | sqlite3 /var/lib/sqlite/zabbix.db
shell> cd ../data
shell> cat data.sql | sqlite3 /var/lib/sqlite/zabbix.db
shell> cat images_sqlite3.sql | sqlite3 /var/lib/sqlite/zabbix.db
```

Step 4

Configure and compile the source code for your system

The sources must be compiled for both the server (monitoring machine) as well as the clients (monitored machines). To configure the source for the server, you must specify which database will be used.

```
shell> ./configure --enable-server --with-ibm-db2 --with-net-snmp --with-jabber --with-libcurl # for IBM D
```

or

```
shell> ./configure --enable-server --with-mysql --with-net-snmp --with-jabber --with-libcurl # for MySQL +
```

or

```
shell> ./configure --enable-server --with-oracle --with-net-snmp --with-jabber --with-libcurl # for Oracle
```

or

```
shell> ./configure --enable-server --with-pgsql --with-net-snmp --with-jabber --with-libcurl # for Postgre
```

or

```
shell> ./configure --enable-server --with-sqlite3 --with-net-snmp --with-jabber --with-libcurl # for SQLit
```

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

**Note:**
Flag --with-ucd-snmp can be used instead of --with-net-snmp. If no SNMP support is required, both --with-net-snmp and --with-ucd-snmp may be skipped.

However, if you want to compile client binaries along with server binaries, run:

```
shell> ./configure --enable-server --enable-agent --with-mysql --with-net-snmp --with-jabber --with-libcur
```

**Note:**
Use flag --enable-static 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 under Solaris.

Step 5

Make and install everything

```
shell> make install
```

By default, *make install* will install all the files in /usr/local/sbin, /usr/local/lib etc. Make sure that you have enough permissions.

You can specify an installation prefix other than /usr/local using --prefix, 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.

Step 6

Configure /etc/services

The step is optional. However, it is recommended. On the client (monitored) machines, add the following lines to /etc/services:

```
zabbix-agent    10050/tcp  Zabbix Agent
zabbix-agent    10050/udp  Zabbix Agent
zabbix-trapper  10051/tcp  Zabbix Trapper
zabbix-trapper  10051/udp  Zabbix Trapper
```

Note that the port numbers are official Zabbix ports registered in IANA.

Step 7

Configure /etc/inetd.conf

If you plan to use zabbix_agent instead of the recommended zabbix_agentd, the following line must be added:

```
zabbix_agent stream tcp nowait.3600 zabbix /opt/zabbix/bin/zabbix_agent
```

Restart inetd

```
shell> killall -HUP inetd
```

Modify default settings in configuration files

Step 8

Create a location to hold configuration files:

```
mkdir /etc/zabbix
```

Step 9

Configure /etc/zabbix/zabbix_agentd.conf

You need to configure this file for every host with zabbix_agentd installed. The file should contain the IP address of the Zabbix server. Connections from other hosts will be denied. You may take misc/conf/zabbix_agentd.conf as example.

Step 10

Configure /etc/zabbix/zabbix_server.conf

For small installations (up to ten monitored hosts), default parameters are sufficient. However, you should change default parameters to maximize performance of Zabbix. See section [Performance tuning] for more details. You may take misc/conf/zabbix_server.conf as example.

Step 11

Run server processes

Run zabbix_server on server side.

```
shell> cd sbin
shell> ./zabbix_server
```

Step 12

Run agents

Run zabbix_agentd where necessary.

```
shell> cd sbin
shell> ./zabbix_agentd
```

4.4 Zabbix Proxy

Zabbix Proxy is a special process. It is not required to run Zabbix.

Step 1

Create the Zabbix superuser account

This is the user the Proxy will run as. For production use you should create a dedicated unprivileged account ('zabbix' is commonly used). Running Zabbix Proxy as 'root', 'bin', or any other account with special rights is a security risk. Do not do it!

> **Note:**
> Zabbix Proxy process (zabbix_proxy) is protected from being run under root account.

Step 2

Extract Zabbix sources

```
shell> tar -zxf zabbix-1.8.tar.gz
```

Step 3

Create the Zabbix database. Optional. ::: noteclassic Zabbix Proxy process will create database automatically on the first run if it does not exist. It will use existing database otherwise. Database auto-creation is supported for SQLite only. ::: Zabbix comes with SQL scripts used to create the required database schema. There are separate scripts for IBM DB2, MySQL, Oracle, PostgreSQL and SQLite.

For IBM DB2:

```
shell> db2 "create database zabbix using codeset utf-8 territory us pagesize 32768"
shell> cd create/schema
shell> db2batch -d zabbix -f ibm_db2.sql
```

For MySQL:

```
shell> mysql -u<username> -p<password>
mysql> create database zabbix character set utf8;
mysql> quit;
shell> cd create/schema
shell> cat mysql.sql | mysql -u<username> -p<password> zabbix
```

For Oracle (we assume that user 'zabbix' with password 'password' exists and has permissions to create database objects):

```
shell> cd create/schema
shell> cat oracle.sql | sqlplus zabbix/password >out.log
```

**Note:**
Check file out.log for any error messages. Zabbix requires UTF8 database character set. If database is not UTF8 it can be converted by running: `ALTER DATABASE NATIONAL CHARACTER SET UTF8;`

For PostgreSQL:

```
shell> psql -U <username>
psql> create database zabbix;
psql> \q
shell> cd create/schema
shell> cat postgresql.sql | psql -U <username> zabbix
```

For SQLite:

```
shell> cd create/schema
shell> cat sqlite.sql | sqlite3 /var/lib/sqlite/zabbix.db
```

**Note:**
The database will be automatically created if it does not exist.

Step 4

Configure and compile the source code for your system

The sources must be compiled to enable compilation of Zabbix Proxy process. To configure the source for the Proxy, you must specify which database will be used.

```
shell> ./configure --enable-proxy --with-ibm-db2 --with-net-snmp # for IBM DB2 + SNMP monitoring
```

or

```
shell> ./configure --enable-proxy --with-mysql --with-net-snmp # for MySQL + SNMP monitoring
```

or

```
shell> ./configure --enable-proxy --with-oracle --with-net-snmp # for Oracle + SNMP monitoring
```

or

```
shell> ./configure --enable-proxy --with-pgsql --with-net-snmp # for PostgreSQL + SNMP monitoring
```

or

```
shell> ./configure --enable-proxy --with-sqlite3 --with-net-snmp # for SQLite3 + SNMP monitoring
```

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

**Note:**
Use flag --enable-static to statically link libraries. If you plan to distribute compiled binaries among different hosts, you must use this flag to make these binaries work without required libraries. --enable-static does not work under Solaris. Flag --with-ucd-snmp can be used instead of --with-net-snmp. If no SNMP support required, both --with-net-snmp and --with-ucd-snmp may be skipped.

However, if you want to compile client binaries along with proxy binaries, run:

```
shell> ./configure --enable-proxy --enable-agent --with-mysql --with-net-snmp
```

Parameter --enable-static may be used to force static linkage.

Step 5

Make and install everything

```
shell> make install
```

By default, *make install* will install all the files in /usr/local/sbin, /usr/local/lib etc. You can specify an installation prefix other than /usr/local using --prefix

Step 6

Configure /etc/services

The step is optional. However, it is recommended. On the client (monitored) machines, add the following lines to /etc/services:

```
zabbix_agent 10050/tcp
zabbix_trap 10051/tcp
```

Step 7

Configure /etc/inetd.conf

If you plan to use zabbix_agent instead of the recommended zabbix_agentd, the following line must be added:

```
zabbix_agent stream tcp nowait.3600 zabbix /opt/zabbix/bin/zabbix_agent
```

Restart inetd

```
shell> killall -HUP inetd
```

Step 8

Create a location to hold configuration files:

```
mkdir /etc/zabbix
```

Configure /etc/zabbix/zabbix_proxy.conf

For small installations (up to ten monitored hosts), default parameters are sufficient. However, you should change default parameters to maximize performance of Zabbix Proxy. Make sure you have correct Hostname and Server parameters set. You may take misc/conf/zabbix_proxy.conf as example.

Step 9

Run Proxy processes

Run zabbix_proxy:

```
shell> cd sbin
shell> ./zabbix_proxy
```

4.5 Zabbix Agent

Client side

Step 1

Create the Zabbix account

This is the user the agent will run as. For production use you should create a dedicated unprivileged account (“zabbix” is commonly used). Zabbix agents have protection against running under root account.

Step 2

Extract Zabbix sources

```
shell> tar -zxf zabbix-1.8.tar.gz
```

Step 3

Configure and compile the source code for your system

The sources must be compiled for the client only.

To configure the source for the client:

```
shell> ./configure --enable-agent
```

**Note:**
Use flag --enable-static to statically link libraries. If you plan to distribute compiled binaries among different hosts, you must use this flag to make these binaries work without required libraries.

Step 4

Build agent

```
shell> make
```

Copy created binaries from bin/ to /opt/zabbix/bin or any other directory. Other common directories are /usr/local/bin or /usr/local/zabbix/bin.

Step 5

Configure /etc/services

The step is not real requirement. However, it is recommended.

On the client (monitored) machines, add the following lines to /etc/services:

```
zabbix_agent 10050/tcp
zabbix_trap 10051/tcp
```

Step 6

Configure /etc/inetd.conf

If you plan to use zabbix_agent instead of the recommended zabbix_agentd, the following line must be added:

```
zabbix_agent stream tcp nowait.3600 zabbix /opt/zabbix/bin/zabbix_agent
```

Restart inetd

```
shell> killall -HUP inetd
```

Step 7

Create a location to hold configuration files:

```
mkdir /etc/zabbix
```

Step 8

Configure /etc/zabbix/zabbix_agentd.conf

You need to configure this file for every host with zabbix_agentd installed. The file should contain IP address of Zabbix server. Connections from other hosts will be denied. You may take misc/conf/zabbix_agentd.conf as example.

Step 9

Run zabbix_agentd on all monitored machines

```
shell> /opt/zabbix/bin/zabbix_agentd
```

**Note:**
You should not run zabbix_agentd if you have chosen to use zabbix_agent!

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

4.6 Zabbix WEB Interface

Step 0

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 into the webserver HTML documents directory. It is suggested to use a subdirectory instead of HTML root.

Common locations of the HTML documents directory for Apache web server 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)

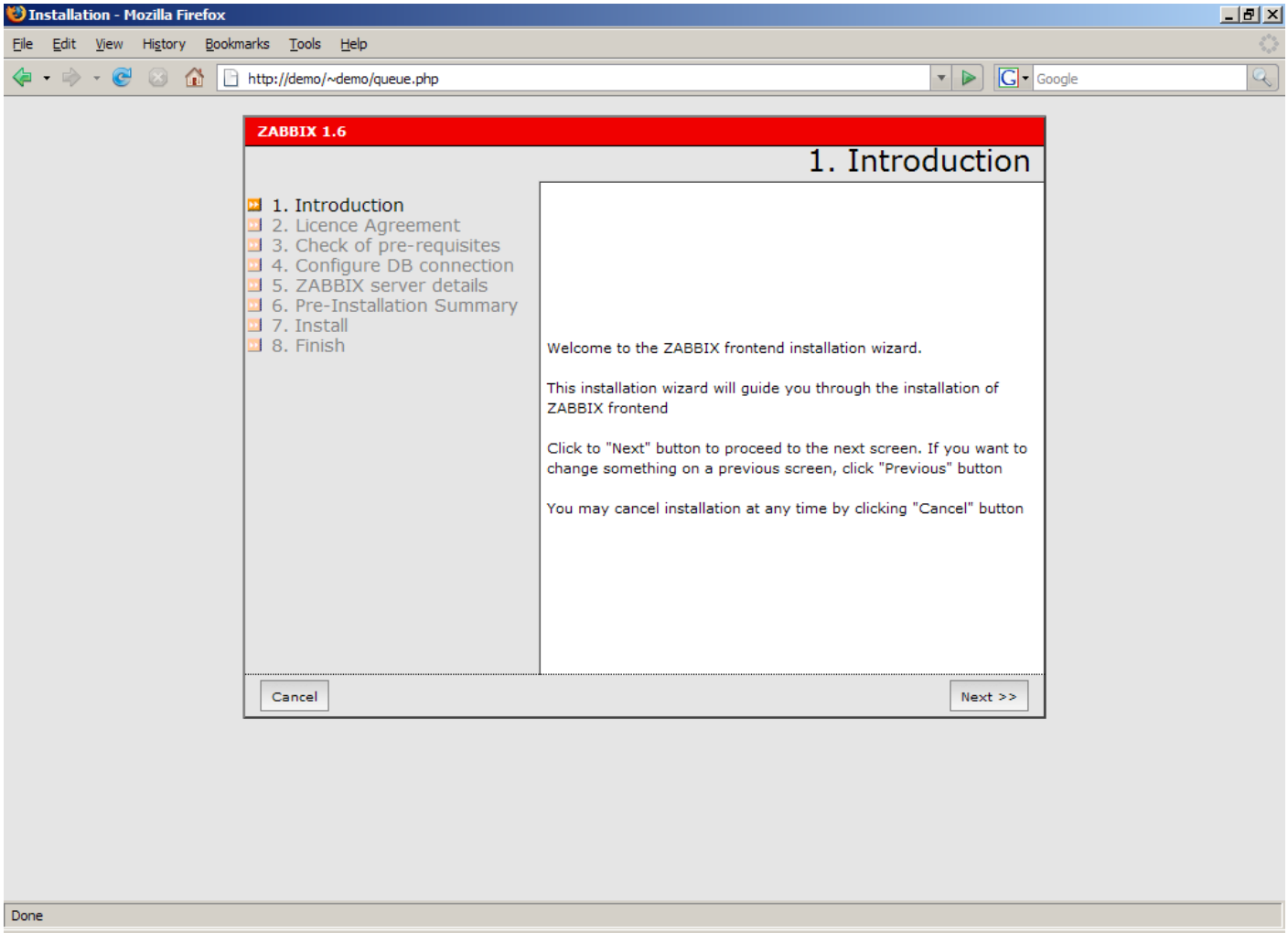
To create a subdirectory and copy Zabbix frontend files into it, execute the following commands, replacing <htdocs> with the correct path in your case:

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

**Attention:**
When upgrading you simply replace the content of <htdocs>/zabbix with the new files copied over from frontends/php, in this step.

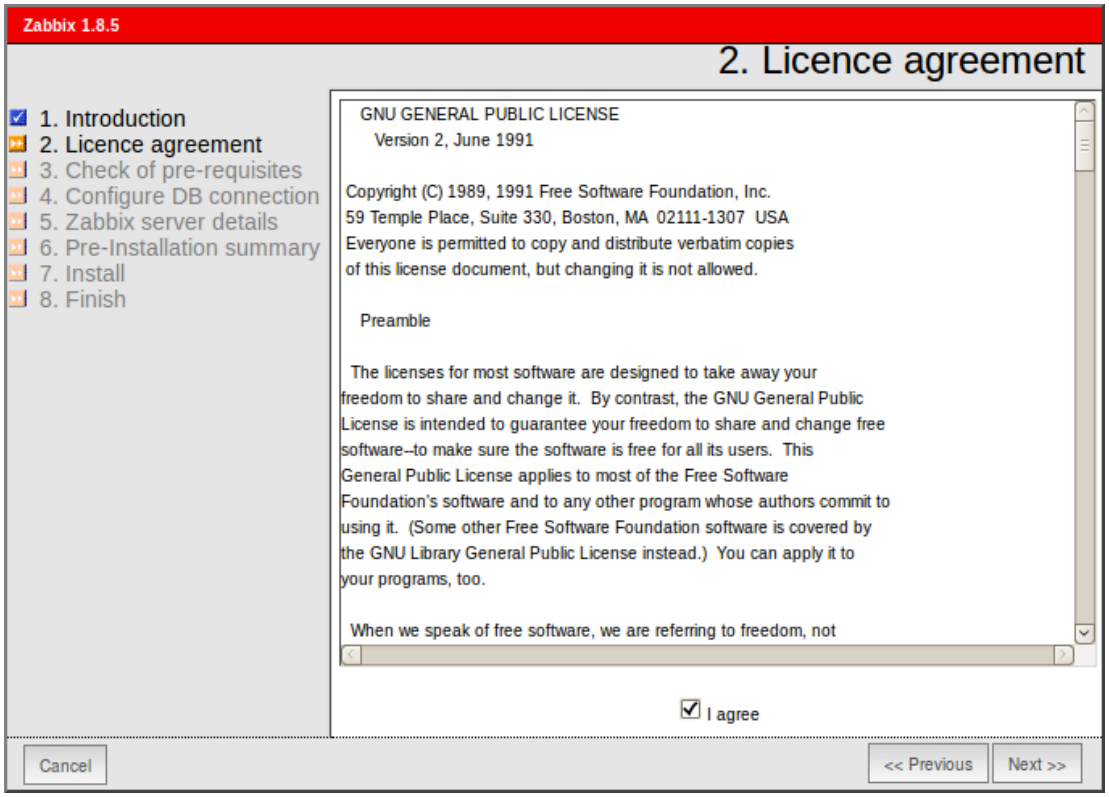
Step 1

Point your browser to Zabbix URL.



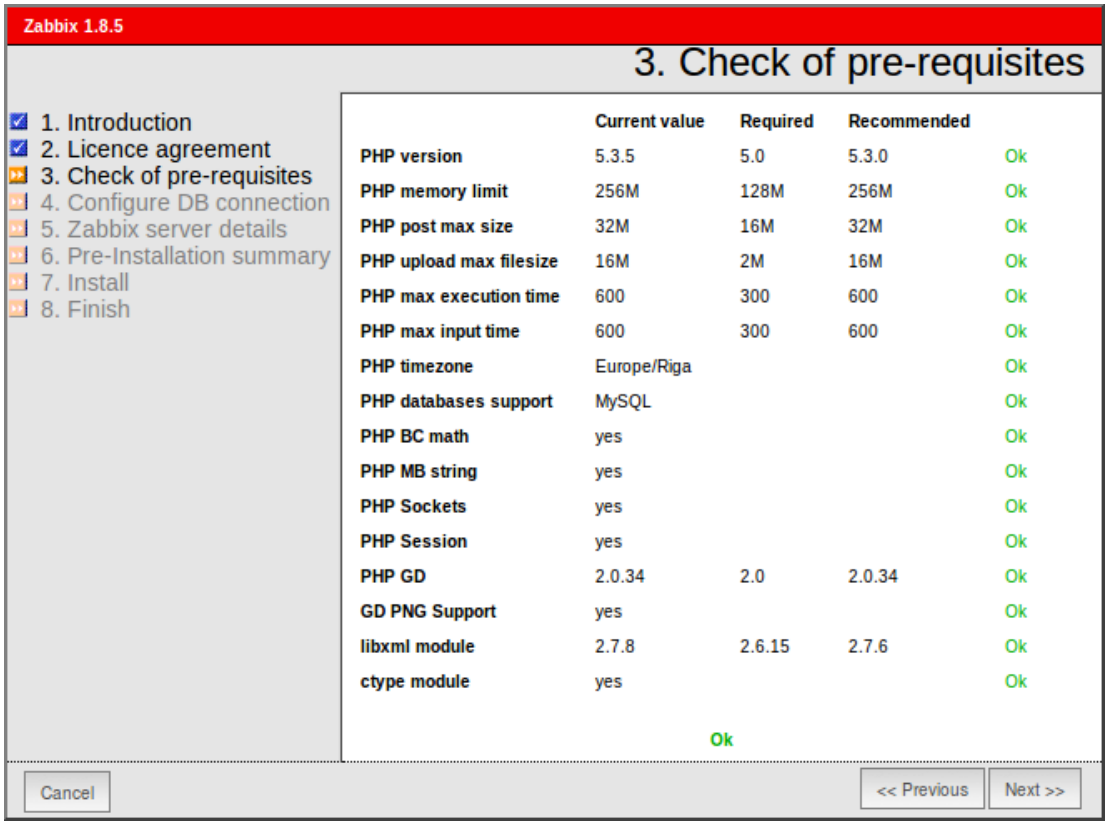

Step 2

Read and accept GPL v2.




Step 3

Make sure that all software pre-requisites are met.

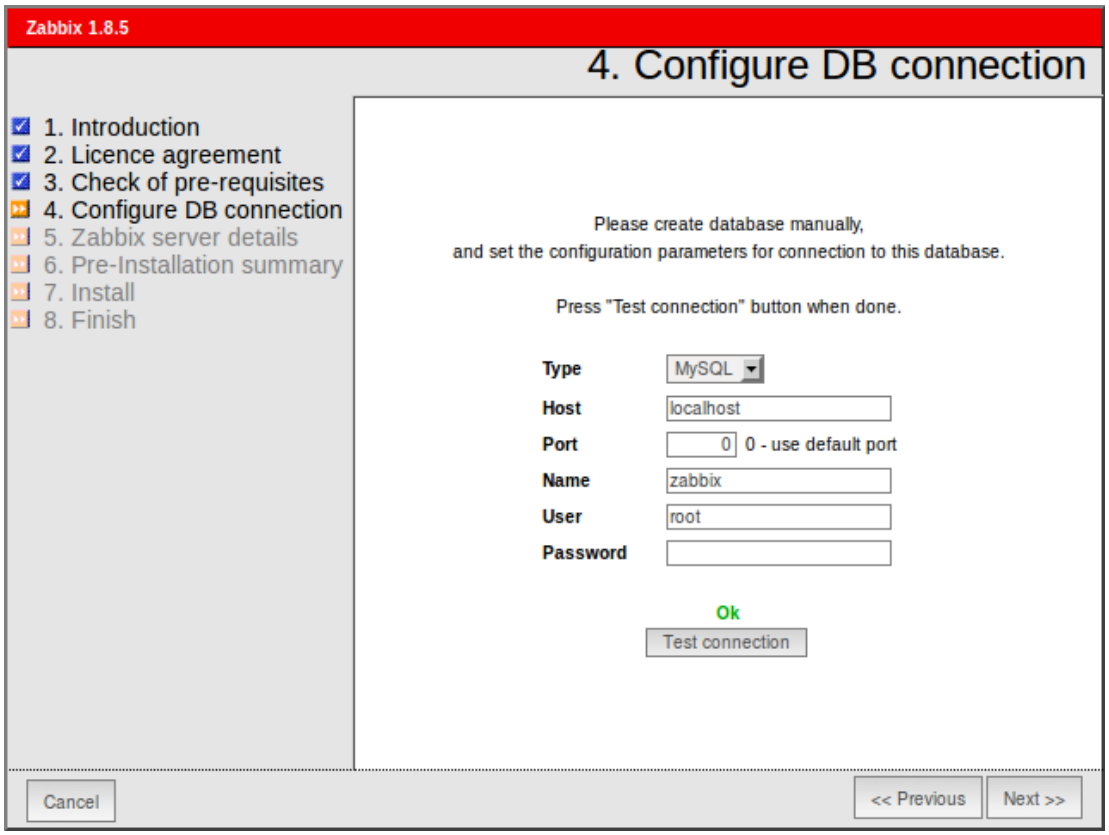



| Pre-requisite | Minimum value | Description |
|---|---|---|
| **PHP version** | 5.0 | |
| **PHP Memory limit** | 8MB | In php.ini:<br>memory_limit = 128M |
| **PHP post max size** | 8MB | In php.ini:<br>post_max_size = 16M |
| **PHP max execution time** | 300 seconds | In php.ini:<br>max_execution_time = 300 |

| Pre-requisite | Minimum value | Description |
|---|---|---|
| **PHP max input time** | 300 seconds | In php.ini:<br>max_input_time = 300 |
| **PHP database support** | One of: IBM DB2, MySQL, Oracle, PostgreSQL, SQLite | One of the following modules must be installed:<br>ibm_db2, php-mysql, oci8, php-pgsql, php-sqlite3 |
| **PHP BC math ** | Any | Compiled in or separate module php-bcmath. |
| **PHP multibyte support** | Any | Compiled in or separate module php-mbstring. |
| **GD Version** | 2.0 or higher | Module php-gd. |
| **Image formats** | At least PNG | Module php-gd. |

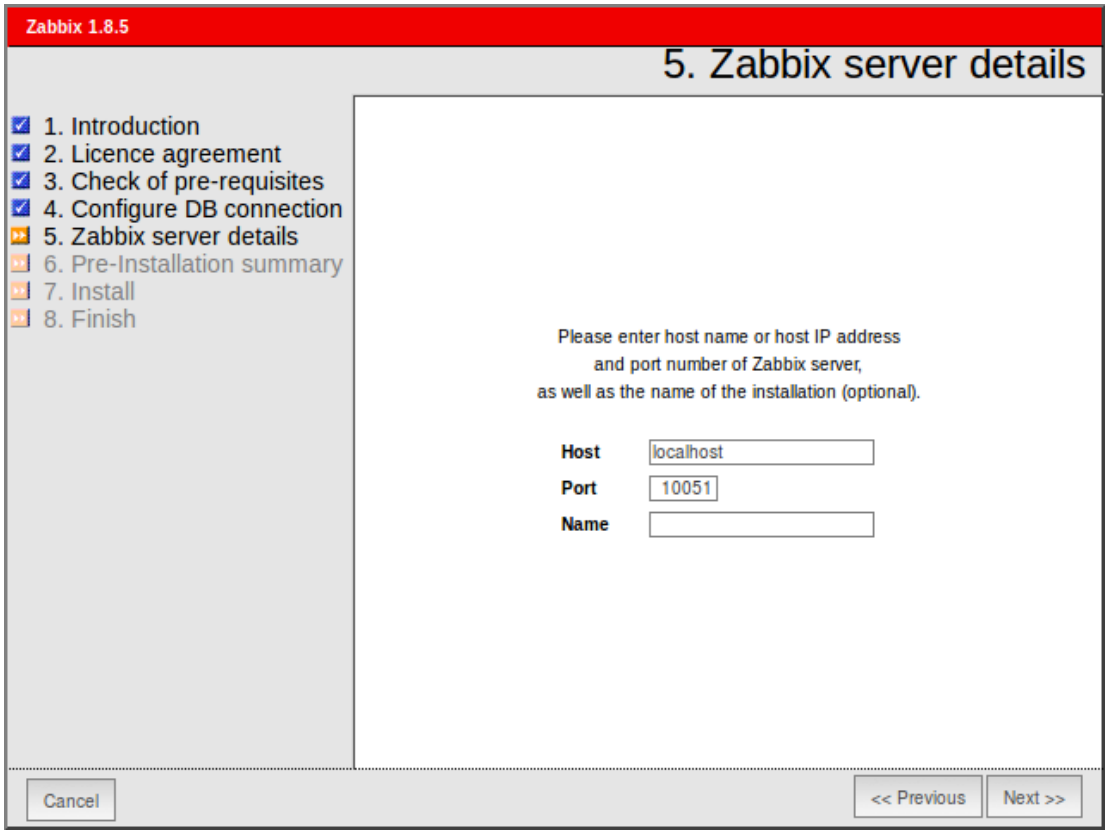
Step 4

Configure database settings. Zabbix database must already be created.



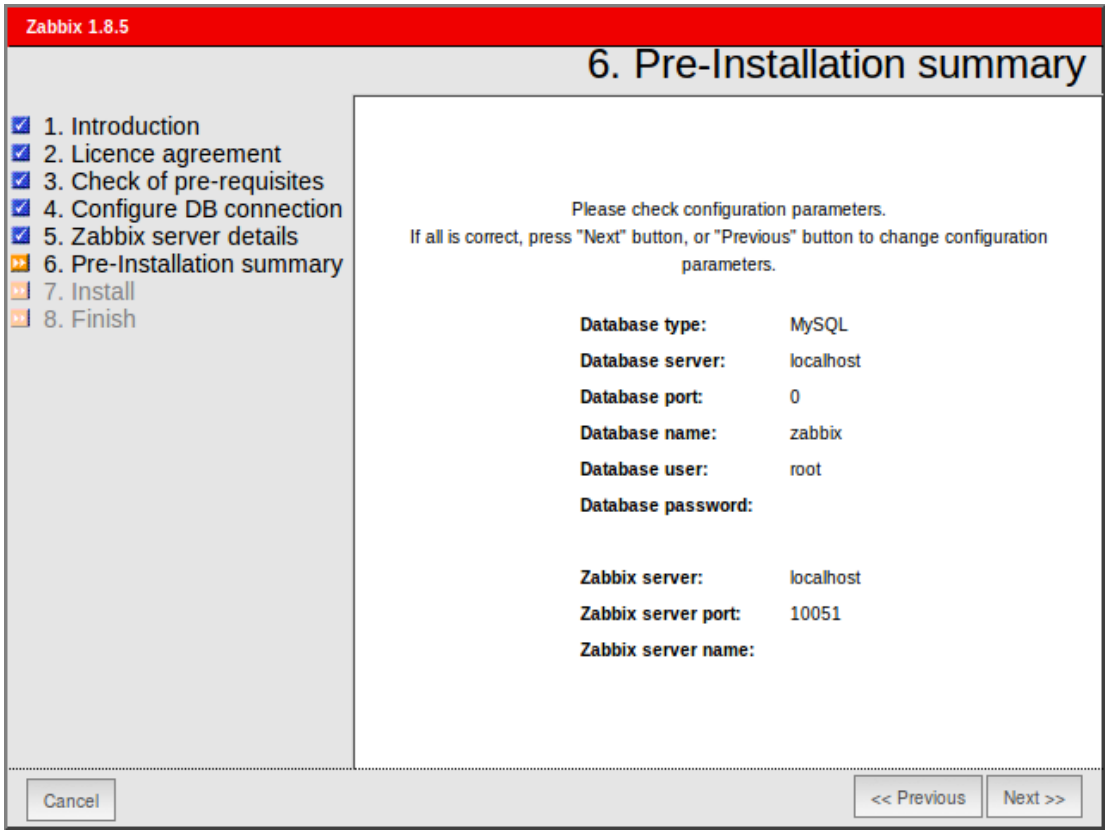

Step 5

Enter Zabbix Server details.



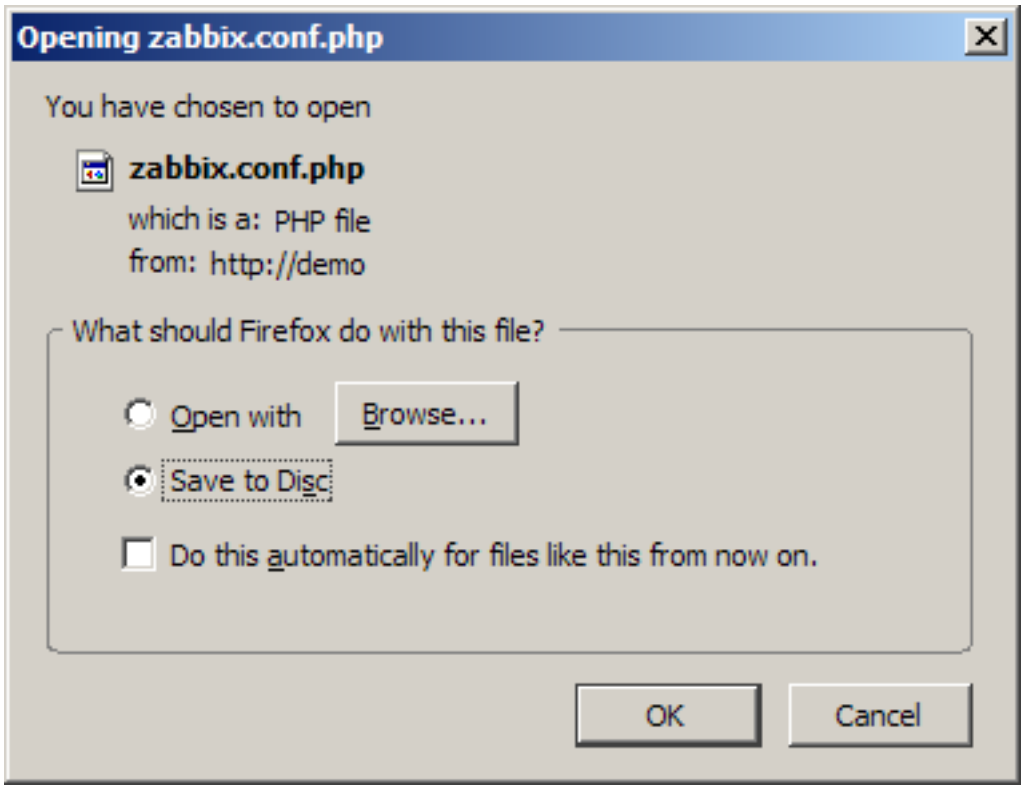

Step 6

See summary of settings.



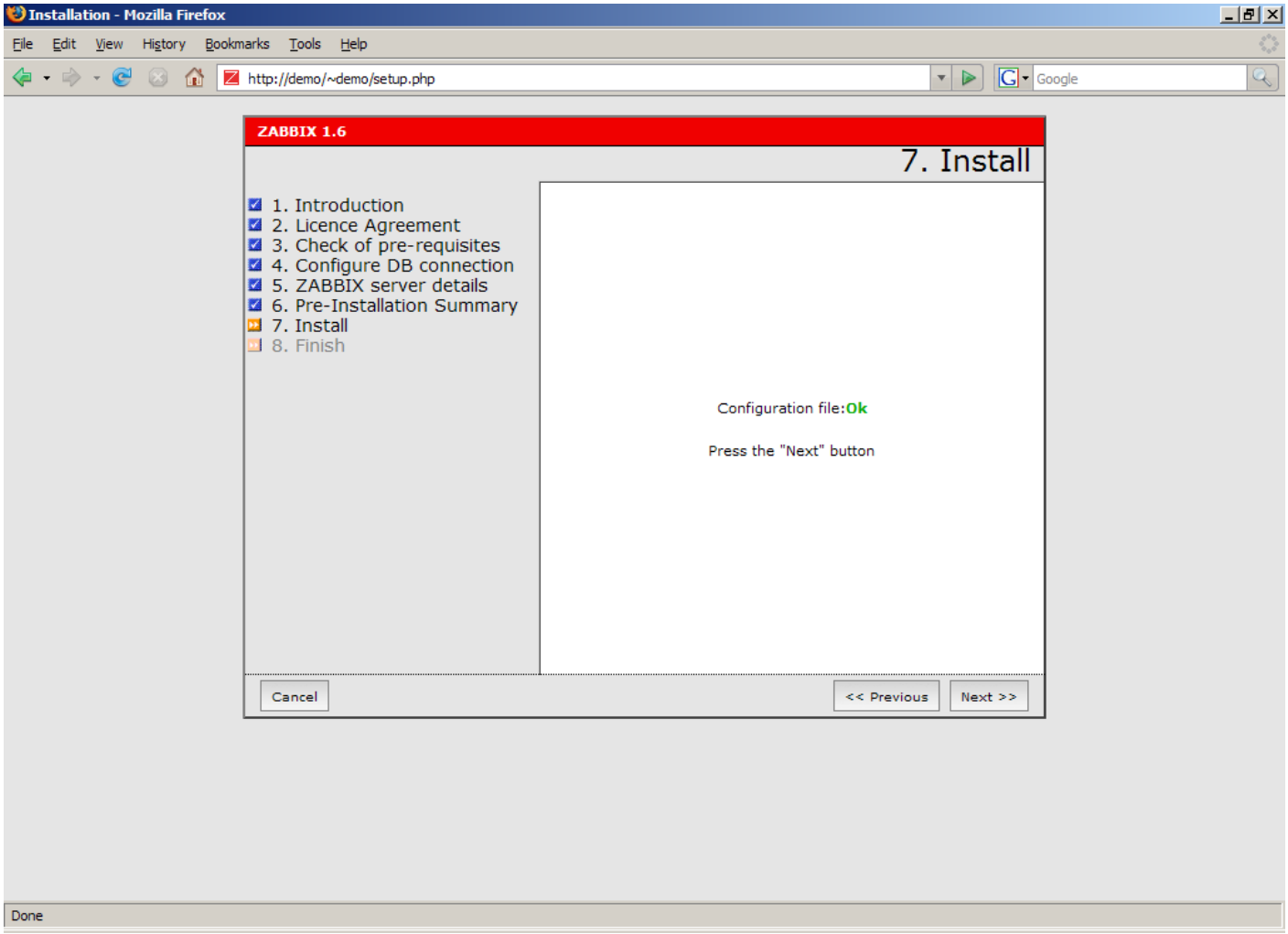

Step 7

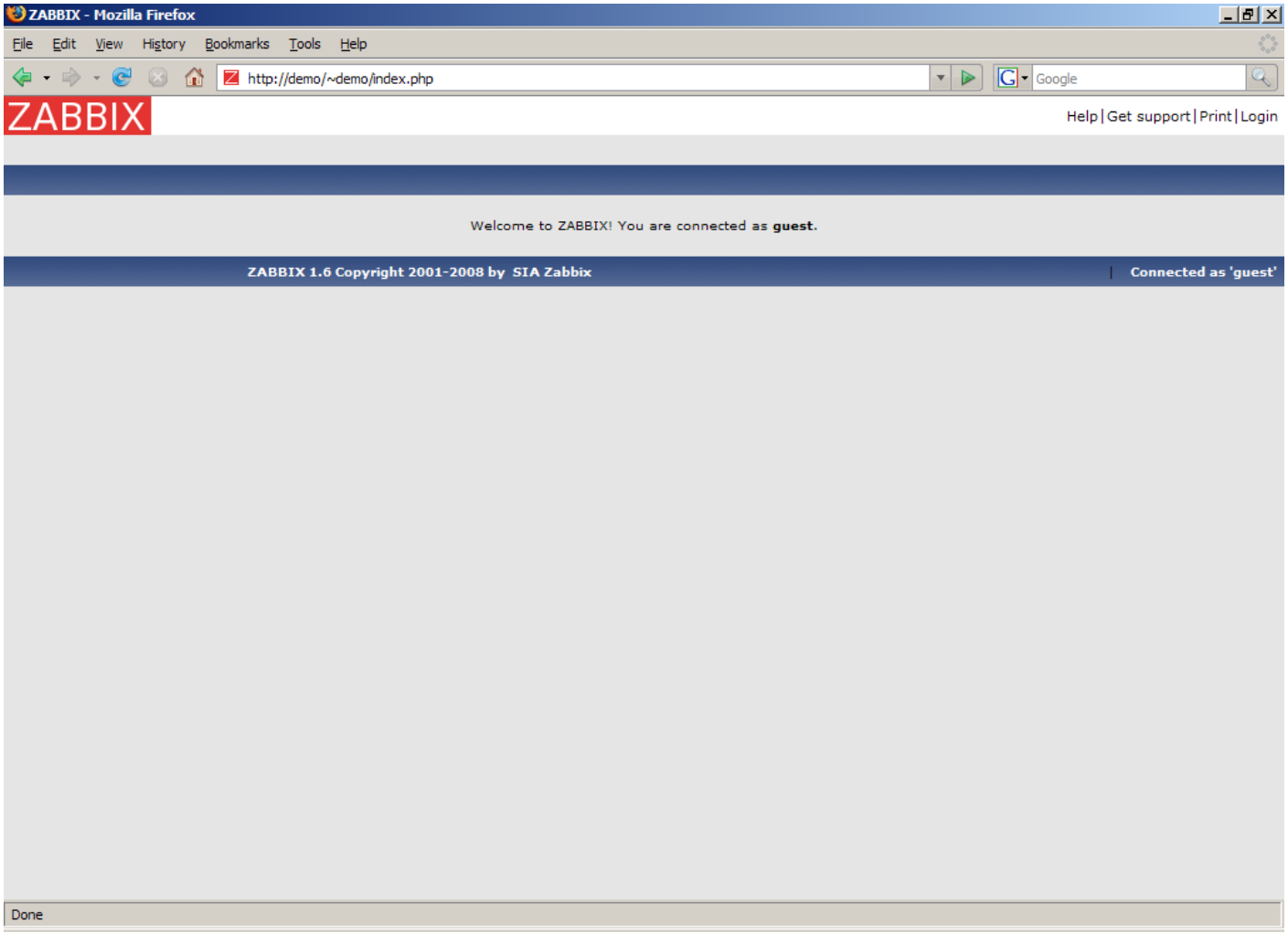
Download configuration file and place it under conf/.


Step 8

Finishing installation.


Installation - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://demo/~demo/setup.php
Google
ZABBIX 1.6
7. Install
1. Introduction
2. Licence Agreement
3. Check of pre-requisites
4. Configure DB connection
5. ZABBIX server details
6. Pre-Installation Summary
7. Install
8. Finish
Configuration file:Ok
Press the "Next" button
Cancel
<< Previous
Next >>
Done




Step 9

For distributed monitoring only!

If used in a distributed environment you have to run only once:

```
shell> ./zabbix_server -n <nodeid>
```

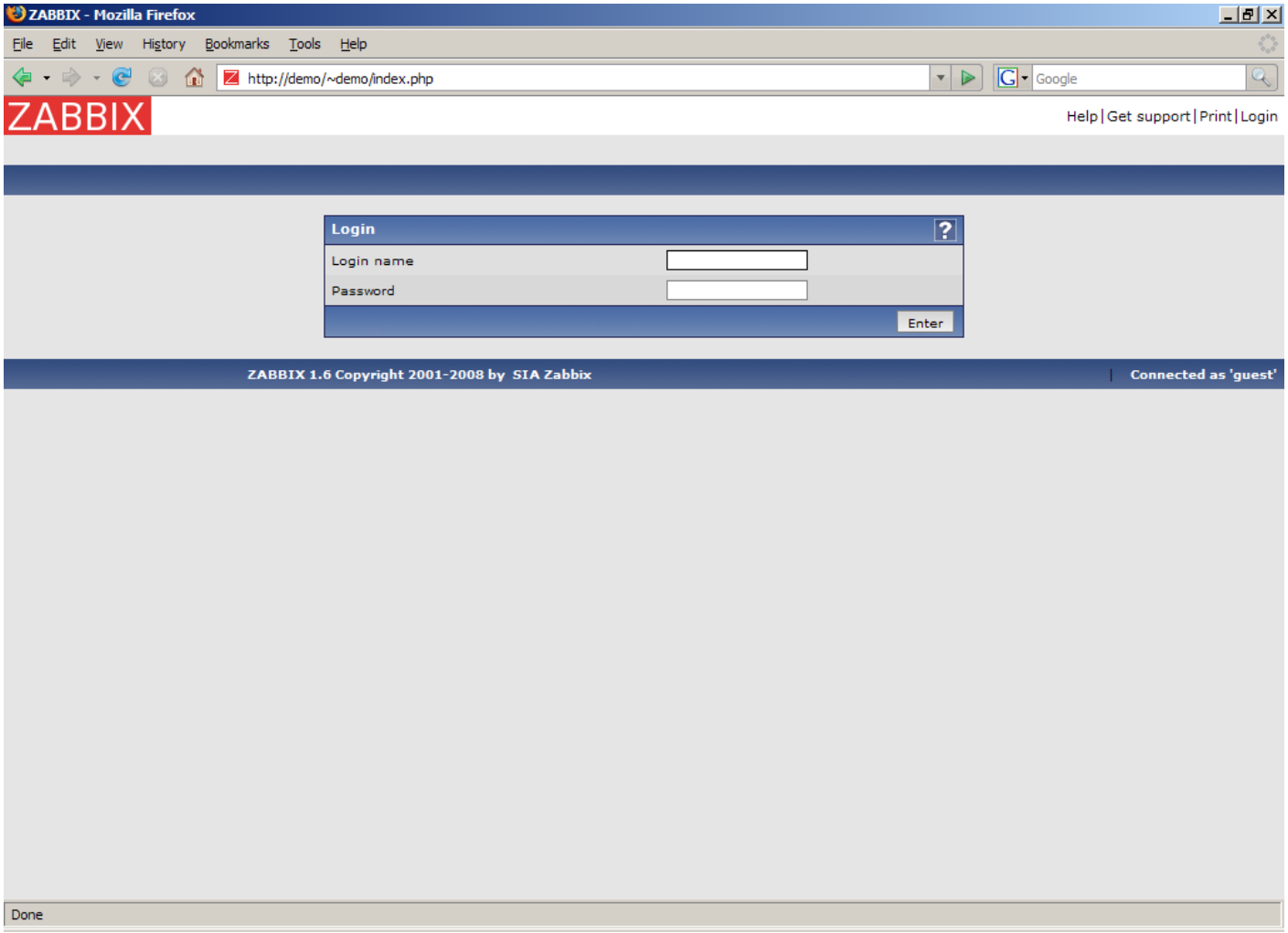
where Node ID is an unique Node identificator. For example:

```
shell> ./zabbix_server -n 1
```

This will convert database data for use with Node ID '1' and also adds a local node.

Step 10

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




## 5 Upgrading

### 5.1 Change level releases

For change level releases only upgrading of server binary and frontend is required. If mentioned in release notes, optional indexes may be added to the database to improve performance. Upgrading can be easily performed over several versions, for example, upgrading from 1.8.1 to 1.8.3 can be performed in single step.

See installation and upgrade notes for more information.

## 6 Using 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.

|<| |<| |-|

Zabbix appliance versions are based upon the following OpenSUSE versions:

| Zabbix appliance version | OpenSUSE version |
|---|---|
| 1.8.2 | 11.2 |
| 1.8.3 | |
| 1.8.4 | 11.3 |
| 1.8.5 | |
| 1.8.6 | |
| 1.8.7 | 11.4 |
| 1.8.8 | |

| Zabbix appliance version | OpenSUSE version |
|---|---|
| 1.8.9 | |
| 1.8.10 | |
| 1.8.11 | |
| 1.8.12 | |

It is available in the following formats:

- vmdk (VMWare/Virtualbox);
- OVF (Open Virtualisation Format);
- CD iso;
- HDD/flash image;
- Preload ISO;
- Xen guest.

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

The appliance has been built using SUSE Studio.

6.1 Changes to SUSE configuration

There are some changed applied to the base OpenSUSE configuration.

6.1.1 MySQL configuration changes

- Binary log is disabled;
- InnoDB is configured to store data for each table in a separate file.

6.1.2 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-eth0* 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.

6.1.3 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
```

6.1.4 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.

6.2 Zabbix configuration

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

6.2.1 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 → WEB*).

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

- MySQL;
- zabbix_server.conf;
- zabbix.conf.php.

6.2.2 File locations

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

6.2.3 Changes to Zabbix configuration

- Some items and triggers in the default Linux template are disabled (mostly those who did not correspond to appliance setup);
- Server name for Zabbix frontend set to "Zabbix 1.8 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.

6.2.4 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.

6.3 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
```

6.4 Firewall

By default, only two ports are open - 22 (SSH) and 80 (HTTP). 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 stop
SuSEfirewall2 start
```

6.5 Monitoring capabilities

Zabbix server is compiled with support for the following:

- SNMP;
- IPMI;
- Web monitoring;
- 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.

6.6 Naming, init and other scripts

Zabbix daemons have their names changed from standard with underscore to dash to conform to SUSE guidelines. They are called:

- zabbix-agentd
- zabbix-server

In a similar fashion, configuration files are:

- /etc/zabbix/zabbix-server.conf
- /etc/zabbix/zabbix-agentd.conf

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

```
service zabbix-server status
rczabbix-server status
/etc/init.d/zabbix-server status
```

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

6.6.1 Scheduled scripts

There is a scheduled script, run from the crontab every 10 minutes that restarts Zabbix server if it is not running, **/var/lib/zabbix/bin**. It logs timestamped problems and starting attempts at **/var/log/zabbix/server_problems.log**. This script is available since Zabbix Appliance version 1.8.3.

> **Attention:**
> Make sure to disable this crontab entry if stopping of Zabbix server is desired.

6.6.2 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.

6.7 Format-specific notes

6.7.1 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.*

6.7.2 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.

6.7.3 HDD/flash image (raw)

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

6.8 Known issues

6.8.1 For appliance 1.8.8

Zabbix appliance 1.8.8 reports itself as being based on 1.8.7 in the boot messages. This is incorrect, actual appliance contains Zabbix 1.8.8.

## 3 Zabbix Processes

**1 Logging** For logging configuration of Zabbix daemons "LogFile" configuration parameter is used. If this parameter is left empty (LogFile=), syslog logging facilities are used. All Zabbix daemons on Unix-like platforms log their messages from "Daemon" environment. The mapping between Zabbix logging levels and syslog levels is as follows:

| Zabbix log level | syslog log level | Comments |
|---|---|---|
| 0 - empty (LOG_LEVEL_EMPTY) | syslog is not used. | All messages are skipped. |

| Zabbix log level | syslog log level | Comments |
|---|---|---|
| 1 - critical information (LOG_LEVEL_CRIT) | critical conditions (LOG_CRIT) | |
| 2 - error information (LOG_LEVEL_ERR) | error conditions (LOG_ERR) | |
| 3 - warnings (LOG_LEVEL_WARNING) | warning conditions (LOG_WARNING) | |
| 4 - for debugging (LOG_LEVEL_DEBUG) | debug-level messages (LOG_DEBUG) | |

For syslog configuration consult the corresponding literature.

Zabbix agent under Windows uses Event Log if "LogFile" configuration parameter is provided empty. Mapping between Zabbix log levels (messages of corresponding type) and Windows Event Log entries type is provided below:

| Zabbix log level | Windows Event Log entry type | Comments |
|---|---|---|
| 0 - empty (LOG_LEVEL_EMPTY) | Event Log is not used. | All messages are skipped. |
| 1 - critical information (LOG_LEVEL_CRIT) | EVENTLOG_ERROR_TYPE | Error |
| 2 - error information (LOG_LEVEL_ERR) | | |
| 3 - warnings (LOG_LEVEL_WARNING) | EVENTLOG_WARNING_TYPE | Warning |
| 4 - for debugging (LOG_LEVEL_DEBUG) | EVENTLOG_INFORMATION_TYPE | Information |

**2 Individual processes** zabbix_server zabbix_proxy zabbix_agentd zabbix_agent zabbix_agentd_win zabbix_sender zabbix_get

**1 Zabbix Server**

Zabbix server is the central process of Zabbix software. Zabbix server can be started by executing:

```
shell> cd sbin
shell> ./zabbix_server
```

Zabbix server runs as a daemon process.

Zabbix server accepts the following command line parameters:

```
-c --config <file>              absolute path to the configuration file (default is /etc/zabbix/zabbix_ser
-n --new-nodeid <nodeid>        convert database data to new nodeid
-R --runtime-control <option>   perform administrative functions
-h --help                       give this help
-V --version                    display version number
```

**Note:**
-R or --runtime-control option is supported since Zabbix 1.8.6.
Runtime control is not supported on OpenBSD and NetBSD.

In order to get more help run:

```
shell> zabbix_server -h
```

Example of 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 |
|---|---|
| config_cache_reload | Reload configuration cache. Ignored if cache is being currently loaded. |

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
```

**Configuration file** The configuration file contains parameters for zabbix_server. The file must exist and it should have read permissions for user 'zabbix'. Supported parameters:

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| AlertScriptsPath | no | | /home/zabbix/bin/ | Location of custom alert scripts |
| CacheSize | no | 128K-1G | 8M | Size of configuration cache, in bytes.<br>Shared memory size for storing hosts and items data. |
| CacheUpdateFrequency | no | 1-3600 | 60 | How often Zabbix will perform update of configuration cache, in seconds. |
| DBHost | no | | Based on the underlying library implementation used. | Database host name.<br>If set to localhost, socket is used for MySQL. |
| DBName | yes | | | Database name.<br>For SQLite3 path to database file must be provided.<br>DBUser and DBPassword are ignored. |
| DBPassword | no | | | Database password. Ignored for SQLite.<br>Comment this line if no password is used. |
| DBPort | no | 1024-65535 | 3306 | Database port when not using local socket. Ignored for SQLite. |
| DBSocket | no | | /tmp/mysql.sock | Path to MySQL socket. |
| DBUser | no | | | Database user. Ignored for SQLite. |
| DebugLevel | no | 0-4 | 3 | Specifies debug level<br>0 - no debug<br>1 - critical information<br>2 - error information<br>3 - warnings<br>4 - for debugging (produces lots of information) |
| DisableHousekeeping | no | 0-1 | 0 | If set to 1, disables housekeeping. |
| ExternalScripts | no | | /etc/zabbix/externalscripts | Location of external scripts |
| Fping6Location | no | | /usr/sbin/fping6 | Location of fping6.<br>Make sure that fping6 binary has root ownership and SUID flag set.<br>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.<br>Make sure that fping binary has root ownership and SUID flag set! |
| HistoryCacheSize | no | 128K-1G | 8M | Size of history cache, in bytes.<br>Shared memory size for storing history data. |
| HistoryTextCacheSize | no | 128K-1G | 16M | Size of text history cache, in bytes.<br>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).<br>Housekeeping is removing unnecessary information from history, alert, and alarms tables.<br>*Note*: To prevent housekeeper from being overloaded (for example, when history and trend periods are greatly reduced), no more than 4xHousekeepingFrequency hours of outdated history are deleted in one housekeeping cycle, for each item. Thus, if HousekeepingFrequency is 1, no more than 4 hours of outdated history (starting from the oldest entry) will be deleted per cycle. |
| Include | no | | | You may include individual files or all files in a directory in the configuration file. See special notes about limitations. |
| ListenIP | no | | 0.0.0.0 | List of comma delimited IP addresses that the trapper should listen on.<br>Trapper will listen on all network interfaces if this parameter is missing.<br>**Multiple IP addresses are supported in version 1.8.3 and higher.** |
| ListenPort | no | 1024-32767 | 10051 | Listen port for trapper. |
| LogFileSize | no | 0-1024 | 1 | Maximum size of log file in MB.<br>0 - disable automatic log rotation. |
| LogFile | no | | | Name of log file. |

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| LogSlowQueries | no | 0-3600000 | 0 | How long a database query may take before being logged (in milliseconds).<br>0 - don't log slow queries.<br>This option becomes enabled starting with DebugLevel=3.<br>**This option is supported in version 1.8.2 and higher.** |
| 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.<br>SQLite3 does not use this parameter, deletes all corresponding rows without a limit.<br>If set to 0 then no limit is used at all. In this case you must know what you are doing!<br>This parameter is supported since **Zabbix 1.8.2** and applies only to deleting history and trends of already deleted items. |
| NodeID | no | 0-999 | 0 | Unique NodeID in distributed setup.<br>0 - standalone server |
| NodeNoEvents | no | 0-1 | 0 | If set to '1' local events won't be sent to master node.<br>This won't impact ability of this node to propagate events from its child nodes. |
| NodeNoHistory | no | 0-1 | 0 | If set to '1' local history won't be sent to master node.<br>This won't impact ability of this node to propagate history from its child nodes. |
| 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.<br>Used only for proxies in a passive mode.<br>**This option is supported in version 1.8.3 and higher.** |
| ProxyDataFrequency | no | 1-3600 | 1 | How often Zabbix Server requests history data from a Zabbix Proxy in seconds.<br>Used only for proxies in a passive mode.<br>**This option is supported in version 1.8.3 and higher.** |
| SSHKeyLocation | no | | | Location of public and private keys for SSH checks |

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| SenderFrequency | no | 5-3600 | 30 | How often Zabbix will try to send unsent alerts (in seconds). |
| SourceIP | no | | | Source IP address for outgoing connections. |
| StartDBSyncers | no | 1-100 | 4 | Number of pre-forked instances of DB Syncers.<br>The upper limit used to be 64 before version 1.8.5.<br>**This option is supported in version 1.8.3 and higher.** |
| StartDiscoverers | no | 0-250 | 1 | Number of pre-forked instances of discoverers.<br>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.<br>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.<br>The upper limit used to be 255 before version 1.8.5. |
| StartPingers | no | 0-1000 | 1 | Number of pre-forked instances of ICMP pingers.<br>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).<br>The upper limit used to be 255 before version 1.8.5.<br>**This option is missing in version 1.8.3.** |
| StartPollers | no | 0-1000 | 5 | Number of pre-forked instances of pollers.<br>The upper limit used to be 255 before version 1.8.5. |
| StartProxyPollers | no | 0-250 | 1 | Number of pre-forked instances of pollers for passive proxies.<br>The upper limit used to be 255 before version 1.8.5.<br>**This option is supported in version 1.8.3 and higher.** |
| StartTrappers | no | 0-1000 | 5 | Number of pre-forked instances of trappers.<br>The upper limit used to be 255 before version 1.8.5. |
| 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-1G | 4M | Size of trend cache, in bytes.<br>Shared memory size for storing trends data. |

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| 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. |

**Note:**
Starting from version 1.8.6 Zabbix Server will not start up if invalid (not following *parameter=value* notation) or unknown parameter entry is present in configuration file.

**Note:**
Zabbix supports configuration files only in UTF-8 encoding without BOM.

**2 Zabbix Proxy**

Zabbix proxy is a process which collects performance and availability data from one or more monitored devices and sends the information to a Zabbix server. Zabbix proxy can be started by:

```
shell> cd sbin
shell> ./zabbix_proxy
```

Zabbix proxy runs as a daemon process.

Zabbix proxy accepts the following command line parameters:

```
-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:**
-R or --runtime-control option is supported since Zabbix 1.8.6.
Runtime control is not supported on OpenBSD and NetBSD.

In order to get more help run:

```
shell> zabbix_proxy -h
```

Example of 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 |
|---|---|
| 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. |

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
```

**Configuration file** The configuration file contains parameters for zabbix_proxy. The file must exist and it should have read permissions for user 'zabbix'. Supported parameters:

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| CacheSize | no | 128K-1G | 8M | Size of configuration cache, in bytes.<br>Shared memory size, for storing hosts and items data. |
| ConfigFrequency | no | 1-604800 | 3600 | How often proxy retrieves configuration data from Zabbix Server in seconds.<br>For a proxy in the passive mode this parameter will be ignored. |
| DBHost | no | | Based on the underlying library implementation used. | Database host name.<br>If set to localhost, socket is used for MySQL. |
| DBName | yes | | | Database name.<br>For SQLite3 path to database file must be provided.<br>DBUser and DBPassword are ignored. |
| DBPassword | no | | | Database password. Ignored for SQLite.<br>Comment this line if no password is used. |
| DBSocket | no | | 3306 | Path to MySQL socket.<br>Database port when not using local socket. Ignored for SQLite. |
| DBUser | | | | Database user. Ignored for SQLite. |
| DataSenderFrequency | no | 1-3600 | 1 | Proxy will send collected data to the Server every N seconds. |
| DebugLevel | no | 0-4 | 3 | Specifies debug level<br>0 - no debug<br>1 - critical information<br>2 - error information<br>3 - warnings<br>4 - for debugging (produces lots of information) |
| DisableHousekeeping | no | 0-1 | 0 | If set to 1, disables housekeeping. |
| ExternalScripts | no | | /etc/zabbix/externalscripts | Location of external scripts |
| Fping6Location | no | | /usr/sbin/fping6 | Location of fping6.<br>Make sure that fping6 binary has root ownership and SUID flag set.<br>Make empty (”Fping6Location=”) if your fping utility is capable to process IPv6 addresses. |
| FpingLocation | no | | /usr/sbin/fping | Location of fping.<br>Make sure that fping binary has root ownership and SUID flag set! |

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| HeartbeatFrequency | no | 0-3600 | 60 | Frequency of heartbeat messages in seconds.<br>Used for monitoring availability of Proxy on server side.<br>0 - heartbeat messages disabled.<br>For a proxy in the passive mode this parameter will be ignored. |
| HistoryCacheSize | no | 128K-1G | 8M | Size of history cache, in bytes.<br>Shared memory size for storing history data. |
| HistoryTextCacheSize | no | 128K-1G | 16M | Size of text history cache, in bytes.<br>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!<br>Allowed characters: alphanumeric, '.', ' ', '_' and '-'.<br>Maximum length: 64 |
| 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[].<br><br>Ignored if Hostname is set.<br><br>**This option is supported in version 1.8.6 and higher.** |

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| HousekeepingFrequency | no | 1-24 | 1 | How often Zabbix will perform housekeeping procedure (in hours).<br>Housekeeping is removing unnecessary information from history, alert, and alarms tables.<br>*Note*: To prevent housekeeper from being overloaded (for example, when history and trend periods are greatly reduced), no more than 4xHousekeepingFrequency hours of outdated history are deleted in one housekeeping cycle, for each item. Thus, if HousekeepingFrequency is 1, no more than 4 hours of outdated history (starting from the oldest entry) will be deleted per cycle. |
| Include | no | | | You may include individual files or all files in a directory in the configuration file. See special notes about limitations. |
| ListenIP | no | | 0.0.0.0 | List of comma delimited IP addresses that the trapper should listen on.<br>Trapper will listen on all network interfaces if this parameter is missing.<br>**Multiple IP addresses are supported in version 1.8.3 and higher.** |
| ListenPort | no | 1024-32767 | 10051 | Listen port for trapper. |
| LogFileSize | no | 0-1024 | 1 | Maximum size of log file in MB.<br>0 - disable automatic log rotation. |
| LogFile | no | | | Name of log file.<br>If not set, syslog is used. |
| LogSlowQueries | no | 0-3600000 | 0 | How long a database query may take before being logged (in milliseconds).<br>0 - don't log slow queries.<br>This option becomes enabled starting with DebugLevel=3.<br>**This option is supported in version 1.8.2 and higher.** |
| PidFile | no | | /tmp/zabbix_proxy.pid | Name of PID file. |
| ProxyLocalBuffer | no | 0-720 | 0 | Proxy will keep data locally for N hours.<br>This parameter may be used if local data will be used by third party applications. |

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| ProxyMode | no | 0-1 | 0 | Proxy operating mode.<br>0 - proxy in the active mode<br>1 - proxy in the passive mode<br>**This option is supported in version 1.8.3 and higher.** |
| ProxyOfflineBuffer | no | 1-720 | 1 | Proxy will keep data for N hours in case of no connectivity with Zabbix Server.<br>Older data will be lost. |
| SSHKeyLocation | no | | | Location of public and private keys for SSH checks |
| ServerPort | no | 1024-32767 | 10051 | Port of Zabbix trapper on Zabbix server.<br>For a proxy in the passive mode this parameter will be ignored. |
| Server | yes | | | IP address (or hostname) of Zabbix server.<br>Active Proxy will get configuration data from the server.<br>For a proxy in the passive mode this parameter will be ignored. |
| SourceIP | no | | | Source IP address for outgoing connections. |
| StartDBSyncers | no | 1-100 | 4 | Number of pre-forked instances of DB Syncers.<br>The upper limit used to be 64 before version 1.8.5.<br>**This option is supported in version 1.8.3 and higher.** |
| StartDiscoverers | no | 0-250 | 1 | Number of pre-forked instances of discoverers.<br>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.<br>The upper limit used to be 255 before version 1.8.5. |
| StartPingers | no | 0-1000 | 1 | Number of pre-forked instances of ICMP pingers.<br>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).<br>The upper limit used to be 255 before version 1.8.5.<br>**This option is missing in version 1.8.3.** |
| StartPollers | no | 0-1000 | 5 | Number of pre-forked instances of pollers.<br>The upper limit used to be 255 before version 1.8.5. |

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| StartTrappers | no | 0-1000 | 5 | Number of pre-forked instances of trappers. The upper limit used to be 255 before version 1.8.5. |
| 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. |
| 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. |

**Note:**
Starting from version 1.8.6 Zabbix Proxy will not start up if invalid (not following *parameter=value* notation) or unknown parameter entry is present in configuration file.

**Note:**
Zabbix supports configuration files only in UTF-8 encoding without BOM.

#### 3 Zabbix Agent (UNIX, Standalone daemon)

Zabbix UNIX agent daemon runs on a host being monitored. The agent provides host's performance and availability information for Zabbix Server. Zabbix agent processes items of type 'Zabbix agent' or 'Zabbix agent (active)'.

Zabbix agent can be started by executing:

```
shell> cd sbin
shell> ./zabbix_agentd
```

Zabbix agent runs as a daemon process.

Zabbix agent accepts the following command line parameters:

```
  -c --config <file>   specify configuration file, default is /etc/zabbix/zabbix_agentd.conf
  -h --help            give this help
  -V --version         display version number
  -p --print           print known items and exit
  -t --test <item key> test specified item and exit
```

In order to get this help run:

```
shell> zabbix_agentd -h
```

Example of command line parameters:

```
shell> zabbix_agentd -c /usr/local/etc/zabbix_agentd.conf
shell> zabbix_agentd --help
shell> zabbix_agentd --print
shell> zabbix_agentd -t "system.cpu.load[all,avg1]"
```

**Configuration file** The configuration file contains configuration parameters for zabbix_agentd. The file must exist and it should have read permissions for user 'zabbix'. Supported parameters:

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| Alias | no | | | Sets an alias for parameter. It can be useful to substitute long and complex parameter name with a smaller and simpler one.<br><br>Starting from version 1.8.6 Zabbix Agent will not start up in case incorrectly formatted Alias entry or duplicate Alias key is present in configuration file. |
| 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.<br>0 - do not allow<br>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<br>all collected data to Zabbix Server or Proxy if the buffer is full. |
| DebugLevel | no | 0-4 | 3 | Specifies debug level<br>0 - no debug<br>1 - critical information<br>2 - error information<br>3 - warnings<br>4 - for debugging (produces lots of information) |
| DisableActive | no | | 0 | Disable active checks. The agent will work in passive mode listening for server. |
| DisablePassive | no | | 0 | Disable passive checks. The agent will not listen on any TCP port.<br>Only active checks will be processed.<br>0 - do not disable<br>1 - disable |
| EnableRemoteCommands | no | | 0 | Whether remote commands from Zabbix server are allowed.<br>0 - not allowed<br>1 - allowed |

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| Hostname | no | | Set by HostnameItem | Unique, case sensitive hostname.<br>Required for active checks and must match hostname as configured on the server.<br>Allowed characters: alphanumeric, '.', ' ', '_' and '-'.<br>Maximum length: 64 |
| HostnameItem | no | | system.hostname | Item used for setting Hostname if it is undefined.<br>Does not support UserParameters, performance counters or aliases, but does support system.run[] regardless of EnableRemoteCommands value.<br><br>Ignored if Hostname is set.<br><br>**This option is supported in version 1.8.6 and higher.** |
| Include | no | | | You may include individual files or all files in a directory in the configuration file. See special notes about limitations. |
| ListenIP | no | | 0.0.0.0 | List of comma delimited IP addresses that the agent should listen on.<br>**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. |
| LogFileSize | no | 0-1024 | 1 | Maximum size of log file in MB.<br>0 - disable automatic log rotation. |
| LogFile | no | | | Name of log file.<br>If not set, syslog is used. |
| LogRemoteCommands | no | | 0 | Enable logging of executed shell commands as warnings.<br>0 - disabled<br>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 processing 'log' and 'eventlog' active checks. The provided value will be overridden by the parameter 'maxlines', provided in 'log' or 'eventlog' item key.<br>*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. |
| Server | yes | | | List of comma delimited IP addresses (or hostnames) of Zabbix servers. No spaces allowed.<br>If ServerActive is not specified, the first entry is used for receiving list of and sending active checks.<br>Note that hostnames must resolve hostname→IP address and IP address→hostname.<br>If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally. |
| ServerActive | no | | | List of comma delimited IP:port (or hostname:port) pairs of Zabbix servers for active checks. No spaces allowed.<br>If ServerActive is specified, first host in the Server option is not used for active checks, only for passive checks.<br>If the port is not specified, ServerPort port is used for that host. If ServerPort is not specified, default port is used.<br>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.<br>**This option is supported in version 1.8.12 and higher.** |
| ServerPort | no | | 10051 | Server port for retrieving list of and sending active checks. |
| SourceIP | no | | | Source IP address for outgoing connections. |

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| StartAgents | no | 1-100 | 3 | Number of pre-forked instances of zabbix_agentd that process passive checks. 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. |
| 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<br><br>Starting from version 1.8.6 Zabbix Agent will not start up in case incorrectly formatted UserParameter entry or duplicate UserParameter key is present in configuration file. |

**Note:**
Starting from version 1.8.6 Zabbix agent daemon will not start up if invalid (not following *parameter=value* notation) or unknown parameter entry is present in configuration file.

**Note:**
Zabbix supports configuration files only in UTF-8 encoding without BOM.

**4 Zabbix Agent (UNIX, Inetd version)**

The file contains configuration parameters for zabbix_agent. The file must exist and it should have read permissions for user 'zabbix'. Supported parameters:

| Parameter | Mandatory | Default value | Description |
|---|---|---|---|
| Alias | no | | Sets an alias for parameter. It can be useful to substitute long and complex parameter name with a smaller and simpler one.<br><br>Starting from version 1.8.6 Zabbix Agent will terminate in case incorrectly formatted Alias entry or duplicate Alias key is present in configuration file. |

| Parameter | Mandatory | Default value | Description |
|---|---|---|---|
| Include | no | | You may include individual files or all files in a directory in the configuration file. 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.<br>Format: UserParameter=<key>,<shell command><br>Note that shell command must not return empty string or EOL only.<br>Example: UserParameter=system.test,who\|wc -l<br><br>Starting from version 1.8.6 Zabbix Agent will terminate in case incorrectly formatted UserParameter entry or duplicate UserParameter key is present in configuration file. |

**Note:**
Starting from version 1.8.6 Zabbix Agent will terminate if invalid (not following *parameter=value* notation) or unknown parameter entry is present in configuration file.

**Note:**
Zabbix supports configuration files only in UTF-8 encoding without BOM.

**5 Zabbix Agent (Windows)**

Installation

Installation is very simple and includes 3 steps:

Step 1

Create configuration file.

Create configuration file c:/zabbix_agentd.conf in UTF8 encoding without BOM (it has similar syntax as the UNIX agent).

An example configuration file is available in Zabbix source archive as *misc/confzabbix_agentd.win.conf*.

Step 2

Install agent as a Windows service.

```
zabbix_agentd.exe --install
```

If you wish to use 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
```

Full path to configuration file should be specified.

Step 3

Run agent.

Now you can use Control Panel to start agent's service or run:

```
zabbix_agentd.exe --start
```

> **Note:**
> Windows NT 4.0 note. Zabbix_agentd.exe uses PDH (Performance Data Helper) API to gather various system information, so PDH.DLL is needed. This DLL is not supplied with Windows NT 4.0, so you need to download and install it by yourself. Microsoft Knowledge Base article number 284996 describes this in detail and contains a download link. You can find this article at http://support.microsoft.com/default.aspx?scid=kb;en-us;284996

Usage

Command line syntax:

```
zabbix_agentd.exe [-Vhp] [-idsx] [-c <file>] [-t <metric>]
```

Zabbix Windows agent accepts the following command line parameters:

Options:

```
-c --config <file>   Specify alternate configuration file (default is c:\zabbix_agentd.conf).
-h --help            Display help information.
-V --version         Display version number.
-p --print           Print known items and exit.
-t --test <item key> Test single item and exit.
```

Functions:

```
-i --install   Install Zabbix agent as a service.
-d --uninstall Uninstall Zabbix agent service.
-s --start     Start Zabbix agent service.
-x --stop      Stop Zabbix agent service.
-m --multiple-agents  Service name will include hostname
```

**Configuration file** The configuration file (c:/zabbix_agentd.conf) contains configuration parameters for zabbix_agentd.exe. Supported parameters:

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| Alias | no | | | Sets an alias for parameter. It can be useful to substitute long and complex parameter name with a smaller and simpler one.<br>For example, if you wish to retrieve paging file usage in percents from the server, you may use parameter ”perf_counter[\Paging File(_Total)\% Usage]”, or you may define an alias by adding the following line to configuration file<br>Alias = pg_usage:perf_counter[\Paging File(_Total)\% Usage]<br>After that you can use parameter name ”pg_usage” to retrieve the same information.<br>You can specify as many ”Alias” records as you wish.<br>Aliases cannot be used for parameters defined in ”PerfCounter” configuration file records.<br><br>Starting from version 1.8.6 Zabbix Agent will not start up in case incorrectly formatted Alias entry or duplicate Alias key is present in configuration file. |
| 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<br>all collected data to Zabbix server or Proxy if the buffer is full. |
| DebugLevel | no | 0-4 | 3 | Specifies debug level<br>0 - no debug<br>1 - critical information<br>2 - error information<br>3 - warnings<br>4 - for debugging (produces lots of information) |
| DisableActive | no | | 0 | Disable active checks. The agent will work in passive mode listening for server. |
| DisablePassive | no | | 0 | Disable passive checks. The agent will not listen on any TCP port.<br>Only active checks will be processed.<br>0 - do not disable<br>1 - disable |

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| EnableRemoteCommands | no | | 0 | Whether remote commands from Zabbix server are allowed.<br>0 - not allowed<br>1 - allowed |
| Hostname | no | | Set by HostnameItem | Unique, case sensitive hostname.<br>Required for active checks and must match hostname as configured on the server.<br>Allowed characters: alphanumeric, '.', ' ', '_' and '-'.<br>Maximum length: 64 |
| HostnameItem | no | | system.hostname | Item used for setting Hostname if it is undefined.<br>Does not support UserParameters, performance counters or aliases, but does support system.run[] regardless of EnableRemoteCommands value.<br><br>Ignored if Hostname is set.<br><br>**This option is supported in version 1.8.6 and higher.** |
| Include | no | | | You may include individual file in the configuration file. |
| ListenIP | no | | 0.0.0.0 | List of comma-delimited IP addresses that the agent should listen on.<br>**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. |
| LogFileSize | no | 0-1024 | 1 | Maximum size of log file in MB.<br>0 - disable automatic log rotation. |
| LogFile | no | | | Name of log file.<br>If not set, Windows Event Log is used. |
| LogRemoteCommands | no | | 0 | Enable logging of executed shell commands as warnings.<br>0 - disabled<br>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.<br>The provided value will be overridden by the parameter 'maxlines',<br>provided in 'log', 'logrt' or 'eventlog' item keys. |

| Parameter | Mandatory | Range | Default | Description |
|---|---|---|---|---|
| PerfCounter | no | | | Syntax: <parameter_name>,”<perf_counter_path>”,<period><br>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).<br>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:<br>PerfCounter = interrupts,”\Processor(0)\Interrupts/sec”,60<br>Please note double quotes around performance counter path.<br>The parameter name (interrupts) is to be used as the item key when creating an item.<br>Samples for calculating average value will be taken every second.<br>You may run ”typeperf -qx” to get list of all performance counters available in Windows.<br><br>Starting from version 1.8.6 Zabbix Agent will not start up in case incorrectly formatted PerfCounter entry or duplicate PerfCounter key is present in configuration file. |
| RefreshActiveChecks | no | 60-3600 | 120 | How often list of active checks is refreshed, in seconds. |
| Server | yes | | | List of comma delimited IP addresses (or hostnames) of Zabbix servers. No spaces allowed.<br>If ServerActive is not specified, the first entry is used for receiving list of and sending active checks.<br>Note that hostnames must resolve hostname→IP address and IP address→hostname.<br>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 | | | List of comma delimited IP:port (or hostname:port) pairs of Zabbix servers for active checks. No spaces allowed.<br>If ServerActive is specified, first host in the Server option is not used for active checks, only for passive checks.<br>If the port is not specified, ServerPort port is used for that host. If ServerPort is not specified, default port is used.<br>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.<br>**This option is supported in version 1.8.13 and higher.** |
| ServerPort | no | | 10051 | Server port for retrieving list of and sending active checks. |
| SourceIP | no | | | Source IP address for outgoing connections. |
| StartAgents | no | 1-63 (*) | 3 | Number of pre-forked instances of zabbix_agentd that process passive checks. 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.<br>0 - do not allow<br>1 - allow |
| UserParameter | | | | User-defined parameter to monitor. There can be several user-defined parameters.<br>Format: UserParameter=<key>,<shell command><br>Note that shell command must not return empty string or EOL only.<br>Example: UserParameter=system.test,echo 1<br><br>Starting from version 1.8.6 Zabbix Agent will not start up in case incorrectly formatted UserParameter entry or duplicate UserParameter key is present in configuration file. |

**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:**
Starting from version 1.8.6 Zabbix agent will not start up if invalid (not following *parameter=value* notation) or unknown parameter entry is present in configuration file.

**Note:**
Zabbix supports configuration files only in UTF-8 encoding without BOM.

#### 6 Zabbix Sender (UNIX)

Zabbix UNIX Sender is a command line utility which 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. Zabbix Sender can be started by executing:

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

Starting with Zabbix 1.8.2, objects with whitespaces can be passed using zabbix_sender. In this case, these objects must be quoted using double quotes.

Starting with Zabbix 1.8.4, zabbix_sender has been improved in realtime sending scenarios by gathering multiple values that are passed to it in close succession, and sending them to the server in single connection. Value that is not further apart from 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:**
Starting from version 1.8.6 Zabbix Sender will terminate if invalid (not following *parameter=value* notation) parameter entry is present in specified configuration file.

See Zabbix Sender manpage for more information.

#### 7 Zabbix Get (UNIX)

Zabbix get is a process which communicates with Zabbix agent and retrieves required information.

The utility is usually used for troubleshooting of Zabbix agents.

Zabbix get can be started by executing:

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

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 for.
  -h --help                        Give this help.
  -V --version                     Display version number.
```

In order to get this help run:

```
shell> zabbix_get --help
```

#### 8 Special notes on "Include" configuration 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 '
```

## 4 Configuration

actions macros applications graphs medias host_templates host_groups host_and_trigger_dependencies items user_parameters triggers screens_and_slide_shows it_services user_permissions the_queue utilities regexps defines suffixes time_period

#### 1 Actions

Zabbix reacts to events by executing set of operations. An action can be defined for any event or set of events generated by Zabbix.

Action attributes:

| Parameter | Description |
|---|---|
| Name | Unique action name. |
| Event source | Source of event.<br>Currently three sources are supported:<br>**Triggers** - events generated by trigger status changes<br>**Discovery** - events generated by network discovery module<br>**Auto registration** - events generated by new active agents |
| Enable escalations | Enable escalations. If enabled, the action will be escalated according to operation steps defined for operations. |
| Period (seconds) | Time period for increase of escalation step. |
| Default subject | Default notification subject. The subject may contain macros. |
| Default message | Default notification message. The message may contain macros. |
| Recovery message | If enabled, Zabbix will send a recovery message after the original problem is resolved. The messages will be sent only to those who received any message regarding this problem before. |
| Recovery subject | Subject of the recovery message. It may contain macros. |
| Recovery message | Recovery message. It may contain macros. |
| Status | Action status:<br>**Enabled** - action is active<br>**Disabled** - action is disabled |

**Warning:**
Warning: before enabling recovery messages or escalations, make sure to add "Trigger value = PROBLEM" condition to the action, otherwise remedy events can become escalated as well.

Action conditions

An action is executed only in case if an event matches defined set of conditions.

The following conditions can be defined for trigger based events:

| Condition type | Supported operators | Description |
|---|---|---|
| Application | =<br>like<br>not like | **=** - event came from a trigger, which refers to an item that is linked to the specified application<br>**like** - event came from a trigger, which refers to an item that is linked to an application, containing the string<br>**not like** - event came from trigger, which refers to an item that is linked to an application not containing the string |
| Host group | =<br><> | Compare against host group having a trigger which generated event.<br>**=** - event came from this host group<br>**<>** - event did not come from this host group |
| Host template | =<br><> | Compare against Host Template the trigger belongs to.<br>**=** - event came from a trigger inherited from this Host Template<br>**<>** - event did not come from a trigger inherited from this Host Template |
| Host | =<br><> | Compare against Host having a trigger which generated event.<br>**=** - event came from this Host<br>**<>** - event did not come from this Host |
| Trigger | =<br><> | Compare against Trigger which generated event.<br>**=** - event generated by this Trigger<br>**<>** - event generated by other Trigger |
| Trigger description (name) | like<br>not like | Compare against Trigger Name which generated event.<br>**like** - String can be found in Trigger Name. Case sensitive.<br>**not like** - String cannot be found in Trigger Name. Case sensitive.<br>*Note*: Entered value will be compared to trigger description (name) with all macros expanded. |

| Condition type | Supported operators | Description |
|---|---|---|
| Trigger severity | =<br><><br>>=<br><= | Compare with Trigger Severity. **=** - equal to trigger severity<br>**<>** - not equal to trigger severity<br>**>=** - more or equal to trigger severity<br>**<=** - less or equal to trigger severity |
| Trigger value | = | Compare with Trigger Value.<br>**=** - equal to trigger value (OK or PROBLEM) |
| Time period in | in | Event is within time period.<br>**in** - event time matches the time period.<br>See Time period specification page for description of the format. |
| Maintenance status | =<br><> | Check if host is in maintenance.<br>**=** - Host is in maintenance mode.<br>**<>** - Host is not in maintenance mode. |

Trigger value:

Trigger changes status from OK to PROBLEM (trigger value is PROBLEM) Trigger changes status from PROBLEM to OK (trigger value is OK)

Status change OK→UNKNOWN→PROBLEM is treated as OK→PROBLEM, and PROBLEM→UNKNOWN→OK as PROBLEM→OK.

The following conditions can be defined for Discovery based events:

| Condition type | Supported operators | Description |
|---|---|---|
| Host IP | =<br><> | Check if IP address of a discovered Host is or is not in the range of IP addresses.<br>**=** - Host IP is in the range<br>**<>** - Host IP is out of the range |
| Service type | =<br><> | Check if a discovered service.<br>**=** - matches discovered service<br>**<>** - event came from a different service |
| Service port | =<br><> | Check if TCP port number of a discovered service is or is not in the range of ports.<br>**=** - service port is in the range<br>**<>** - service port is out of the range |
| Discovery status | = | **Up** - matches Host Up and Service Up events<br>**Down** - matches Host Down and Service Down events |

| Condition type | Supported operators | Description |
|---|---|---|
| Uptime/Downtime | >=<br><= | Downtime for Host Down and Service Down events. Uptime for Host Up and Service Up events.<br>**>=** - uptime/downtime is more or equal<br>**<=** - uptime/downtime is less or equal. Parameter is given in seconds. |
| Received value | =<br><><br>>=<br><=<br>like<br>not like | Compare with value received from an agent (Zabbix, SNMP). String comparison.<br>**=** - equal to the value<br>**<>** - not equal to the value<br>**>=** - more or equal to the value<br>**<=** - less or equal to the value<br>**like** - has a substring<br>**not like** - does not have a substring. Parameter is given as a string. |

For example this set of conditions (calculation type: AND/OR):

- 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'**)**

Operations

Operation or a set of operations is executed when event matches conditions.

Zabbix supports the following operations:

- Send message
- Remote command(s), including IPMI.

**Note:**
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. Starting from 1.8.2 the subject and the body of the message are base64-encoded to follow 'SMTP/MIME e-mail' format standard.

**Note:**
Starting with 1.8.3, if the subject contains ASCII characters only, it is not UTF-8 encoded.

Additional operations available for discovery events:

- Add host
- Remove host
- Enable host
- Disable host
- Add to group
- Delete from group
- Link to template
- Unlink from template

When adding a host, its name is decided by standard **gethostbyname** function. If the host can be resolved, resolved name is used. If not, 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 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.

Operation attributes:

| Parameter | Description |
|---|---|
| Step | If escalation is enabled for this action, escalation settings:<br>**From** - execute for each step starting from this one<br>**To** - till this (0, for all steps starting from From)<br>**Period** - increase step number after this period, 0 - use default period. |
| Operation type | Type of action:<br>**Send message** - send message to user<br>**Execute command** - execute remote command |
| Event Source | |
| Send message to | Send message to:<br>**Single user** - a single user<br>**User group** - to all members of a group |
| Default message | If selected, default message will be used. |
| Subject | Subject of the message. The subject may contain macros. |
| Message | The message itself. The message may contain macros. |
| Remote command | List of remote commands. |

**Attention:**
Starting from 1.6.2, Zabbix sends notifications only to those users, which have read permissions to a host (trigger), which generated the event. At least one host of a trigger expression must be accessible.

**Note:**
As with some triggers event generation can be defined for every PROBLEM evaluation of the trigger, it is worthy of note that if escalations are defined for actions on these events, 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.

Macros for messages and remote commands

The macros can be used for more efficient reporting.

Example 1

Subject:

```
{TRIGGER.NAME}: {TRIGGER.STATUS}
```

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(0)}
```

The message will be replaced by something like:

```
Processor load is: 1.45
```

Example 3

Message:

```
Latest value: {{HOSTNAME}:{TRIGGER.KEY}.last(0)}
MAX for 15 minutes: {{HOSTNAME}:{TRIGGER.KEY}.max(900)}
MIN for 15 minutes: {{HOSTNAME}:{TRIGGER.KEY}.min(900)}
```

The message will be replaced by something like:

```
Latest value: 1.45
MAX for 15 minutes: 2.33
MIN for 15 minutes: 1.01
```

**2 Macros**

Zabbix supports number of macros which may be used in various situations. Effective use of macros allows to save time and make Zabbix configuration more transparent.

List of supported macros

The table contains complete list of macros supported by Zabbix. *X* means ”supported”.

| Item descriptions | | | | | | | | | | DESCRIPTION |
|---|---|---|---|---|---|---|---|---|---|---|
| Trigger names | | | | | | | | | ▼▼ | |
| Trigger expressions | | | | | | | | ▼▼ | | |
| Map labels[1] | | | | | | | ▼▼ | | | |
| Item key’s parameters | | | | | | ▼▼ | | | | |
| GUI Scripts | | | | | ▼▼ | | | | | |
| Auto registration notifications | | | | ▼▼ | | | | | | |
| Discovery notifications | | | ▼▼ | | | | | | | |
| Notifications and commands | | ▼▼ | | | | | | | | |
| MACRO | ▼▼ | | | | | | | | | |
| ▼▼ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| {DATE} | X | X | X | | | | | | | Current date in yyyy.mm.dd. format. |

| | |
|---|---|
| {DISCOVERY.DEVICE.IPADDRESS} | IP address of the discovered device. Available always, does not depend on host being added. |
| {DISCOVERY.DEVICE.STATUS} | Status of the discovered device: can be either UP or DOWN. |
| {DISCOVERY.DEVICE.UPTIME} | Time since the last change of discovery status for a particular device. For example: 1h 29m. For devices with status DOWN, this is the period of their downtime. |
| {DISCOVERY.RULE.NAME} | Name of the discovery rule that discovered the presence or absence of the device or service. |

| | |
|---|---|
| {DISCOVERY.SERVICE.NAME} | Name of the service that was discovered. For example: HTTP. |
| {DISCOVERY.SERVICE.PORT} | Port of the service that was discovered. For example: 80. |
| {DISCOVERY.SERVICE.STATUS} | Status of the discovered service: can be either UP or DOWN. |
| {DISCOVERY.SERVICE.UPTIME} | Time since the last change of discovery status for a particular service. For example: 1h 29m. For services with status DOWN, this is the period of their downtime. |

| | | | | |
|---|---|---|---|---|
| {ESC.HISTORY[X]} | | | | Escalation history. Log of previously sent messages. Shows previously sent notifications, on which escalation step they were sent and their status (*sent*, *in progress* or *failed*). |
| {EVENT.ACK.HISTORY} | | | | |
| {EVENT.ACK.STATUS} | | | | |
| {EVENT.AGE} | X | X | X | Age of the event. Useful in escalated messages. |
| {EVENT.DATE} | X | X | X | Date of the event. |
| {EVENT.ID} | X | X | X | Numeric event ID which triggered this action. |
| {EVENT.TIME} | X | X | X | Time of the event. |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| {HOSTNAME<1-9>} | | X | | X | X | | X | | Host name of the Nth item of the trigger which caused a notification. Supported in auto registration notifications since 1.8.4. |
| {HOST.CONN<1-9>} | | | X | X | X | | | | IP and host DNS name depending on host settings. |
| {HOST.DNS<1-9>} | | | X | X | X | | | | Host DNS name. |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| {IPADDRESS<1-9>} | | | X | X | X | | | | IP address of the Nth item of the trigger which caused a notification. |
| {ITEM.ID<1-9>} | | | | | | | | | Numeric ID of the Nth item of the trigger which caused a notification. Supported since 1.8.12. |

| | | | | |
|---|---|---|---|---|
| {ITEM.LASTVALUE<1-9>} | X | | X | The latest value of the Nth item of the trigger expression which caused a notification. Supported from Zabbix 1.4.3. It is alias to {{HOSTNAME}:{TRIGGER.K |
| {ITEM.LOG.AGE<1-9>} | X | | | |
| {ITEM.LOG.DATE<1-9>} | X | | | |
| {ITEM.LOG.EVENTID<1-9>} | X | | | |
| {ITEM.LOG.NSEVERITY<1-9>} | X | | | |
| {ITEM.LOG.SEVERITY<1-9>} | X | | | |
| {ITEM.LOG.SOURCE<1-9>} | X | | | |
| {ITEM.LOG.TIME<1-9>} | X | | | |
| {ITEM.NAME<1-9>} | X | | | Name of the Nth item of the trigger which caused a notification. |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| {ITEM.VALUE<1-9>} | X | | | | | | | X | | The latest value of Nth item of the trigger expression if used for displaying triggers. Historical (when event happened) value of Nth item of the trigger expression if used for displaying events and notifications. Supported from Zabbix 1.4.3. |
| {NODE.ID<1-9>} | X | X | X | | | | | | | |
| {NODE.NAME<1-9>} | X | X | X | | | | | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| {PROFILE.CONTACT<1-9>} | X | | | | | | | | | Contact from host profile. |
| {PROFILE.DEVICETYPE<1-9>} | X | | | | | | | | | Device type from of host profile. |
| {PROFILE.HARDWARE<1-9>} | X | | | | | | | | | Hardware from host profile. |
| {PROFILE.LOCATION<1-9>} | X | | | | | | | | | Location from host profile. |
| {PROFILE.MACADDRESS<1-9>} | X | | | | | | | | | Mac Address from host profile. |

| | | | | |
|---|---|---|---|---|
| {PROFILE.NAME<1-9>} | X | | | Name from host profile. |
| {PROFILE.NOTES<1-9>} | X | | | Notes from host profile. |
| {PROFILE.OS<1-9>} | X | | | OS from host profile. |
| {PROFILE.SERIALNO<1-9>} | X | | | Serial No from host profile. |
| {PROFILE.SOFTWARE<1-9>} | X | | | Software from host profile. |
| {PROFILE.TAG<1-9>} | X | | | Tag from host profile. |
| {PROXY.NAME<1-9>} | X | X | X | Proxy name of the Nth item of the trigger which caused a notification. Supported since 1.8.4. |
| {TIME} | X | X | X | Current time in hh:mm:ss. |
| {TRIGGER.COMMENT} | X | | | Trigger comment. |

| | | |
|---|---|---|
| {TRIGGER.EVENTS.UNACK} | X | Number of unacknowledged events for a map element in maps, or for the trigger which generated current event in notifications. Supported in map element labels since 1.8.3. |
| {TRIGGER.EVENTS.PROBLEM.UNACK} | X | Number of unacknowledged PROBLEM events for all triggers disregarding their state. Supported since 1.8.3. |
| {TRIGGER.PROBLEM.EVENTS.PROBLEM.UNACK} | X | Number of unacknowledged PROBLEM events for triggers in PROBLEM state. Supported since 1.8.3. |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| {TRIGGER.EVENTS.ACK} | | | | | X | | | | Number of acknowledged events for a map element in maps, or for the trigger which generated current event in notifications. Supported since 1.8.3. |
| {TRIGGER.EVENTS.PROBLEM.ACK} | | | | | X | | | | Number of acknowledged PROBLEM events for all triggers disregarding their state. Supported since 1.8.3. |
| {TRIGGER.PROBLEM.EVENTS.PROBLEM.ACK} | | | | | X | | | | Number of acknowledged PROBLEM events for triggers in PROBLEM state. Supported since 1.8.3. |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |

| | |
|---|---|
| {TRIGGER.EXPRESSION} | Trigger expression. Supported since 1.8.12. |
| {TRIGGER.ID} | Numeric trigger ID which triggered this action. |
| {TRIGGER.KEY<1-9>} | Key of the Nth item of the trigger which caused a notification. |
| {TRIGGER.NAME} | Name (description) of the trigger. |
| {TRIGGER.NSEVERITY} | Numerical trigger severity. Possible values: 0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster. Supported starting from Zabbix 1.6.2. |

| | | | |
|---|---|---|---|
| {TRIGGER.SEVERITY} | X | | Trigger severity. Possible values: Not classified, Information, Warning, Average, High, Disaster, Unknown |
| {TRIGGER.STATUS} | X | | Trigger state. Can be either PROBLEM or OK. **{STATUS} is deprecated.** |
| {TRIGGER.URL} | X | | Trigger URL. |
| {TRIGGER.VALUE} | X | X | Current trigger value: 0 - trigger is in OK state, 1 – trigger is in PROBLEM state, 2 – trigger UNKNOWN. This macro can also be used in trigger expressions. |

| | | |
|---|---|---|
| {TRIGGERS.UNACK} | X | Number of unacknowledged triggers for a map element, disregarding trigger state. Trigger is considered to be unacknowledged if at least one of its PROBLEM events is unacknowledged. |
| {TRIGGERS.PROBLEM.UNACK} | X | Number of unacknowledged PROBLEM triggers for a map element. Trigger is considered to be unacknowledged if at least one of its PROBLEM events is unacknowledged. Supported since 1.8.3. |

| | | |
|---|---|---|
| {TRIGGERS.ACK} | X | Number of acknowledged triggers for a map element, disregarding trigger state. Trigger is considered to be acknowledged if all of it's PROBLEM events are acknowledged. Supported since 1.8.3. |
| {TRIGGERS.PROBLEM.ACK} | X | Number of acknowledged PROBLEM triggers for a map element. Trigger is considered to be acknowledged if all of it's PROBLEM events are acknowledged. Supported since 1.8.3. |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| {host:key.func(param)} | | | | | X[2] | X | | | Simple macros as used in trigger expressions. |
| {$MACRO} | | | | X | | X | X | X | User macros. Supported in trigger names and item descriptions since 1.8.4. |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |

**Note:**
[1] Macros for map labels are supported since 1.8.
[2] Only functions **last**, **avg**, **max** and **min** with seconds as arguments are supported in map labels.

Macro {TRIGGER.ID} is supported in trigger URL since Zabbix 1.8.8.

User macros

For a 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 descriptions (since Zabbix 1.8.4)
- item key parameters
- trigger names (since Zabbix 1.8.4)
- trigger expression parameters and constants (see examples)

The following characters are allowed in the macro names: **A-Z** , **0-9** , **_** , **.**

Zabbix substitutes macros according to the following precedence:

1. host 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 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 macros, open host or template properties and look for the *Macros* block on the right hand side

**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

Example 1

Use of host macro in item ”Status of SSH daemon” key:

**ssh,{$SSH_PORT}**

Example 2

Use of host macro in trigger ”CPU load is too high”:

**{ca_001:system.cpu.load[,avg1].last(0)}>{$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 like this: SOME_PERIOD => #3

Example 3

Use of two macros in trigger ”CPU load is too high”:

**{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.

**Note:**
User macros are supported in SNMP OID field since Zabbix 1.8.4.

### 3 Applications

Application is a set of host items. For example, application ’MySQL Server’ may contain all items which are related to the MySQL server: availability of MySQL, disk space, processor load, transactions per second, number of slow queries, etc.

An item may be linked with one or more applications.

Applications are used in Zabbix front-end to group items.

**Attention:**
Currently a host cannot be linked to different templates having same application.

### 4 Graphs

Custom (user defined) graphs allow the creation of complex graphs.

These graphs, once configured, can be easily accessed via Monitoring→Graphs.

Configuration of custom graphs can be accessed by navigating to Configuration→Templates or Configuration→Hosts and clicking on *Graphs* link for corresponding template or host.

**Note:**
When creating a new graph, first item can be added from any template or host. Then, depending on the choice, further items can be added :
1. if the first item was from a template, only from that template;
2. if the first item was from any host, from any host (but not from templates anymore)

## 5 Media

A medium is a delivery channel for Zabbix alerts. None, one or more media types can be assigned to user.

Email

Email notification.

Jabber

Notifications using Jabber messaging.

When sending notifications, Zabbix tries to look up a 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.

Looking up Jabber SRV records is supported since Zabbix 1.8.6. Prior to that Zabbix only tried an address record.

Script

Custom media scripts are executed from the path defined in the Zabbix server configuration file variable **AlertScriptsPath**. The script has three command line variables passed to it:

- Recipient
- Subject
- Message

Environment variables are not preserved or created for the script, so they should be handled explicitly.

GSM Modem

Zabbix supports sending of SMS messages using Serial GSM Modem connected to Zabbix Server's serial port.

Make sure that:

- Speed of a serial device (normally /dev/ttyS0 under Linux) matches GSM Modem. Zabbix does not set speed of the serial link. It uses default settings.
- The serial device has read/write access for user zabbix. Run commans ls –l /dev/ttyS0 to see current permission of the serial device.
- 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 present) in a terminal software, such as Unix minicom or Windows HyperTerminal.

Zabbix has been tested with the following GSM modems:

- Siemens MC35
- Teltonika ModemCOM/G10

## 6 Host templates

Use of templates is an excellent way of making maintenance of Zabbix much easier.

A template can be linked to a number of hosts. Items, triggers and graphs of the template will be automatically added to the linked hosts. Change definition of a template item (trigger, graph) and the change will be automatically applied to the hosts.

Host template attributes:

| Parameter | Description |
|---|---|
| **Name** | Unique template (host) name. The name must be unique within ZABBIX Node. |
| **Groups** | List of host groups the template belongs to. |
| **New group** | Assign new host group to the template. |
| **Link with template** | Used to create hierarchical templates. |

### 7 Host groups

Host group may have zero, one or more hosts.

Host group attributes:

| Parameter | Description |
|---|---|
| **Group name** | Unique host group name. The name must be unique within a Zabbix node. |
| **Hosts** | List of hosts of this group. |

### 8 Host and trigger dependencies

Zabbix does not support host dependencies. Host dependencies can be defined using more flexible option, i.e. trigger dependencies.

**How it works?**

A trigger may have list of one or more triggers it depends on. It means that the trigger will still change its status regardless of state of the triggers in the list, yet the trigger won't generate notifications and actions in case if one of the trigger in the list has state PROBLEM.

Example 1

Host dependency

Suppose you have two hosts: a router and a server. The server is behind the router. So, we want to receive only one notification if the route is down:

*"The router is down"*

instead of:

*"The router is down" and "The host is down"*

In order to achieve this, we create a trigger dependency:

`"The host is down" depends on "The router is down"`

In case if both the server and the router is down, Zabbix will not execute actions for trigger "The host is down".

### 10 User Parameters

Functionality of Zabbix agents can be enhanced by defining user parameters (**UserParameter** configuration parameter) in agent's configuration file. Once user parameters are defined, they can be accessed in the same way as any other agent items by using the key, specified in the parameter definition.

User parameters are commands executed by Zabbix agent. **/bin/sh** is used as a command line interpreter under UNIX operating systems.

See a step-by-step tutorial on making use of user parameters.

1 Simple user parameters

In order to define a new parameter for monitoring, one line has to be added to configuration file of Zabbix agent and the agent must be restarted.

User parameter has the following syntax:

`UserParameter=key,command`

| Parameter | Description |
|---|---|
| **Key** | Unique item key. |

| Parameter | Description |
|---|---|
| **Command** | Command to be executed to evaluate value of the Key. |

Example 1

Simple command

```
UserParameter=ping,echo 1
```

The agent will always return '1' for item with key 'ping'.

Example 2

More complex example

```
UserParameter=mysql.ping,mysqladmin -uroot ping | grep -c alive
```

The agent will return '1', if MySQL server is alive, '0' - otherwise.

2 Flexible user parameters

Flexible user parameters can be used for more control and flexibility.

For flexible user parameters,

```
UserParameter=key[*],command
```

| Parameter | Description |
|---|---|
| **Key** | Unique item key. The [*] defines that this key accepts parameters. |
| **Command** | Command to be executed to evaluate value of the Key.<br>Zabbix parses content of [] and substitutes $1,...,$9 in the command.<br>$0 will be substituted by the original command (prior to expansion of $0,...,$9) to be run. |

**Note:**
To use positional references unaltered, specify double dollar sign - for example, awk '{print $$2}'.

**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.

**Warning:**
Command used should always return a value that is not empty (and not a newline). If non-valid value is returned, ZBX_NOTSUPPORTED will be sent back by the agent.

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]
```

**Attention:**
Note that Zabbix agent daemon does not support user parameters with **-t** or **-p** agent switches (used to test single item or print out a list of all supported items) until version 1.8.3. See manpage in earlier versions for more information.

#### 11 Windows performance counters

Windows performance counter can be effectively monitored using perf_counter[].

For example:

```
perf_counter["\Processor(0)\Interrupts/sec"]
```

or

```
perf_counter["\Processor(0)\Interrupts/sec", 10]
```

In order to get full list of performance counter available for monitoring you may run:

```
typeperf -qx
```

Unfortunately, depending on local settings naming of the performance counters can be different on different Windows servers. This introduces certain problem when creating a template for monitoring number of Windows machines having different locales.

Every performance counter can be translated into numeric form, which is unique and exactly the same regardless of language settings.

Run **regedit**, then find *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\009*. The registry entry contains information like:

```
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
....
```

So, in order to translate string name of a performance counter into numeric form, find corresponding numbers for each part of the performance counter, like:

```
System → 2
% Processor Time → 6
\System\% Processor Time
```

Then use these numbers to create a numeric format:

\2\6

1 Performance counter parameters

In order to define a new parameter for monitoring performance counters, one line can be added to configuration file of Zabbix agent and the agent must be restarted. For example:

```
PerfCounter=UserPerfCounter1,"\Memory\Page Reads/sec",30
or
PerfCounter=UserPerfCounter2,"\4\24",30
```

Then it is possible to use ”UserPerfCounter1” and ”UserPerfCounter2” as usual item checks in the frontend or elsewhere, simalar to UserParameter.

## 12 Triggers

Trigger is defined as a logical expression and represents system state.

A trigger may have the following values:

| VALUE | DESCRIPTION |
|---|---|
| PROBLEM | Normally means that something happened. For example, processor load is too high. Called TRUE in older Zabbix versions. |
| OK | This is a normal trigger state. Called FALSE in older Zabbix versions. |
| UNKNOWN | In this case, Zabbix cannot evaluate trigger expression. This may happen because of several reasons:<br>server is unreachable<br>trigger expression cannot be evaluated<br>trigger expression has been recently changed |

Triggers are evaluated based on history data only; trend data are never considered.

1 Expression for triggers

The expressions used in triggers are very flexible. You can use them to create complex logical tests regarding monitored statistics.

1.1 Expression operators

The following operators are supported for triggers **(descending priority of execution)**:

| PRIORITY | OPERATOR | DEFINITION |
|---|---|---|
| **1** | **/** | Division |
| **2** | *** \|**Multiplication** \| \|3** | **-** Arithmetical minus |
| **4** | **+** | Arithmetical plus |
| **5** | **<** | Less than. The operator is defined as: A<B ⇔ (A<=B-0.000001) |
| **6** | **>** | More than. The operator is defined as: A>B ⇔ (A>=B+0.000001) |
| **7** | **#** | Not equal. The operator is defined as: A#B ⇔ (A<=B-0.000001) \| (A>=B+0.000001) |
| **8** | **=** | Is equal. The operator is defined as: A=B ⇔ (A>B-0.000001) & (A<B+0.000001) |
| **9** | **&** | Logical AND |
| **10** | **\|** | Logical OR |

2 Trigger functions

Trigger functions allow to reference collected values, current time and other factors.

2.1 Time based functions

Trigger status (expression) is recalculated every time Zabbix server receives new value, if this value is part of this expression. If time based functions are used in the expression, it 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.

Time based functions are:

- nodata()
- date()
- dayofmonth()
- dayofweek()
- time()
- now()

2.2 List of trigger functions

The following functions are supported:

**Attention:**
1) All functions return numeric values only. Comparison to strings is not supported, for example.

2) String arguments should be double quoted. Otherwise, they might get misinterpreted.

| ▼ | FUNCTION | *Parameter(s)* | *Supported value types* |
|---|---|---|---|
| | **Definition** | | |
| abschange | Returns absolute difference between last and previous values.<br>For strings:<br>0 - values are equal<br>1 - values differ | *ignored* | *float, int, str, text, log* |
| avg | Average value for period of time.<br>Parameter defines length of the period in seconds.<br>The function accepts a second, optional parameter **time_shift**. It is useful when there is a need to compare the current average value with the average value time_shift seconds back. For instance, avg(3600,86400) will return the average value for an hour one day ago.<br>Parameter time_shift is supported from Zabbix 1.8.2. | *sec or #num* | *float, int* |
| change | | *ignored* | *float, int, str, text, log* |

| ▼ | FUNCTION | *Parameter(s)* | *Supported value types* |
|---|---|---|---|
| | Returns difference between last and previous values.<br>For strings:<br>0 - values are equal<br>1 - values differ | | |
| count | | *sec or #num* | *float, int, str, text, log* |

| ▼ | FUNCTION | *Parameter(s)* | *Supported value types* |
|---|---|---|---|
| | Number of historical values for period of time in seconds or number of last **#num** values matching condition.<br>The function accepts second optional parameter **pattern**, third parameter **operator**, and fourth parameter **time_shift**.<br>For example, *count(600,12)* will return exact number of values equal to ’12’ stored in the history.<br>Integer items: exact match<br>Float items: match within 0.000001<br>String, text and log items: operators like (default), eq, ne are supported<br>Supported operators:<br>**eq** - equal<br>**ne** - not equal<br>**gt** - greater<br>**ge** - greater or equal<br>**lt** - less<br>**le** - less or equal<br>**like** (textual search only) - matches if contains pattern.<br>For example, *count(600,12,”gt”)* will return exact number of values which are more than ’12’ stored in the history for the last 600 seconds.<br>Another example: *count(#10,12,”gt”,86400)* will return exact number of values which are larger than ’12’ stored in the history among last 10 values 24 hours ago.<br>If there is a need to count arbitrary values, for instance, for the last 600 seconds | | |

| ▼ | FUNCTION | *Parameter(s)* | *Supported value types* |
|---|---|---|---|
| date | Returns current date in YYYYMMDD format.<br>For example: 20031025 | *ignored* | *any* |
| dayofmonth | Returns day of month in range of 1 to 31.<br>This function is supported since Zabbix 1.8.5. | *ignored* | *any* |
| dayofweek | Returns day of week in range of 1 to 7. Mon - 1, Sun - 7. | *ignored* | *any* |
| delta | Same as max()-min().<br>Since Zabbix 1.8.2, the function supports a second, optional parameter time_shift. See function avg for an example of its use. | *sec or #num* | *float, int* |
| diff | Returns:<br>1 - last and previous values differ<br>0 - otherwise | *ignored* | *float, int, str, text, log* |
| fuzzytime | Returns 1 if timestamp (item value) does not differ from Zabbix server time for more than N seconds, 0 - otherwise.<br>Usually used with system.localtime to check that local time is in sync with local time of Zabbix server. | *sec* | *float, int* |
| iregexp | This function is non case-sensitive analogue of **regexp**. | *1st - string, 2nd - sec or #num* | *str, log, text* |
| last | | *sec or #num* | *float, int, str, text, log* |

| ▼ | FUNCTION | *Parameter(s)* | *Supported value types* |
|---|---|---|---|
| | Last (most recent) value. Parameter: sec - ignored #num - Nth value For example, last(0) is always equal to last(#1) last(#3) - third most recent value The function also supports a second optional **time_shift** parameter. For example, last(0,86400) will return the most recent value one day ago. Zabbix does not guarantee exact order of values if more than two values exist within one second in history. Parameter #num is supported starting from Zabbix 1.6.2. Parameter time_shift is supported starting from Zabbix 1.8.2. See function avg for an example of its use. | | |
| logeventid | Check if Event ID of the last log entry matches a regular expression. Parameter defines the regular expression, POSIX extended style. Returns: 0 - does not match 1 - matches This function is supported since Zabbix 1.8.5. | *string* | *log* |
| logseverity | | *ignored* | *log* |

| ▼ | FUNCTION | *Parameter(s)* | *Supported value types* |
|---|---|---|---|
| | Returns log severity of the last log entry.<br>Parameter is ignored.<br>0 - default severity<br>N - severity (integer, useful for Windows event logs). Zabbix takes log severity from field **Information** of Windows event log. | | |
| logsource | Check if log source of the last log entry matches parameter.<br>0 - does not match<br>1 - matches<br>Normally used for Windows event logs. For example, logsource(”VMware Server”). | *string* | *log* |
| max | Maximal value for period of time. Parameter defines length of the period in seconds. Since Zabbix 1.8.2, the function supports a second, optional parameter time_shift. See function avg for an example of its use. | *sec or #num* | *float, int* |
| min | Minimal value for period of time. Parameter defines length of the period in seconds. Since Zabbix 1.8.2, the function supports a second, optional parameter time_shift. See function avg for an example of its use. | *sec or #num* | *float, int* |
| nodata | Returns:<br>1 - if no data received during period of time in seconds. The period should not be less than 30 seconds.<br>0 - otherwise | *sec* | *any* |

| ▼ | FUNCTION | *Parameter(s)* | *Supported value types* |
|---|---|---|---|
| now | Returns number of seconds since the Epoch (00:00:00 UTC, January 1, 1970). | *ignored* | *any* |
| prev | Returns previous value. Parameter is ignored.<br>Same as last(#2) | *ignored* | *float, int, str, text, log* |
| regexp | Check if last value matches regular expression.<br>Parameter defines regular expression, POSIX extended style.<br>Second optional parameter is number of seconds or number of lines to analyse. In this case more than one value will be processed.<br>This function is case-sensitive.<br>Returns:<br>1 - found<br>0 - otherwise | *1st - string, 2nd - sec or #num* | *str, log, text* |
| str | Find string in last (most recent) value. Parameter defines string to find. Case sensitive!<br>Second optional parameter is number of seconds or number of lines to analyse. In this case more than one value will be processed.<br>Returns:<br>1 - found<br>0 - otherwise | *1st - string, 2nd - sec or #num* | *str, log, text* |
| strlen | | *sec or #num* | *str, log, text* |

| ▼ | FUNCTION | *Parameter(s)* | *Supported value types* |
|---|---|---|---|
| | Length of the last (most recent) value in characters (not bytes).<br>Parameters are the same as for function last.<br>For example, strlen(0) is equal to strlen(#1)<br>strlen(#3) - length of the third most recent value<br>strlen(0,86400) - length of the most recent value one day ago.<br>This function is supported since Zabbix 1.8.4. | | |
| sum | Sum of values for period of time.<br>Parameter defines length of the period in seconds.<br>Since Zabbix 1.8.2, the function supports a second, optional parameter time_shift. See function avg for an example of its use. | *sec or #num* | *float, int* |
| time | Returns current time in HHMMSS format. Example: 123055 | *ignored* | *any* |

**Note:**
Some of the functions cannot be used for non-numeric parameters!

2.3 Trigger function parameters

Most of numeric functions accept number of seconds as an argument. You may also use prefix # to specify that argument has a different meaning:

| FUNCTION CALL | MEANING |
|---|---|
| **sum(600)** | Sum of all values within 600 seconds |
| **sum(#5)** | Sum of the last 5 values |

Function **last** uses a different meaning for values, prefixed with the hash mark - it makes it choose n-th previous value, so given values (from most recent to least recent) 3, 7, 2, 6, 5, **last(#2)** would return *7* and **last(#5)** would return *5*.

Trigger expressions support using various multipliers as suffixes.

A simple useful expression might look like:

```
{<server>:<key>.<function>(<parameter>)}<operator><constant>
```

A parameter must be given even for those functions which ignore it. Example: last(0)

Example 1

Processor load is too high on www.zabbix.com

```
{www.zabbix.com:system.cpu.load[all,avg1].last(0)}>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(0)}>5|{www.zabbix.com:system.cpu.load[all,avg1].min(600)}>2
```

The expression is true when either the current processor load is more than 5 or the processor load was more than 2 during last 10 minutes.

Example 3

/etc/passwd has been changed

Use of function diff:

```
{www.zabbix.com:vfs.file.cksum[/etc/passwd].diff(0)}=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(300)}>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)}=0&{smtp2.zabbix.com:net.tcp.service[smtp].last(0)}=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(1800,0)}>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(180)}=1
```

'tick' must have type 'Zabbix trapper'. In order to make this trigger work, item 'tick' must be defined. The host should periodically send data for this parameter using zabbix_sender. If no data is received within 180 seconds, the trigger value becomes PROBLEM.

Example 9

CPU activity at night time

Use of function time():

```
{zabbix:system.cpu.load[all,avg1].min(300)}>2&{zabbix:system.cpu.load[all,avg1].time(0)}>000000&{zabbix:sy
```

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.

3 Trigger dependencies

Trigger dependencies can be used to define relationship between triggers.

Trigger dependencies is a very convenient way of limiting number of messages to be sent in case if an event belongs to several resources.

For example, a host Host is behind router Router2 and the Router2 is behind Router1.

```
Zabbix - Router1 - Router2 - Host
```

If the Router1 is down, then obviously the Host and the Router2 are also unreachable. One does not want to receive three notifications about the Host, the Router1 and the Router2. This is when Trigger dependencies may be handy.

In this case, we define these dependencies:

```
trigger 'Host is down' depends on trigger 'Router2 is down'
trigger 'Router2 is down' depends on trigger 'Router1 is down'
```

Before changing status of trigger 'Host is down', Zabbix will check if there are corresponding trigger dependencies defined. If so, and one of the triggers is in PROBLEM state, then 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 is a trigger. Zabbix supports following trigger severities:

| SEVERITY | DEFINITION | COLOR |
|---|---|---|
| **Not classified** | Unknown severity. | Gray. |
| **Information** | For information purposes. | Light green. |
| **Warning** | Be warned. | Light yellow. |
| **Average** | Average problem. | Dark red. |
| **High** | Something important has happened. | Red. |
| **Disaster** | Disaster. Financial losses, etc. | Bright red. |

The severities are used to:

- visual representation of triggers. Different colors for different severities.
- audio alarms in Status of Triggers screen. Different audio for different severities.
- user media. Different media (notification channel) for different severities. For example, SMS - high severity, email - other.

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&{server:temp.last(0)}>20)|
({TRIGGER.VALUE}=1&{server:temp.last(0)}>15)
```

Note use of 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&{server:vfs.fs.size[/,free].max(5m)}<10G) |
({TRIGGER.VALUE}=1&{server:vfs.fs.size[/,free].min(10m)}<40G)
```

Note use of macro {TRIGGER.VALUE}. The macro returns current trigger value.

**13 Screens and Slide Shows**

Zabbix screens allow grouping of various information for quick access and display on one screen. An easy-to-use screen builder makes creating screens easy and intuitive.

A *screen* is a table which may contain the following elements in each cell:

- simple graphs
- user-defined graphs
- maps
- other screens
- plain text information
- server information (overview)
- trigger information (overview)
- data overview
- clock
- history of events
- history of actions
- URL (data taken from other location)

The number of elements per screen is unlimited.

You can configure screens in Configuration → Screens and view them in Monitoring → Screens as well as include your favourite screens in the favourites section of Monitoring → Dashboard.

**Attention:**
If graph height is set as less than 120 pixels, no trigger will be displayed in the legend.

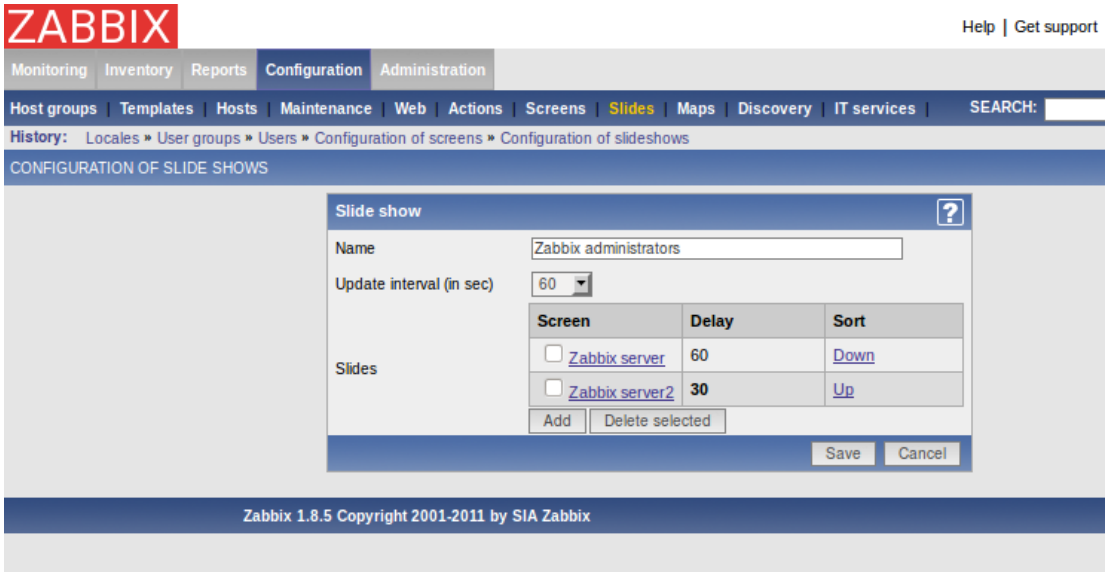
A *slide show* is a series of screens, which will be automatically rotated according to configured update intervals.

You can configure slide shows in Configuration → Slides.

| PARAMETER | Description |
|---|---|
| **Name** | Name of slide show. |
| **Update interval (in sec)** | This parameter defines the default interval between screen rotation, in seconds. |
| **Slides** | List of individual slides (screens) |
| **Screen** | Screen name |
| **Delay** | How long the screen will be displayed, in seconds. If set to 0, Update Interval of the slide show will be used. |

Example 1

Slide show ”Zabbix administrators”




The slide show consists of two screens which will be displayed in the following order:

Zabbix Server ⇒ Pause 60 seconds ⇒ Zabbix Server2 ⇒ Pause 30 seconds ⇒ Zabbix Server ⇒ Pause 60 seconds ⇒ Zabbix Server2 ⇒ ...

### 14 IT Services

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 lack of disk space, high processor load, etc. What we are interested in is 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, structure of existing IT infrastructure, and many other information of higher level.

Zabbix IT Services provide answers to all mentioned questions.

IT Services is hierarchy representation of monitored data.

A very simple IT service structure may look like:

```
IT Service
|
|-Workstations
| |
| |-Workstation1
| |
| |-Workstation2
|
|-Servers
```

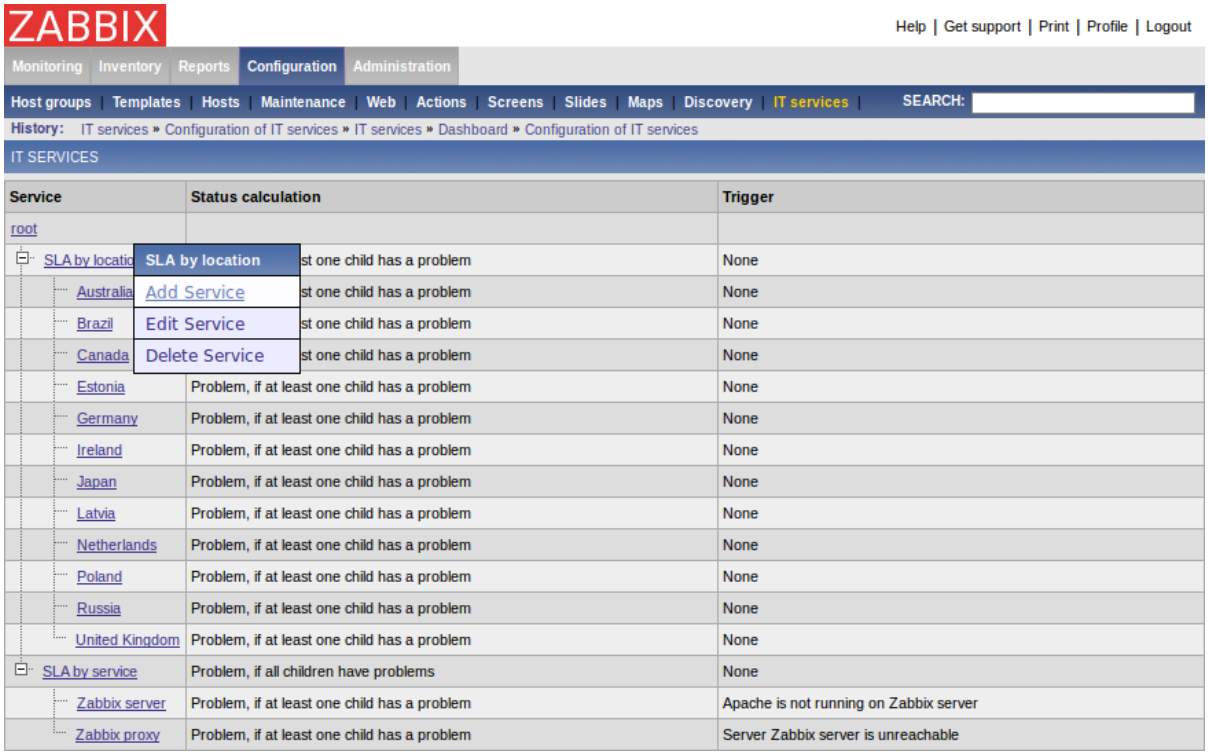
Each node of the structure has attribute status. The status is calculated and propagated to upper levels according to selected algorithm. At the lowest level of IT Services are triggers. The status of individual nodes is affected by the status of their triggers.

**Warning:**
Note that triggers with severities **Not classified** and **Information** do not impact SLA calculation.

Configuring IT Services

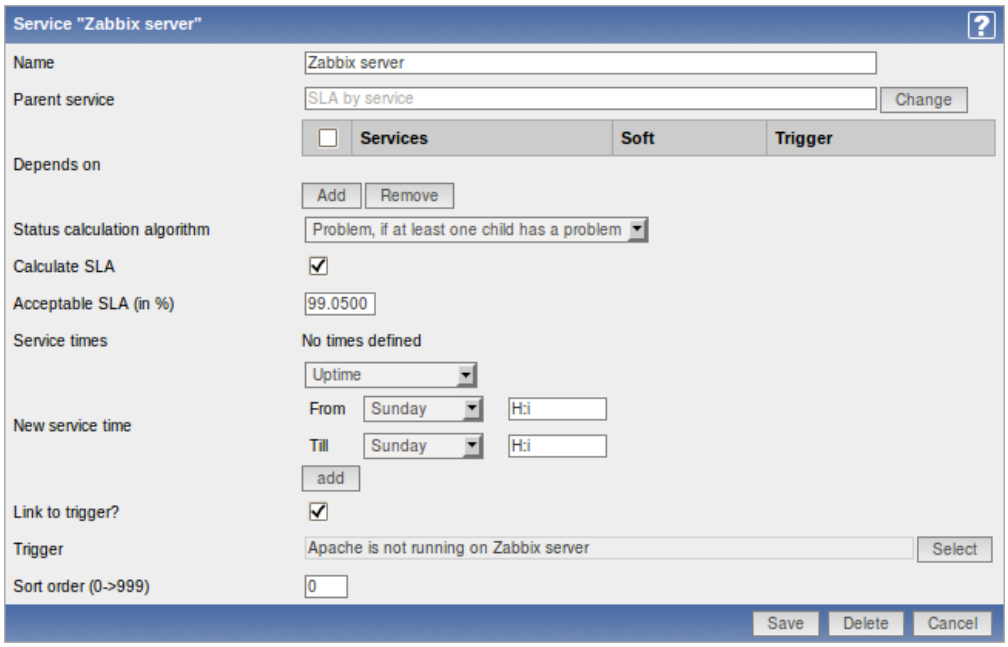
To configure IT Services, go to Configuration → IT Services.

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 service attributes.

Configuring an IT Service



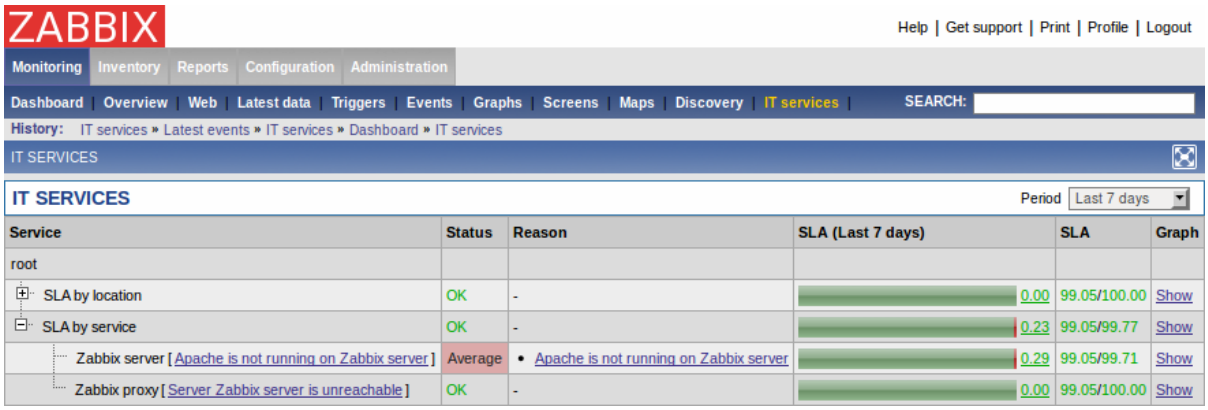

IT Service attributes:

| Parameter | Description |
|---|---|
| **Name** | Service name. |
| **Parent service** | Parent service the service belongs to. |
| **Depends on** | List of child services the service depends on. |
| **Status calculation algorithm** | Method of calculating service status:<br>**Do not calculate** - do not calculate service status<br>**Problem, if at least one child has a problem** - considered to be a problem if already one child service has a problem<br>**Problem, if all children have problems** - considered to be a problem only 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. |

| 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:<br>**One-time downtime** - a single downtime. Service state within this period does not affect SLA.<br>**Uptime** - service uptime<br>**Downtime** - service state within this period does not affect SLA.<br>Add the respective hours.<br>*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). |
| **Link to trigger** | Linkage to trigger:<br>**None** - no linkage<br>**trigger name** - linked to the trigger, thus depends on the trigger status<br>Services of the lowest level must be linked to triggers. (Otherwise their state will not be represented accurately.) |
| **Sort order** | Sort order for display, lowest comes first. |

Monitoring IT Services

To monitor IT Services, go to Monitoring → IT Services.




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:<br>**OK** - no problems<br>**(trigger colour and severity)** - indicates a problem and its severity |
| **Reason** | Indicates the reason of problem (if any). |
| **SLA (period)** | Displays SLA bar. Green/red ratio indicates the proportion of availability/problems. |
| **SLA** | Displays acceptable SLA/current SLA value. If current value is below the acceptable level, it is displayed in red. |
| **Graph** | Contains link to a graph of availability data. |

You can also click on the green/red SLA bar to access the *IT Services Availability Report*.

IT SERVICES AVAILABILITY REPORT "Zabbix server" Period Monthly Year 2011

| 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.

**15 User permissions**

All Zabbix users access the Zabbix application through the Web-based front end. Each Zabbix user is assigned a unique login name and a password. All user passwords are encrypted and stored on the Zabbix database. Users can not 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's browser can be protected using SSL.

Access permissions on screen within the menu may be set for each user. By default, no permissions are granted on a screen when user is registered to the Zabbix.

Note that a user is automatically disconnected after 30 minutes of inactivity.

1 Overview

Zabbix has a flexible user permission schema which can be efficiently used to manage user permission within one Zabbix installation or in a distributed environment.

Permissions are granted to user groups on a host group level.

Zabbix supports several types of users. The type controls what administrative functions a user has permission to.

2 User types

User types are used to define access to administrative functions and to specify default permissions.

| User type | Description |
|---|---|
| **Zabbix User** | The user has access to Monitoring menu. The user has no access to any resources by default. Permissions to host groups must be explicitly assigned. |
| **Zabbix Admin** | The user has access to Monitoring and Configuration. The user has no access to any host groups by default. Permissions to host groups must be explicitly given. |
| **Zabbix Super Admin** | The user has access to everything: Monitoring, Configuration and Administration. The user has Read-Write access to all host groups. Permissions cannot be revoked by denying access to specific host groups. |

**16 The Queue**

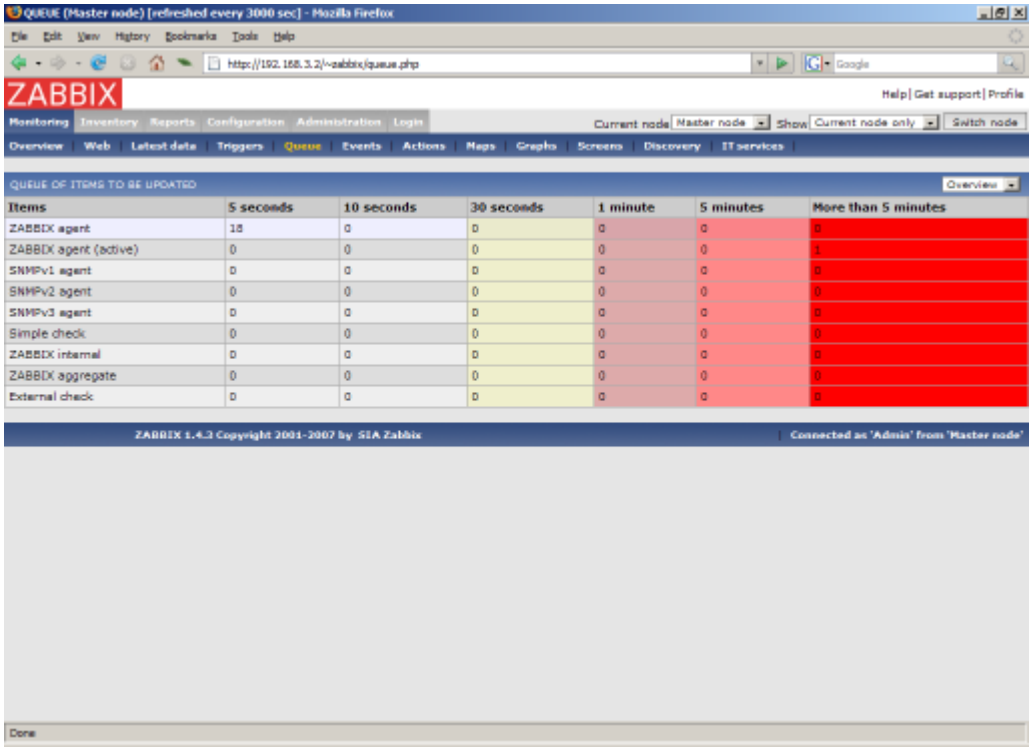
1 Overview

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

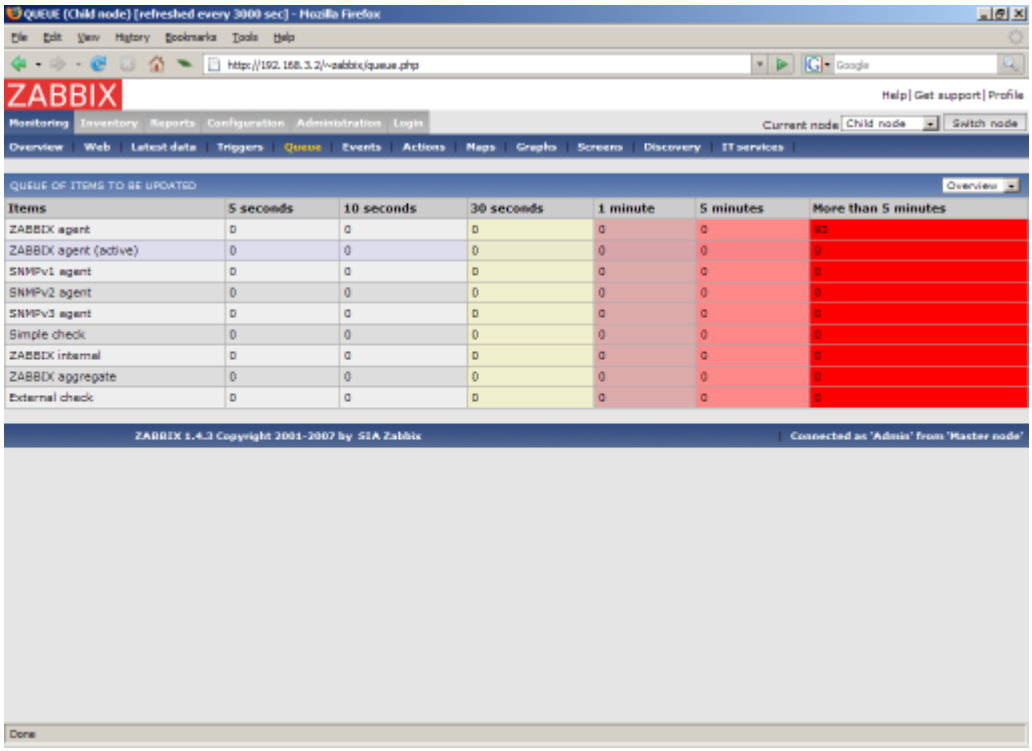
Statistics shown by the Queue is a good indicator of performance of Zabbix server.

2 How to read

The Queue on a standalone application or when displayed for a master node shows items waiting for a refresh.




In this case, we see that we have three items of type *Zabbix agent* waiting to be refreshed 0-5 seconds, and one item of type *Zabbix agent (active)* waiting more than five minutes (perhaps the agent is down?). Note that information displayed for a child node is not up-to-date. The master node receives historical data with a certain delay (normally, up-to 10 seconds for inter-node data transfer), so the information is delayed.




On the screenshot we see that there are 93 items waiting more than 5 minutes for refresh on node ”Child”, however we should not trust the information as it depends on:

- performance of the Child node
- communications between Master and Child nodes
- possible local time difference between Master and Child nodes

**Note:**
A special item key **zabbix[queue]** can be used to monitor health of the queue by Zabbix. There’s a full list of such internal items in item configuration section.

## 17 Utilities

1 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.

2 snmptrap.sh

The script is used to receive SNMP traps. The script must be used in combination with snmptrapd, which is part of package net-snmp.

Configuration guide:

- Install snmptrapd (part of net-snmp or ucd-snmp)
- Edit snmptrapd.conf.
  Add this line:

```
traphandle default /bin/bash /home/zabbix/bin/snmptrap.sh
* Copy misc/snmptrap/snmptrap.sh to ~zabbix/bin
* Edit snmptrap.sh to configure some basic parameters
* Add special host and trapper (type "string") item to Zabbix. See snmptrap.sh for the item's key.
* Run snmptrapd
```

## 18 Regular expressions

Complex regular expressions can be created and tested in the Zabbix frontend by going to Administration → General → Regular expressions.

1 Using regular expressions

After a regular expression has been created, it can be used everywhere regular expressions are supported by referring to it's name, prefixed with **@*, *for example,* @mycustomregexp*.

2 Regular expression types

All regular expressions in Zabbix, whether created with the advanced editor, or entered manually, support POSIX extended regular expressions.

## 19 Items

An item is a single performance or availability check (metric).

1 Item key

1.1 Flexible and non-flexible parameters

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

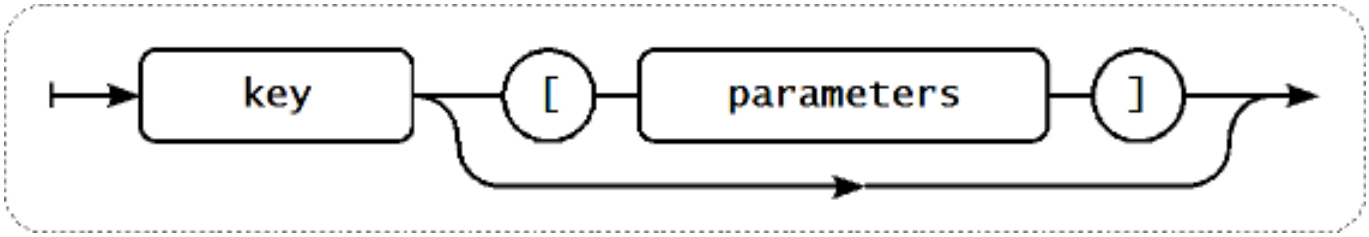
- vfs.fs.size[/]
- vfs.fs.size[/opt]

1.2 Key format

Item key format, including key parameters, must follow syntax rules. The following illustrations depict 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.

**Item key**

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.



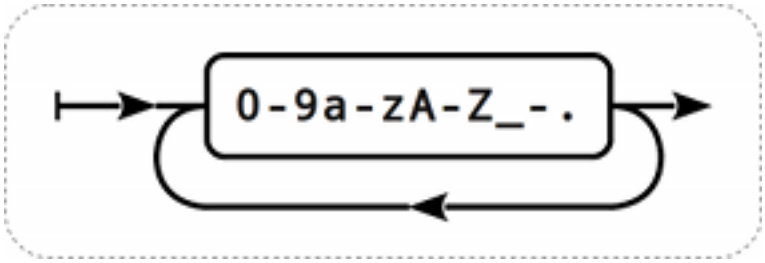

**Key name**

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

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

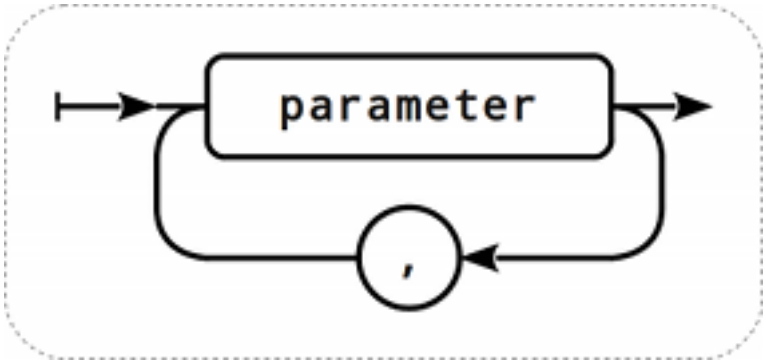
Which means:

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



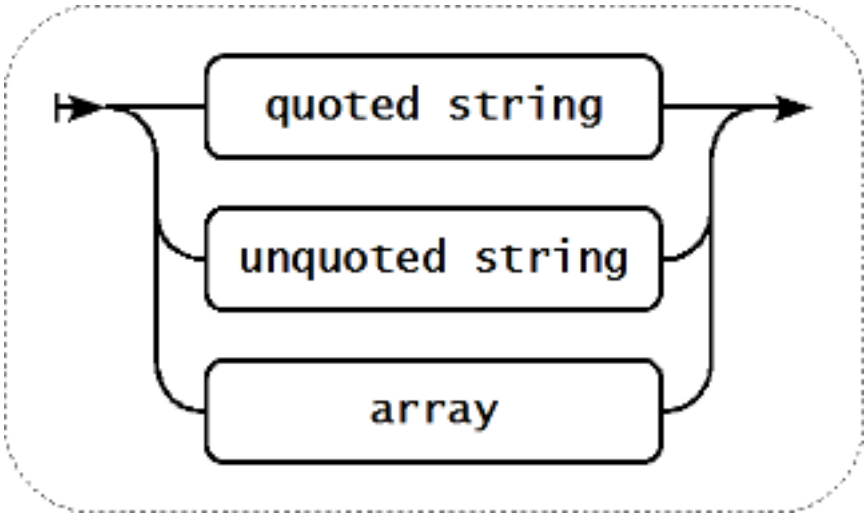

**Key parameters**

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




**Individual key parameter**

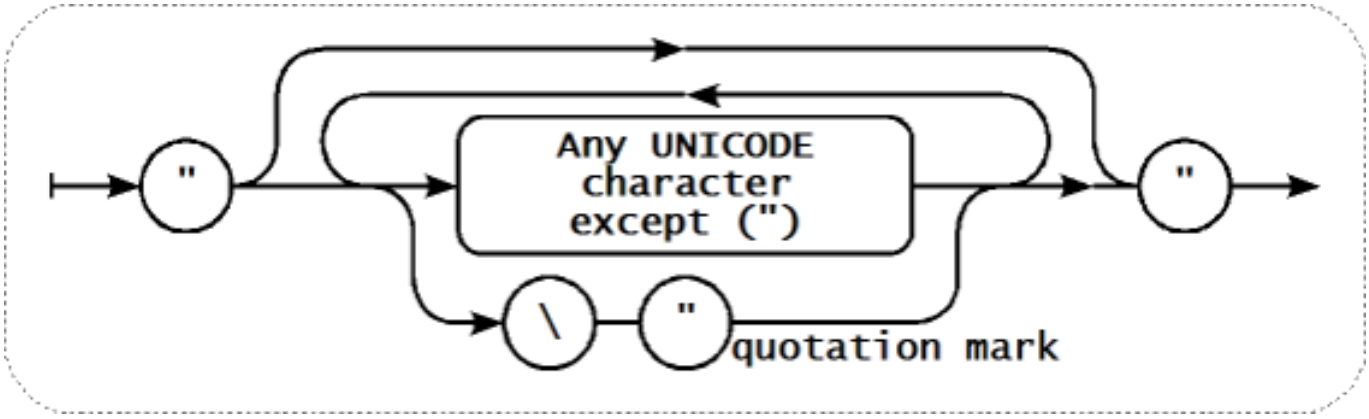
Each key parameter can be either a quoted string, an unquoted string or an array.



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.

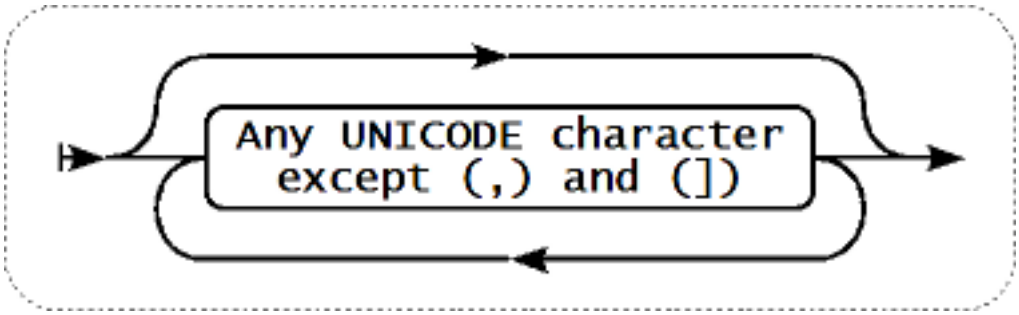
**Parameter - quoted string**

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



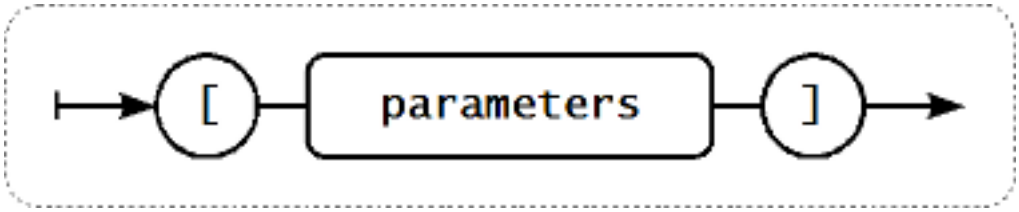

**Parameter - unquoted string**

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 following multiple parameters specifying rules and syntax.




1.3 Available encodings

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

2 Unsupported items

An item can become unsupported if its value can not be retrieved for some reason. Such items are still rechecked at a fixed interval, configurable in Administration section.

3 Supported by Platform

**Note:**
In the following lists parameters that are included in angle brackets <like_this> are optional.

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.
If a parameter is marked with ”i”, it means that it is ignored.

| NetBSD<br>OpenBSD<br>Mac<br>OS X<br>Tru64<br>AIX<br>HP-UX<br>Solaris<br>FreeBSD<br>Linux<br>2.6<br>Linux<br>2.4<br>Windows<br>Parameter<br>/ sys-<br>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>]** | X | X | X | X | X | X | X | X | X | X | X |
| **logrt[file_format,<regexp>,<encoding>,<maxlines>]** | X | X | X | X | X | X | X | X | X | X | X |
| **eventlog[name,<regexp>,<severity>,<source>,<eventid>,<br><maxlines>]** | | | | | | | | | - | - | - |
| **net.if.collisions[if]** | X | X | X | X | X | - | X | - | - | X | r |
| **net.if.in[if,<mode>]** | X | X | X | X | X | - | X | - | - | X | r |
| *mode* ▲ | bytes *(default)* | X | X | X | X | X[1] | - | X | - | - | X | r |
| | packets | X | X | X | X | X | - | X | - | - | X | r |
| | errors | X | X | X | X | X[1] | - | X | - | - | X | r |
| | dropped | X | X | X | X | - | - | - | - | - | X | r |
| **net.if.list** | X | - | - | - | - | - | - | - | - | - | - |
| **net.if.out[if,<mode>]** | X | X | X | X | X | - | X | - | - | X | r |
| *mode* ▲ | bytes *(default)* | X | X | X | X | X[1] | - | X | - | - | X | r |
| | packets | X | X | X | X | X | - | X | - | - | X | r |
| | errors | X | X | X | X | X[1] | - | X | - | - | X | r |
| | dropped | X | X | X | - | - | - | - | - | - | - | - |
| **net.if.total[if,<mode>]** | X | X | X | X | X | - | X | - | - | X | r |
| *mode* ▲ | bytes *(default)* | X | X | X | X | X[1] | - | X | - | - | X | r |
| | packets | X | X | X | X | X | - | X | - | - | X | r |
| | errors | X | X | X | X | X[1] | - | X | - | - | X | r |
| | dropped | X | X | X | - | - | - | - | - | - | - | - |
| **net.tcp.dns[<ip>,zone]** | X | X | X | X | X | X | X | X | X | X | X |
| **net.tcp.dns.query[<ip>,zone,<type>]** | X | X | X | X | X | X | X | X | X | X | X |
| **net.tcp.listen[port]** | 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 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **net.tcp.service.perf[service,<ip>,<port>]** | | X | X | X | X | X | X | X | X | - | X | X |
| **net.udp.listen[port]** | | X | X | - | - | - | - | - | - | - | - | - |
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| **proc.mem[<name>,<user>,<mode>,<cmdline>]** | | X | X | X | X | X | - | X | X | ? | X | X |
| *mode* | sum | - | X | X | X | X | - | X | X | ? | X | X |
| ▲ | *(de-fault)* | | | | | | | | | | | |
| | 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 | - | X | X | ? | X | X |
| *state* | all | - | X | X | X | X | - | X | X | ? | X | X |
| ▲ | *(de-fault)* | | | | | | | | | | | |
| | sleep | - | X | X | X | X | - | X | X | ? | X | X |
| | zomb | - | X | X | X | X | - | X | X | ? | X | X |
| | run | - | X | X | X | X | - | X | X | ? | X | X |
| **sensor[device,sensor,<mode>]** | | - | X | - | - | - | - | - | - | - | X | - |
| **services[<type>,<state>,<exclude>]** | | - | X | - | - | - | - | - | - | - | - | - |
| **system.boottime** | | - | 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 |
| *mode* | avg1 | X | X | X | X | X | X | - | X | ? | X | X |
| ▲ | *(de-fault)* | | | | | | | | | | | |
| | avg5 | X | X | X | X | X | X | - | X | ? | X | X |
| | avg15 | X | X | X | X | X | X | - | X | ? | X | X |
| **system.cpu.num[<type>]** | | X | X | X | X | X | X | X | - | - | X | X |
| *type* | online | X | X | X | X | X | X | X | - | - | X | X |
| ▲ | *(de-fault)* | | | | | | | | | | | |
| | max | - | 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 | - | X | X | X | X | X | X | X | ? | X | X |
| ▲ | *(de-fault)* | | | | | | | | | | | |
| | 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 |
| | kernel | - | - | - | - | X | - | - | - | - | - | - |
| | iowait | - | - | X | - | - | - | X | - | - | - | - |
| | wait | - | - | - | - | X | - | - | - | - | - | - |
| | interrupt | - | - | X | X | - | - | - | - | - | X | - |
| | softirq | - | - | X | - | - | - | - | - | - | - | - |
| | steal | - | - | X | - | - | - | - | - | - | - | - |
| *mode* | avg1 | X | X | X | X | - | X | X | X | ? | X | - |
| ▲ | *(de-fault)* | | | | | | | | | | | |
| | avg5 | X | X | X | X | - | X | X | - | ? | X | - |
| | avg15 | 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.localtime** | | X | X | X | X | X | X | X | X | X | X | X |
| *type* | utc | X | X | X | X | X | X | X | X | X | X | X |
| ▲ | *(de-fault)* | | | | | | | | | | | |
| | 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 | X | X | X | X | X | X | X | X | X | X | X |
| ▲ | *(de-fault)* | | | | | | | | | | | |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | nowait | X | X | X | X | X | X | X | X | X | X | X |
| **system.stat[resource,<type>]** | | - | | - | | - | | - | | X | - | - |
| **system.swap.in[<device>,<type>]** | | X | | - | | X | | - | | - | X | - |
| *(specifying a device is only supported under Linux)* | | | | | | | | | | | | |
| *type* ▲ *(pages will only work if<br>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>]** | | - | X | - | | X | - | - | - | - | X | - |
| *(specifying a device is only supported under Linux)* | | | | | | | | | | | | |
| *type* ▲ *(pages will only work if<br>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>]** | | X | X | X | X | X | - | - | X | ? | X | - |
| *type* ▲ | free *(default)* | X | X | X | X | X | - | - | X | ? | X | - |
| | total | X | X | X | X | X | - | - | X | ? | X | - |
| | used | - | X | X | X | - | - | - | - | - | X | - |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | pfree | - | X | X | X | X | - | - | - | ? | X | - |
| | pused | - | X | X | X | X | - | - | - | ? | X | - |
| **system.uname** | | X | X | X | X | X | X | X | X | - | X | X |
| **system.uptime** | | X | X | X | X | X | - | X | ? | ? | X | X |
| **system.users.num** | | - | 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 | - |
| *type* | sectors | - | X | X | - | - | - | - | - | - | - | - |
| ▲ | | | | | | | | | | | | |
| *(defaults are differ-ent<br>under vari-ous OSes)* | | | | | | | | | | | | |
| | operations | - | X | X | X | X | - | - | - | - | X | - |
| | bytes | - | - | - | X | X | - | - | - | - | X | - |
| | sps | - | X | X | - | - | - | - | - | - | - | - |
| | ops | - | X | X | X | - | - | - | - | - | - | - |
| | bps | - | - | - | X | - | - | - | - | - | - | - |
| *mode* | avg1 | - | X | X | X | - | - | - | - | - | i | - |
| ▲ *(compatible only with<br>type in: sps, ops, bps)* | *(de-fault)* | | | | | | | | | | | |
| | avg5 | - | X | X | X | - | - | - | - | - | i | - |
| | avg15 | - | X | X | X | - | - | - | - | - | i | - |
| **vfs.dev.write[<device>,<type>,<mode>]** | | X | X | X | X | X | - | - | - | - | X | - |
| *type* | sectors | - | X | X | - | - | - | - | - | - | - | - |
| ▲ | | | | | | | | | | | | |
| *(defaults are differ-ent<br>under vari-ous OSes)* | | | | | | | | | | | | |
| | operations | - | X | X | X | X | - | - | - | - | X | - |
| | bytes | - | - | - | X | X | - | - | - | - | X | - |
| | sps | - | X | X | - | - | - | - | - | - | - | - |
| | ops | - | X | X | X | - | - | - | - | - | - | - |
| | bps | - | - | - | X | - | - | - | - | - | - | - |
| *mode* | avg1 | - | X | X | X | - | - | - | - | - | i | - |
| ▲ *(compatible only with<br>type in: sps, ops, bps)* | *(de-fault)* | | | | | | | | | | | |
| | avg5 | - | X | X | X | - | - | - | - | - | i | - |
| | avg15 | - | X | X | X | - | - | - | - | - | i | - |
| **vfs.file.cksum[file]** | | 2 | X | X | X | X | X | X | X | - | X | X |
| **vfs.file.exists[file]** | | 3 | X | X | X | X | X | X | X | X | X | X |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **vfs.file.md5sum[file]** | | X | X | X | X | X | X | X | - | X | X |
| **vfs.file.regexp[file,regexp,<encoding>]** | | X | X | X | X | X | X | X | - | X | X |
| **vfs.file.regmatch[file,regexp,<encoding>]** | | X | X | X | X | X | X | X | - | X | X |
| **vfs.file.size[file]** | | 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 |
| *mode* | modify | X | X | X | X | X | X | X | - | X | X |
| ▲ | *(default)* | | | | | | | | | | |
| | access | X | X | X | X | X | X | X | - | X | X |
| | change | X | X | X | X | X | X | X | - | X | X |
| **vfs.fs.inode[fs,<mode>]** | | X | X | X | X | X | X | X | ? | X | X |
| *mode* | total | - | X | X | X | X | X | X | ? | X | X |
| ▲ | *(default)* | | | | | | | | | | |
| | free | - | 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 | X |
| **vfs.fs.size[fs,<mode>]** | | X | X | X | X | X | X | X | ? | X | X |
| *mode* | total | X | X | X | X | X | X | X | ? | X | X |
| ▲ | *(default)* | | | | | | | | | | |
| | free | X | X | X | X | X | X | X | ? | X | X |
| | used | X | X | X | X | X | X | X | ? | X | X |
| | pfree | X | X | X | X | X | X | X | ? | X | X |
| | pused | X | X | X | X | X | X | X | ? | X | X |
| **vm.memory.size[<mode>]** | | X | X | X | X | X | X | X | ? | X | X |
| *mode* | total | X | X | X | X | X | X | X | ? | X | X |
| ▲ | *(default)* | | | | | | | | | | |
| | free | X | X | X | X | X | X | X | ? | X | X |
| | used | - | - | - | X | - | - | - | - | - | X | X |
| | shared | - | X | X | X | - | - | - | - | ? | X | X |
| | buffers | - | X | X | - | - | - | - | - | ? | X | X |
| | cached | X | X | X | X | - | - | X | - | ? | X | X |
| | pfree | X | X | X | X | - | - | - | - | - | X | X |
| | pused | - | - | - | X | - | - | - | - | - | X | X |
| | available | - | 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>]** | | X | X | X | X | X | X | X | X | X | X | X |
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

[1] These values for these items are not supported for loopback interfaces on Solaris systems prior to 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.

4 Zabbix Agent

List of supported parameters

| Key | | | | |
|---|---|---|---|---|
| ▲ | **Description** | **Return value** | **Parameters** | **Comments** |
| agent.hostname | | | | |

| Key | | | | |
|---|---|---|---|---|
| | Returns agent host name. | String value | - | Returns the actual value of the agent hostname from a configuration file.<br>**This item is supported starting from version 1.8.13.** |
| agent.ping | | | | |
| | Check the agent availability. | Returns '1' if agent is available, nothing if unavailable. | - | Use function **nodata()** 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. | Number of files. Integer. | | |
| kernel.maxproc | | | | |
| | Maximum number of processes supported by OS. | Number of processes. Integer. | | |
| log[file,<regexp>,<encoding>,<maxlines>] | | | | |
| | Monitoring of log file. | Log. | **file** – full file name<br>**regexp** – regular expression for pattern<br>**encoding** - Code Page identifier<br>**maxlines** - Maximum number of new lines per second the agent will send to Zabbix Server or Proxy. This parameter overrides the 'MaxLinesPer-Second' option in zab-bix_agentd.conf | Must be configured as an Active Check.<br>Example: log[/home/zabbix/logs/logfile„<br>See detailed description. |
| logrt[file_pattern,<regexp>,<encoding>,<maxlines>] | | | | |

| Key | | | | | |
|---|---|---|---|---|---|
| | Monitoring of log file with log rotation support. | Log. | **file_pattern** – absolute path to file and regexp describing the file name pattern<br>**regexp** – regular expression describing the required content pattern<br>**encoding** - Code Page identifier<br>**maxlines** - Maximum number of new lines per second the agent will send to Zabbix Server or Proxy. This parameter overrides the 'MaxLinesPerSecond' option in zabbix_agentd.conf | Must be configured as an Active Check.<br><br>Examples:<br>logrt["/home/zabbix/logs/^log<br>9]{1,3}$",,100] - will match a file like "logfile1" (will not match ".logfile1")<br>logrt["/home/user/logfile_.*_[0<br>9]{1,3}","pattern_to_match",<br>8",100] - will collect data from files such "logfile_abc_1" or "logfile__001".<br><br>Log rotation is based on last modification times of files. See detailed description. |
| eventlog[name,<regexp>,<severity>,<source>,<eventid>,<maxlines>] | | | | | |

| Key | | | | |
|---|---|---|---|---|
| | Monitoring of event logs. | Log. | **name** – event log name<br>**regexp** – regular expression<br>**severity** – regular expression<br>The parameter accepts the following values: ”Information”, ”Warning”, ”Error”, ”Failure Audit”, ”Success Audit”<br>**source** - Source identifier<br>**eventid** - regular expression<br>**maxlines** - Maximum number of new lines per second the agent will send to Zabbix Server or Proxy. This parameter overrides the ’MaxLinesPerSecond’ option in zabbix_agentd.conf | Must be configured as an Active Check.<br><br>Examples:<br><br>eventlog[Application]<br><br>eventlog[Security,,”Failure Audit”,,529\|680]<br><br>eventlog[System,,”Warning\|Er<br><br>eventlog[System,,,,^1$]<br><br>eventlog[System,,,,@TWOSHO<br>- here custom regular expression **TWOSHORT** is defined as type **Result is TRUE** and expression itself is **^1$\|^70$**. |
| net.if.collisions[if] | Out-of-window collision. | Number of collisions. Integer. | **if** - interface | |
| net.if.in[if,<mode>] | | | | |

| Key | | | | |
|---|---|---|---|---|
| | Network interface incoming statistic. | Integer. | **if** - interface<br>**mode** – **bytes** number of bytes (default)<br>**packets** number of packets<br>**errors** number of errors<br>**dropped** number of dropped packets | Multi-byte interface names on Windows supported since Zabbix agent version 1.8.6.<br><br>Examples:<br>net.if.in[eth0,errors]<br>net.if.in[eth0]<br>You may use this key with Delta (speed per second) in order to get bytes per second statistics. |
| net.if.list | List of network interfaces:<br>Type Status IPv4 Description | String | | Supported since Zabbix agent version 1.8.1.<br>Multi-byte interface names supported since Zabbix agent version 1.8.6. Disabled interfaces are not listed.<br><br>Note that enabling/disabling some components may change their ordering in the Windows interface name. |
| net.if.out[if,<mode>] | | | | |

| Key | | | | |
|---|---|---|---|---|
| | Network interface outgoing statistic. | Integer. | **if** - interface<br>**mode** – **bytes** number of bytes (default)<br>**packets** number of packets<br>**errors** number of errors<br>**dropped** number of dropped packets | Multi-byte interface names on Windows supported since Zabbix agent version 1.8.6.<br><br>Examples:<br>net.if.out[eth0,errors]<br>net.if.out[eth0]<br>You may use this key with Delta (speed per second) in order to get bytes per second statistics. |
| net.if.total[if,<mode>] | | | | |
| | Sum of network interface incoming and outgoing statistics. | Integer. | **if** - interface<br>**mode** – **bytes** number of bytes (default)<br>**packets** number of packets<br>**errors** number of errors<br>**dropped** number of dropped packets | Examples:<br>net.if.total[eth0,errors]<br>net.if.total[eth0]<br>You may use this key with Delta (speed per second) in order to get bytes per second statistics.<br>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.dns[<ip>,zone] | | | | |
| | Checks if DNS service is up. | 0 - DNS is down<br>1 - DNS is up | **ip** - IP address of DNS server (ignored)<br>**zone** - zone to test the DNS | Example:<br>net.tcp.dns[127.0.0.1,zabbix.c<br><br>Internationalized domain names are not supported, please use IDNA encoded names instead. |
| net.tcp.dns.query[<ip>,zone,<type>] | | | | |

| Key | | | | |
|---|---|---|---|---|
| | Performs a query for the supplied DNS record type. | On success returns a character string with the required type of information. | **ip** - IP address of DNS server (ignored)<br>**zone** - zone to test the DNS<br>**type** - Record type to be queried (default is SOA) | Example: net.tcp.dns.query[127.0.0.1,za<br>**type** can be one of: A, NS, CNAME, MB, MG, MR, PTR, MD, MF, MX, SOA, NULL, WKS, HINFO, MINFO, TXT, SRV<br><br>SRV record type is supported on Unix since Zabbix agent version 1.8.6.<br><br>Internationalized domain names are not supported, please use IDNA encoded names instead. |
| net.tcp.listen[port] | Checks if this TCP port is in LISTEN state. | 0 - it is not<br>1 - it is in LISTEN state | **port** - TCP port number | Example: net.tcp.listen[80]<br><br>On Linux supported since Zabbix agent version 1.8.4 |
| net.tcp.port[<ip>,port] | Check, if it is possible to make TCP connection to port number port. | 0 - cannot connect<br>1 - can connect | **ip** - IP address(default is 127.0.0.1)<br>**port** - port number | Example: net.tcp.port[,80] can be used to test availability of web server running on port 80.<br>Old naming: check_port[*]<br>For simple TCP performance testing use net.tcp.service.perf[tcp,<ip>,<br>Note that these checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually). |

| Key | | | | |
|---|---|---|---|---|
| net.tcp.service[service,<ip>,<port>] | Check if service is running and accepting TCP connections. | 0 - service is down<br>1 - service is running | **service** - one of ssh, ntp, ldap, smtp, ftp, http, pop, nntp, imap, tcp<br>**ip** - IP address (default is 127.0.0.1)<br>**port** - port number (by default standard service port number is used) | Example: net.tcp.service[ftp„45] can be used to test availability of FTP server on TCP port 45.<br>Old naming: check_service[*]<br>Note that before Zabbix version 1.8.3 **service.ntp** should be used instead of **ntp**.<br>Note that these checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually).<br>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.<br>Checking of LDAP by Windows agent is currently not supported. |
| net.tcp.service.perf[service,<ip>,<port>] | | | | |

| Key | | | | |
|---|---|---|---|---|
| | Check performance of service | 0 - service is down<br>sec - number of seconds spent while connecting to the service | **service** - one of ssh, ntp, ldap, smtp, ftp, http, pop, nntp, imap, tcp<br>ip - IP address (default is 127.0.0.1)<br>**port** - port number (by default standard service port number is used) | Example: net.tcp.service.perf[ssh] can be used to test speed of initial response from SSH server.<br>Old naming: check_service_perf[*]<br>Note that before Zabbix version 1.8.3 **service.ntp** should be used instead of **ntp**.<br>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.<br>Checking of LDAP by Windows agent is currently not supported. |
| net.udp.listen[port] | Checks if this UDP port is in LISTEN state. | 0 - it is not<br>1 - it is in LISTEN state | **port** - UDP port number | Example: net.udp.listen[68]<br><br>On Linux supported since Zabbix agent version 1.8.4 |
| proc.mem[<name>,<user>,<mode>,<cmdline>] | | | | |

| Key | | | | |
|---|---|---|---|---|
| | Memory used by process name running under user user | Memory used by process. | **name** - process name<br>**user** - user name (default is all users)<br>**mode** - one of avg, max, min, sum (default)<br>**cmdline** - filter by command line | Example:<br>proc.mem[,root] - memory used by all processes running under user ”root”.<br>proc.mem[zabbix_server,zabb<br>- memory used by all processes zabbix_server running under user zabbix<br>proc.mem[,oracle,max,oracleZ<br>- memory used by most memory hungry process running under oracle having oracleZABBIX in its command line |
| proc.num[<name>,<user>,<state>,<cmdline>] | Number of processes name having state running under user user | Number of processes. | **name** - process name<br>**user** - user name (default is all users)<br>**state** - one of all (default), run, sleep, zomb<br>**cmdline** - filter by command line | Example:<br>proc.num[,mysql] - number of processes running under user mysql<br>proc.num[apache2,www-data] - number of apache2 running under user www-data<br>proc.num[,oracle,sleep,oracleZ<br>- number of processes in sleep state running under oracle having oracleZABBIX in its command line<br>On Windows, only name and user arguments are supported. |
| sensor[device,sensor,<mode>] | | | | |

| Key | | | | |
|---|---|---|---|---|
| | Hardware sensor reading. | | **device** - device name (if <mode> is used, it is a regular expression)<br>**sensor** - sensor name (if <mode> is used, it is a regular expression)<br>**mode** - one of avg, max, min (if omitted, device and sensor are treated verbatim). | On Linux 2.4, reads */proc/sys/dev/sensors*. Example: sensor[w83781d-i2c-0-2d,temp1] Prior to Zabbix 1.8.4, format sensor[temp1] was used.<br><br>On OpenBSD, reads *hw.sensors* MIB. Example: 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 | | | | |
| | Timestamp of system boot. | Integer. | | Time in seconds. |
| system.cpu.intr | | | | |
| | Device interrupts. | Integer. | | |
| system.cpu.load[<cpu>,<mode>] | | | | |
| | CPU load. | Processor load. Float. | **cpu** - CPU number (default is all CPUs, only default ”all” is supported)<br>**mode** - one of avg1 (default),avg5 (average within 5 minutes), avg15 | Example: system.cpu.load[] Old naming: system.cpu.loadX |
| system.cpu.num[<type>] | | | | |
| | Number of CPUs. | Number of available processors. | **type** - one of online (default), max | Example: system.cpu.num |
| system.cpu.switches | | | | |
| | Context switches. | Switches count. | | Old naming: system[switches] |
| system.cpu.util[<cpu>,<type>,<mode>] | | | | |

| Key | | | | |
|---|---|---|---|---|
| | CPU(s) utilisation. | Processor utilisation in percents | **cpu** - CPU number (default is all CPUs)<br>**type** - one of idle, nice, user (default), system, kernel, iowait, interrupt, softirq, steal<br>**mode** - one of avg1 (default),avg5 (average within 5 minutes), avg15 | Old naming: system.cpu.idleX, system.cpu.niceX, system.cpu.systemX, system.cpu.userX<br>Example: system.cpu.util[0,user,avg5] |
| system.hostname[<type>] | | | | |
| | Returns host name. | String value | **type** (only on Windows, ignored on other systems) - netbios (default) or host | On Windows the value is acquired from either GetComputerName() (for **netbios**) or gethostname() (for **host**) function and from ”hostname” command on other systems.<br><br>Example of returned value www.zabbix.com<br><br>**Parameter for this item is supported starting from version 1.8.6.** |
| system.localtime | | | | |
| | System time. | Integer or string value. | **utc** - (default) the time since the Epoch (00:00:00 UTC, January 1, 1970), measured in seconds.<br>**local** - the time in the ’yyyy-mm-dd,hh:mm:ss.nn,+hh:mm’ format | |
| system.run[command,<mode>] | | | | |

| Key | | | | |
|---|---|---|---|---|
| | Run specified command on the host. | Text result of the command | **command** - command for execution<br>**mode** - one of wait (default, wait end of execution), nowait (do not wait) | Example: system.run[ls -l /] - detailed file list of root directory.<br>Note:<br>To enable this functionality, agent configuration file must have EnableRemoteCommands=1 option. |
| system.stat[resource,<type>] | | | | |

| Key | | | |
|---|---|---|---|
| | Virtual memory statistics | Numeric value | **ent** - number of processor units this partition is entitled to receive (float)<br>**kthr,<type>** - information about kernel thread states:<br>**r** - average number of runnable kernel threads (float)<br>**b** - average number of kernel threads placed in the Virtual Memory Manager wait queue (float)<br>**memory,<type>** - information about the usage of virtual and real memory:<br>**avm** - active virtual pages (integer)<br>**fre** - size of the free list (integer)<br>**page,<type>** - information about page faults and paging activity:<br>**fi** - file page-ins per second (float)<br>**fo** - file page-outs per second (float)<br>**pi** - pages paged in from paging space (float)<br>**po** - pages paged out to paging space (float)<br>**fr** - pages freed (page replacement) (float)<br>**sr** - pages scanned by page-replacement algorithm (float)<br>**faults,<type>** - trap and |

| Key | | | | |
|---|---|---|---|---|
| system.swap.in[<device>,<type>] | Swap in (from device, into memory) statistics | Numeric value | **device** - swap device (default is all), **type** - one of 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]<br>Old naming: swap[in] |
| system.swap.out[<device>,<type>] | Swap out (from memory, onto device) statistics | Numeric value | **device** - swap device (default is all), **type** - one of 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]<br>Old naming: swap[out] |
| system.swap.size[<device>,<type>] | Swap space. | Number of bytes or percentage[1] | **device** - swap device (default is all), **type** - one of free (default, free swap space), total (total swap space), pfree (free swap space, percentage), pused (used swap space, percentage) | Example: system.swap.size[,pfree] - percentage of free swap space<br>Old naming: system.swap.free, system.swap.total |
| system.uname | Returns detailed host information. | String value | | Example of returned value:<br>FreeBSD localhost 4.4-RELEASE FreeBSD 4.4-RELEASE #0: Tue Sep 18 11:57:08 PDT 2001 murray@builder.FreeBSD.org: /usr/src/sys/compile/GENERIC i386 |

| Key | | | | |
|---|---|---|---|---|
| system.uptime | | | | |
| | System's uptime in seconds. | Number of seconds | | Use Units **s** or **uptime** to get readable values. |
| system.users.num | | | | |
| | Number of users connected. | Number of users | | Command **who** is used on agent side. |
| vfs.dev.read[<device>,<type>,<mode>] | | | | |
| | Disk read statistics. | Integer for **type** in: sectors, operations, bytes<br>Float for **type** in: sps, ops, bps | **device** - disk device (default is "all"[2])<br>**type** - one of sectors, operations, bytes, sps, ops, bps (must specify exactly which parameter to use, since defaults are different under various OSes).<br>sps, ops, bps means: sectors, operations, bytes per second respectively<br>**mode** - one of avg1 (default),avg5 (average within 5 minutes), avg15.<br>Compatible only with **type** in: sps, ops, bps | Default values of 'type' parameter for different OSes:<br>FreeBSD - bps<br>Linux - sps<br>OpenBSD - operations<br>Solaris - bytes<br><br>Example: vfs.dev.read[,operations]<br>Old naming: io[*]<br><br>The **type** parameters ops, bps and sps on supported platforms are limited to 8 devices (7 individual devices and one "all").<br><br>Supports LVM since Zabbix 1.8.6.<br><br>Until Zabbix 1.8.6, only relative device names may be used (for example, **sda**), since 1.8.6 optional **/dev/** prefix may be used (for example, **/dev/sda**) |
| vfs.dev.write[<device>,<type>,<mode>] | | | | |

| Key | | | | |
|---|---|---|---|---|
| | Disk write statistics. | Integer for **type** in: sectors, operations, bytes<br>Float for **type** in: sps, ops, bps | **device** - disk device (default is "all"[2])<br>**type** - one of sectors, operations, bytes, sps, ops, bps (must specify exactly which parameter to use, since defaults are different under various OSes).<br>sps, ops, bps means: sectors, operations, bytes per second respectively<br>**mode** - one of avg1 (default),avg5 (average within 5 minutes), avg15.<br>Compatible only with **type** in: sps, ops, bps | Default values of 'type' parameter for different OSes:<br>FreeBSD - bps<br>Linux - sps<br>OpenBSD - operations<br>Solaris - bytes<br><br>Example: vfs.dev.write[,operations]<br>Old naming: io[*]<br><br>The **type** parameters ops, bps and sps on supported platforms are limited to 8 devices (7 individual devices and one "all").<br><br>Supports LVM since Zabbix 1.8.6.<br><br>Until Zabbix 1.8.6, only relative device names may be used (for example, **sda**), since 1.8.6 optional **/dev/** prefix may be used (for example, **/dev/sda**) |
| vfs.file.cksum[file] | | | | |
| | Calculate file checksum | File checksum, calculated by algorithm used by UNIX cksum. | **file** - full path to file | Example of returned value: 1938292000<br><br>Example: vfs.file.cksum[/etc/passwd]<br><br>Old naming: cksum |
| vfs.file.exists[file] | | | | |
| | Check if file exists | 1 - regular file or a link (symbolic or hard) to regular file exists.<br><br>0 - otherwise | **file** - full path to file | Example: vfs.file.exists[/tmp/application<br><br>The return value depends on what S_ISREG POSIX macro returns. |

| Key | | | | |
|---|---|---|---|---|
| vfs.file.md5sum[file] | File's MD5 checksum | MD5 hash of the file. | **file** - full path to file | Example of returned value: b5052decb577e0fffd622d6dd<br><br>Example: vfs.file.md5sum[/etc/zabbix/za<br><br>The file size limit (64 MB) for this item was removed in version 1.8.6. |
| vfs.file.regexp[file,regexp,<encoding>] | Find string in a file | Matched string or EOF if expression not found | **file** - full path to file<br>**regexp** - GNU regular expression<br>**encoding** - Code Page identifier | Only the first matching line is returned.<br>Example: vfs.file.regexp[/etc/passwd,zal |
| vfs.file.regmatch[file,regexp,<encoding>] | Find string in a file | 0 - expression not found<br>1 - found | **file** - full path to file<br>**regexp** - GNU regular expression<br>**encoding** - Code Page identifier | Example: vfs.file.regmatch[/var/log/app. |
| vfs.file.size[file] | File size | Size in bytes. | **file** - full path to file | File must have read permissions for user zabbix<br><br>Example: vfs.file.size[/var/log/syslog] |
| vfs.file.time[file,<mode>] | File time information. | Unix timestamp. | **file** - full path to the file<br>**mode** - one of modify (default, modification time), access - last access time, change - last change time | Example: vfs.file.time[/etc/passwd,modi |
| vfs.fs.inode[fs,<mode>] | Number of inodes | Numeric value | **fs** - filesystem<br>**mode** - one of total (default), free, used, pfree (free, percentage), pused (used, percentage) | Example: vfs.fs.inode[/,pfree]<br>Old naming: vfs.fs.inode.free[*], vfs.fs.inode.pfree[*], vfs.fs.inode.total[*] |

| Key | | | | |
|---|---|---|---|---|
| vfs.fs.size[fs,<mode>] | | | | |
| | Disk space | Disk space in bytes | **fs** - filesystem<br>**mode** - one of total (default), free, used, pfree (free, percentage), pused (used, percentage) | In case of a mounted volume, disk space for local file system is returned.<br>Example: vfs.fs.size[/tmp,free]<br>Old naming: vfs.fs.free[*], vfs.fs.total[*], vfs.fs.used[*], vfs.fs.pfree[*], vfs.fs.pused[*] |
| vm.memory.size[<mode>] | | | | |
| | Memory size | Memory size in bytes | **mode** - one of total (default), shared, free, buffers, cached, pfree, available | 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<br>**path** - path to HTML document (default is /)<br>**port** - port number (default is 80) | Returns EOF on fail. Example: web.page.get[www.zabbix.cor |
| web.page.perf[host,<path>,<port>] | | | | |
| | Get timing of loading full web page | Time in seconds | **host** - hostname<br>**path** - path to HTML document (default is /)<br>**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>] | | | | |
| | Get first occurrence of regexp in web page | Matched string | **host** - hostname<br>**path** - path to HTML document (default is /)<br>**port** - port number (default is 80)<br>**regexp** - GNU regular expression<br>**length** - maximum number of characters to return | Returns EOF in case of no match or any other failures (such as timeout, failed connection, etc). Example: web.page.regexp[www.zabbix |

See this section for the difference of an item being performed as a passive or an active check.

**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:**
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.

5 Windows-specific parameters

This section contains descriptions of parameters supported by Zabbix Windows agent only.

| Key | Description | Return value | Comments |
|---|---|---|---|
| ▲ perf_counter[counter,<interval>] | Value of any performance counter, where "counter" is the counter path, and "interval" is the time period for storing the average value. | Average value of the "counter" during last "interval" seconds. 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. |
| service_state[service] | State of service. Parameter is service name. | 0 – running<br>1 – paused<br>2 - start pending<br>3 - pause pending<br>4 - continue pending<br>5 - stop pending<br>6 – stopped<br>7 - unknown<br>255 – no such service | Parameter must be real service name as seen in service properties under "Name:" or name of EXE file. |

| Key | | | |
|---|---|---|---|
| services[<type>,<state>,<exclude>] | List of services, separated by a newline or 0, if list would be empty. | **type** - one of all (default), automatic, manual, disabled<br>**state** - one of all (default), stopped, started, start_pending, stop_pending, running, continue_pending, pause_pending, paused<br>**exclude** - list of services to exclude it from the result. Excluded services should be written in double quotes, separated by comma, without spaces.<br>**This parameter is supported starting from version 1.8.1.** | 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 |
| proc_info[process,<attribute>,<type>] | | | |

| Key | | | | |
|---|---|---|---|---|
| | | Different information about specific process(es). | **process** - process name<br>**attribute** - requested process attribute.<br>**type** - representation type (meaningful when more than one process with the same name exists) | The following attributes are currently supported:<br>**vmsize** - Size of process virtual memory in Kbytes<br>**wkset** - Size of process working set (amount of physical memory used by process) in Kbytes<br>**pf** - Number of page faults<br>**ktime** - Process kernel time in milliseconds<br>**utime** - Process user time in milliseconds<br>**io_read_b** - Number of bytes read by process during I/O operations<br>**io_read_op** - Number of read operation performed by process<br>**io_write_b** - Number of bytes written by process during I/O operations<br>**io_write_op** - Number of write operation performed by process<br>**io_other_b** - Number of bytes transferred by process during operations other than read and write operations<br>**io_other_op** - Number of I/O operations performed by process, other than read and write operations<br>**gdiobj** - Number of GDI objects used by process<br>**userobj** - Number of USER objects used by process<br><br>Valid types are:<br>**min** - minimal value among all |

| Key |
|---|

## 6 SNMP Agent

Zabbix must be configured with SNMP support in order to be able to retrieve data provided by SNMP agents.

**Warning:**
If monitoring SNMPv3 devices, make sure that msgAuthoritativeEngineID (also known as snmpEngineID or "Engine ID") is never shared by two devices. It must be unique for each device.

**Note:**
For SNMPv3 privacy and authentication currently MD5 and DES protocols are supported.

The following steps have to be performed in order to add monitoring of SNMP parameters:

Step 1

Create a host for the SNMP device.

Enter an IP address. Set the host Status to NOT MONITORED. You can use one of the SNMP templates (Template_SNMPv1_Device, Template_SNMPv2_Device), which will automatically add the set of items. However, the template may not be compatible with the host.

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

Step 2

Find out the SNMP string of the item you want to monitor.

After creating the host, use 'snmpwalk' (part of ucd-snmp/net-snmp software which you should have installed as part of the Zabbix installation) or equivalent tool:

```
shell> snmpwalk <host or host IP> public
```

This will 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 would then go through the list until you find the string you want to monitor, e.g. you wanted to monitor the bytes coming in to your switch on port 3 you would use:

```
interfaces.ifTable.ifEntry.ifOctetsIn.3 = Counter 32: 614794138
```

You should now use the snmpget command to find the OID for interfaces.ifTable.ifEntry.ifInOctets.3:

```
shell> snmpget -On 10.62.1.22 interfaces.ifTable.ifEntry.ifOctetsIn.3
```

where the last number in the string is the port number you are looking to monitor. This should give you something like the following:

```
.1.3.6.1.2.1.2.2.1.10.3 = Counter32: 614794138
```

again the last number in the OID is the port number.

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

Step 3

Create an item for monitoring.

So, now go back to Zabbix and click on Items, selecting 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 a new item box. 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 enter a plain English description in the 'Description' field of the new item box. Make sure the 'Host' field has your switch/router in it and change the 'Type' field to "SNMPv* agent". Enter the community (usually public) and enter the numeric OID that you retrieved earlier in to the 'SNMP OID' field, i.e. .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 Multiplier if you want one and enter an 'update interval' and 'keep history' if you want it to be different from the default. Set the 'Status' to Monitored, the 'Type of information' to *Numeric (float)* and the 'Store value' to DELTA (important otherwise you will get cumulative values from the SNMP device instead of the latest change).

Now save the item and go back to the hosts area of Zabbix. From here check that the SNMP device Status shows 'Monitored' and check in *Latest data* for your SNMP data!

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><br>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 either --with-net-snmp or --with-ucd-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 |

7 Simple checks

Simple checks are normally used for agent-less monitoring or for remote checks of services. Note that Zabbix agent is not needed for simple checks. Zabbix server is responsible for processing of simple checks (making external connections, etc).

All simple checks, except tcp and tcp_perf, accept one optional parameter:

- port - port number. If missing, standard default service port is used.

Examples of using simple checks:

```
ftp,155
http
http_perf,8080
```

**Attention:**
IP is taken from the Zabbix host definition.

**Warning:**
Checking of encrypted protocols (like IMAP on port 993 or POP on port 995) is currently not supported. As a workaround, please use tcp and tcp_perf for checks like these.

List of supported simple checks:

| Key | Description | Return value |
|---|---|---|
| ▲<br>**ftp,<port>** | Checks if FTP server is running and accepting connections | 0 - FTP server is down<br>1 - FTP server is running |
| **ftp_perf,<port>** | | |

| Key | | |
|---|---|---|
| | Checks if FTP server is running and accepting connections | 0 - FTP server is down<br>Otherwise, number of seconds spent connecting to FTP server. |
| **http,<port>** | | |
| | Checks if HTTP server is running and accepting connections | 0 - HTTP server is down<br>1 - HTTP server is running |
| **http_perf,<port>** | | |
| | Checks if HTTP (web) server is running and accepting connections | 0 - HTTP (web) server is down<br>Otherwise, number of seconds spent connecting to HTTP server. |
| **icmpping[<target>,<packets>,<interval>,<size>,<timeout>]** | | |
| | Checks if server is accessible by ICMP ping<br>**target** - host IP or DNS name<br>**packets** - number of packets<br>**interval** - time between successive packets in milliseconds<br>**size** - packet size in bytes<br>**timeout** - timeout in milliseconds | 0 - ICMP ping fails<br>1 - ICMP ping successful<br><br>Example:<br>**icmpping[,4]** - if at least one packet of the four is returned, the item will return 1. |
| **icmppingloss[<target>,<packets>,<interval>,<size>,<timeout>]** | | |
| | Return percentage of lost packets<br>**target** - host IP or DNS name<br>**packets** - number of packets<br>**interval** - time between successive packets in milliseconds<br>**size** - packet size in bytes<br>**timeout** - timeout in milliseconds | Loss of packets in percents |
| **icmppingsec[<target>,<packets>,<interval>,<size>,<timeout>,<mode>]** | | |
| | Return ICMP ping response time<br>**target** - host IP or DNS name<br>**packets** - number of packets<br>**interval** - time between successive packets in milliseconds<br>**size** - packet size in bytes<br>**timeout** - timeout in milliseconds<br>**mode** - one of min, max, avg (default) | Number of seconds<br><br>If host is not available (timeout reached), the item will return 0. |

| Key | | |
|---|---|---|
| **imap,<port>** | Checks if IMAP server is running and accepting connections | 0 - IMAP server is down<br>1 - IMAP server is running |
| **imap_perf,<port>** | Checks if IMAP server is running and accepting connections | 0 - IMAP server is down<br>Otherwise, number of seconds spent connecting to IMAP server. |
| **ldap,<port>** | Checks if LDAP server is running and accepting connections | 0 - LDAP server is down<br>1 - LDAP server is running |
| **ldap_perf,<port>** | Checks if LDAP server is running and accepting connections | 0 - LDAP server is down<br>Otherwise, number of seconds spent connecting to LDAP server. |
| **nntp,<port>** | Checks if NNTP server is running and accepting connections | 0 - NNTP server is down<br>1 - NNTP server is running |
| **nntp_perf,<port>** | Checks if NNTP server is running and accepting connections | 0 - NNTP server is down<br>Otherwise, number of seconds spent connecting to NNTP server. |
| **ntp,<port>** | Checks if NTP server is running and accepting connections | 0 - NTP server is down<br>1 - NTP server is running |
| **ntp_perf,<port>** | Checks if NTP server is running and accepting connections | 0 - NTP server is down<br>Otherwise, number of seconds spent connecting to NTP server. |
| **pop,<port>** | Checks if POP server is running and accepting connections | 0 - POP server is down<br>1 - POP server is running |
| **pop_perf,<port>** | Checks if POP server is running and accepting connections | 0 - POP server is down<br>Otherwise, number of seconds spent connecting to POP server. |
| **smtp,<port>** | Checks if SMTP server is running and accepting connections | 0 - SMTP server is down<br>1 - SMTP server is running |
| **smtp_perf,<port>** | | |

| Key | | | |
|---|---|---|---|
| | | Checks if SMTP server is running and accepting connections | 0 - SMTP server is down<br>Otherwise, number of seconds spent connecting to SMTP server. |
| **ssh,<port>** | | | |
| | | Checks if SSH server is running and accepting connections | 0 - SSH server is down<br>1 - SSH server is running |
| **ssh_perf,<port>** | | | |
| | | Checks if SSH server is running and accepting connections | 0 - SSH server is down<br>Otherwise, number of seconds spent connecting to SSH server. |
| **tcp,port** | | | |
| | | Checks if TCP service is running and accepting connections | 0 - TCP service is down<br>1 - TCP service is running |
| **tcp_perf,port** | | | |
| | | Checks if TCP service is running and accepting connections | 0 - the service on the port is down<br>Otherwise, number of seconds spent connecting to the TCP service. |

Timeout processing

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

ICMP pings

Zabbix uses external utility **fping** for processing of ICMP pings. The utility is not part of Zabbix distribution and has to be additionally installed. If the utility is missing, has wrong permissions or its location does not match **FpingLocation** defined in configuration file, ICMP pings (**icmpping**, **icmppingsec** and **icmppingloss**) will not be processed.

**fping** must be executable by user Zabbix daemons run as and setuid root. Run these commands as user **root** in order to setup 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.

The default values for ICMP checks parameters:

| Parameter | Value | Description | fping flag | Min | Max |
|---|---|---|---|---|---|
| packets | 3 | pings to the target | -C | 1 | 10000 |
| interval | 1000 | milliseconds, ”fping” default | -p | 20 | |
| size | 56 or 68 | bytes, ”fping” default; 56 bytes on x86, 68 bytes on x86_64 | -b | 24 | 65507 |
| timeout | 500 | milliseconds, ”fping” default | -t | 50 | |

**Warning:**
Warning: fping defaults can differ depending on platform and version - if in doubt, check fping documentation.

Zabbix writes addresses to be checked to a temporary file, which is then passed to **fping**. If items have different parameters, only ones with identical parameters are written to a single file.

8 Internal checks

Internal checks allow monitoring of the internals of Zabbix. Internal checks are calculated by Zabbix server.

**Note:**
Internal checks are still processed by Zabbix pollers.

| Key | | |
|---|---|---|
| ▲ | Description | Comments |
| zabbix[boottime] | Startup time of Zabbix server process in seconds. | In seconds since the epoch. |
| zabbix[history] | Number of values stored in table HISTORY | Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! |
| zabbix[history_log] | Number of values stored in table HISTORY_LOG | Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! **This item is supported starting from version 1.8.3.** |
| zabbix[history_str] | Number of values stored in table HISTORY_STR | Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! |
| zabbix[history_text] | Number of values stored in table HISTORY_TEXT | Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! **This item is supported starting from version 1.8.3.** |
| zabbix[history_uint] | Number of values stored in table HISTORY_UINT | Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! **This item is supported starting from version 1.8.3.** |
| zabbix[items] | Number of items in Zabbix database | |
| zabbix[items_unsupported] | | |

| Key | | |
|---|---|---|
| | Number of unsupported items in Zabbix database | |
| zabbix[log] | | |
| | Stores warning and error messages generated by Zabbix server. | Character. Add item with this key to have Zabbix internal messages stored. |
| 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 last minute only.<br><br>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.<br>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.<br>Processes report what they are doing in shared memory and the self-monitoring process summarizes that data each second. State changes (busy/idle) are registered upon change - thus a process that becomes busy registers as such and doesn't change | The following process types are currently supported: **alerter** - process for sending notifications **configuration syncer** - process for managing in-memory cache of configuration data **db watchdog** - sender of a warning message in case DB is not available **discoverer** - process for discovery of devices **escalator** - process for escalation of actions **history syncer** - history DB writer **http poller** - web monitoring poller **housekeeper** - process for removal of old historical data **icmp pinger** - poller for icmpping checks **ipmi poller** - poller for IPMI checks **node watcher** - process for sending historical data and configuration changes between nodes **self-monitoring** - process for collecting internal server statistics **poller** - normal poller for passive checks |

| Key | | |
|---|---|---|
| zabbix[proxy,<name>,<param>] | | |
| | Access to Proxy related information. | <name> - Proxy name<br>List of supported parameters (<param>):<br>lastaccess – timestamp of last heart beat message received from Proxy<br>For example, zab-bix[proxy,”Germany”,lastacce<br>Trigger function fuzzytime() can be used to check availability of proxies. |
| zabbix[queue,<from>,<to>] | | |
| | Number of server monitored items in the Queue which are delayed by <from> to <to> seconds, inclusive. | <from> - default: 6 seconds<br><to> - default: infinity<br>Suffixes s,m,h,d,w are supported for these parameters.<br>**Parameters from and to are supported starting from version 1.8.3.** |
| zabbix[requiredperformance] | | |
| | Required performance of the Zabbix server, in new values per second expected. | Approximately correlates with ”Required server performance, new values per second” in *Reports → Status of Zabbix*.<br>Supported since Zabbix 1.6.2. |
| zabbix[trends] | | |
| | Number of values stored in table TRENDS | Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! |
| zabbix[trends_uint] | | |

| Key | | | | |
|---|---|---|---|---|
| | Number of values stored in table TRENDS_UINT | | | Do not use if MySQL InnoDB, Oracle or PostgreSQL is used!<br>**This item is supported starting from version 1.8.3.** |
| zabbix[triggers] | | | | |
| | Number of triggers in Zabbix database | | | |
| zabbix[uptime] | | | | |
| | Uptime of Zabbix server process in seconds. | | | |
| zabbix[wcache,<cache>,<mode>] | | | | |
| | **Cache** | **Mode** | | |
| | values | all | Number of values processed by Zabbix server, except not supported. | Counter. |
| | | float | | Counter. |
| | | uint | | Counter. |
| | | str | | Counter. |
| | | log | | Counter. |
| | | text | | Counter. |
| | | not supported | Number of processed not supported items. | Counter.<br>**This item is supported starting from version 1.8.6.** |
| | history | pfree | Free space in the history buffer in percentage. | Low number indicates performance problems on the database side. |
| | | total | | |
| | | used | | |
| | | free | | |
| | trend | pfree | | |
| | | total | | |
| | | used | | |
| | | free | | |
| | text | pfree | | |
| | | total | | |
| | | used | | |
| | | free | | |
| zabbix[rcache,<cache>,<mode>] | | | | |
| | **Cache** | **Mode** | | |
| | buffer | pfree | | |
| | | total | | |
| | | used | | |
| | | free | | |

9 Aggregated checks

Aggregate checks do not require any agent running on a host being monitored. Zabbix server collects aggregate information by doing direct database queries.

Syntax of an aggregate item's key

```
groupfunc["Host group","Item key",itemfunc,parameter]
```

Multiple host groups may be used since Zabbix 1.8.2 by inserting a comma-delimited array.

Supported group functions:

| GROUP FUNCTION | DESCRIPTION |
|---|---|
| **grpavg** | Average value |
| **grpmax** | Maximum value |
| **grpmin** | Minimum value |
| **grpsum** | Sum of values |

Supported item functions:

| ITEM FUNCTION | DESCRIPTION |
|---|---|
| **avg** | Average value |
| **count** | Number of values |
| **last** | Last value |
| **max** | Maximum value |
| **min** | Minimum value |
| **sum** | Sum of values |

**Warning:**
Amount of values (prefixed with **#**) is not supported.

Examples of keys for aggregate items:

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

Average (5min) number of queries per second for host group 'MySQL Servers'

```
grpavg["MySQL Servers",mysql.qps,avg,300]
```

Example 4

Average CPU load on all hosts in multiple host groups.

```
grpavg[["Servers A","Servers B","Servers C"],system.cpu.load,last,0]
```

10 External checks

External check is a check executed by Zabbix Server by running a shell script or a binary.

External checks do not require any agent running on a host being monitored.

Syntax of item's key:

```
script[parameters]
* script – name of the script.
* parameters – list of command line parameters. Parameters will be used in command line without any change
```

If you don't want to pass your parameters to the script you may use:

```
script[] or
script   <- this simplified syntax is supported starting from Zabbix 1.8.1
```

Zabbix server will find and execute the command in the directory defined in configuration parameter **ExternalScripts** in zabbix_server.conf. 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.

> **Note:**
> This directory is located on the Zabbix server. For custom command execution using Zabbix agents see user parameter documentation.

First command line parameter is host IP address or DNS name, other parameters are substituted by **parameters**.

Zabbix uses the first line (trimmed from trailing whitespace) in the standard output of the script as the value. The following lines, standard error and the exit code are discarded.

> **Warning:**
> Do not overuse external checks! It can decrease performance of the Zabbix system a lot.

Example 1

Execute script check_oracle.sh with parameters "-h 192.168.1.4". Host DNS name 'www1.company.com'.

```
check_oracle.sh[-h 192.168.1.4]
```

Zabbix will execute:

```
check_oracle.sh www1.company.com -h 192.168.1.4.
```

11 SSH checks

Zabbix must be configured with SSH2 support. ::: noteimportant The minimal supported libssh2 library version is 1.0.0. :::

SSH checks are used for agent-less monitoring. Note that Zabbix agent is not needed for SSH checks.

Actual commands 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.

| Key | Description | Comments |
|---|---|---|
| **ssh.run[<unique short description>,<ip>,<port>,<encoding>]** | Run a command by using SSH remote session | |

12 Telnet checks

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

Actual commands 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.

Till version 1.8.1, supported characters that the prompt can end with:

- $
- #
-

Zabbix version 1.8.2 adds support for additional character:

- %

| Key | Description | Comments |
|---|---|---|
| **telnet.run[<unique short description>,<ip>,<port>,<encoding>]** | Run a command on a remote device using telnet connection | |

13 Calculated items

**Attention:**
Support of calculated items was introduced in Zabbix 1.8.1

With calculated items you can create calculations on the basis of other items. Thus, calculated items are a way of creating virtual data sources. Item values will be periodically calculated based on an arithmetical expression.

Resulting data will be stored in the Zabbix database as for any other item - this means storing both history and trends 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**. 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 (named **Expression** in 1.8.1 and 1.8.2). There is virtually no connection between the formula and key. The key parameters are not used in formula in any way - variables may be passed to the formula with user macros.

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**.<br>*Note:* Putting the whole key in double quotes (”...”) is strongly recommended to avoid incorrect parsing because of spaces or commas within the key.<br>If there are also quoted parameters within the key, those double quotes must be escaped by using the backslash (\). See **Examples 5 and 6** below. |
| **parameter(s)** | Any additional parameters that may be required. See **Example 5** below. |

**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.

A more complex formula may use a combination of functions, operators and brackets. You could use all functions and operators supported in trigger expressions. Note that syntax is slightly different, however logic and operator precedence are exactly the same.

Supported characters for a hostname:

```
a..zA..Z0..9 ._-
```

Supported characters for a key:

```
a..zA..Z0..9.,_
```

Supported characters for a function:

```
a..zA..Z0..9_
```

Unlike trigger expressions, Zabbix processes calculated items according to item update interval, not upon receiving a new value.

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

Example 1

Calculate 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

Calculate 10 minute average 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 the Zabbix Server.

Example 3

Calculate total bandwidth on eth0.

Sum of two functions:

```
last("net.if.in[eth0,bytes]")+last("net.if.out[eth0,bytes]")
```

Example 4

Calculate 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

Calculate count of records in a log file for last 10 minutes.

Take note of how double quotes are escaped within the quoted key and first function parameter is required:

```
count("logrt[\"/tmp/test.log\",\"some words pattern\"]",600)
```

Example 6

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.
```

**20 Frontend definitions**

While many things in the frontend can be configured using the frontend itself, some customisations are currently only possible by editing a definitions file. Located in the frontend directory, this file is **include/defines.inc.php**. Parameters in this file that could be of interest to users:

- TRIGGER_FALSE_PERIOD

For how long to show triggers in OK state after their state changed from PROBLEM, in seconds.

Default: 1800

- TRIGGER_BLINK_PERIOD

For how long a trigger should blink after its state changed, in seconds.

Default: 1800

- 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.

- GRAPH_YAXIS_SIDE_DEFAULT

Default location of Y axis in simple graphs and default value for drop down box when adding items to custom graphs. Possible values: 0 - left, 1 - right.

Default: 0

- ZBX_UNITS_ROUNDOFF_THRESHOLD

Threshold value for roundoff constants. Values less than it will be rounded to ZBX_UNITS_ROUNDOFF_LOWER_LIMIT number of digits after comma, greater to ZBX_UNITS_ROUNDOFF_UPPER_LIMIT.

Default: 0.01

- ZBX_UNITS_ROUNDOFF_UPPER_LIMIT

Number of digits after comma, when value is greater than roundoff threshold

Default: 2

- ZBX_UNITS_ROUNDOFF_LOWER_LIMIT

Number of digits after comma, when value is less than roundoff threshold

Default: 6

- ZBX_HISTORY_DATA_UPKEEP (available since 1.8.4)

Number of days, which will reflect on frontend choice when deciding which history or trends table to process for selected period on data graphing. When this define is:

```
* less than zero - zabbix takes item values for selected graph period configured in item "keep in histor
* equal to zero - zabbix takes item values only from trends;
* greater then zero - zabbix overwrites item "keep in history" configured value with this define;
```

This define could be useful for partitioned history data storage.

Default: -1

- ZAPCAT_COMPATIBILITY

Enables support for Zapcat Zabbix Java JMX bridge item keys syntax

Default: false

> **Warning:**
> ZAPCAT_COMPATIBILITY is only available for 1.8.4.

**21 Suffixes**

It is possible to simplify Zabbix trigger expressions or item keys by using suffixes.

1 Standard multipliers

The following table summarises available standard multipliers in Zabbix frontend and server:

| | Till_1.8.2 | Additional in 1.8.2 |
|---|---|---|
| Server | K (Kilo) | T (Tera) |
| | M (Mega) | |
| | G (Giga) | |
| Frontend | K (Kilo) | P (Peta) |
| | M (Mega) | E (Exa) |
| | G (Giga) | Z (Zetta) |
| | T (Tera) | Y (Yotta) |

2 Time-related multipliers

Since Zabbix version 1.8.2 the following time-related multipliers are available:

- **s** - seconds; when used, works the same as raw value;
- **m** - minutes;
- **h** - hours;

- **d** - days
- **w** - weeks.

These are supported in trigger expression constants and function parameters, as well as in the internal item **zabbix[queue,<from>,<to>]** parameters.

3 Examples

These multipliers allow to write expressions that are easier to understand and maintain, for example the following expressions:

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>120
{host:system.uptime[].last(0)}<86400
{host:system.cpu.load.avg(600)}<10
```

could be changed to:

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>2m
{host:system.uptime.last(0)}<1d
{host:system.cpu.load.avg(10m)}<10
```

**22 Time period specification**

1 The format

Time period has the following format:

`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

| Format | Description |
|---|---|
| **d** | Day of week: 1 - Monday, 2 - Tuesday ,... , 7 - Sunday |
| **hh** | Hours: 00-24 |
| **mm** | Minutes: 00-59 |

**Attention:**
The upper bound of time period specification is not included. E. g. if you specify 09:00-18:00 the last second included in the time period will be 17:59:59. This is true starting from version 1.8.7 for everything except Working time which has always worked this way.

3 Default

Empty time specification equals to 01-07,00:00-24:00

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`

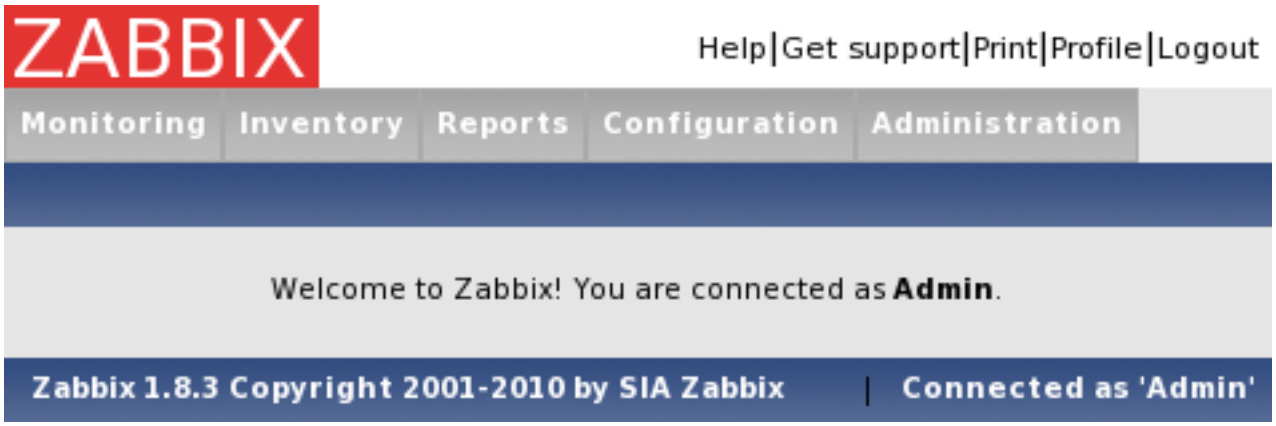
## 5 Quick Start Guide

login add_user email_settings add_host set_up_notifications

**1 Login**

This is ”Welcome to Zabbix” screen. When installed, use user name **Admin** with password **zabbix** to connect as Zabbix superuser.


When logged in, you will see ”Connected as Admin” in the lower right corner of the page and access to *Configuration* and *Administration* areas will be granted:




1.1 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.

IP address of a failed login attempt will be displayed after successful login.

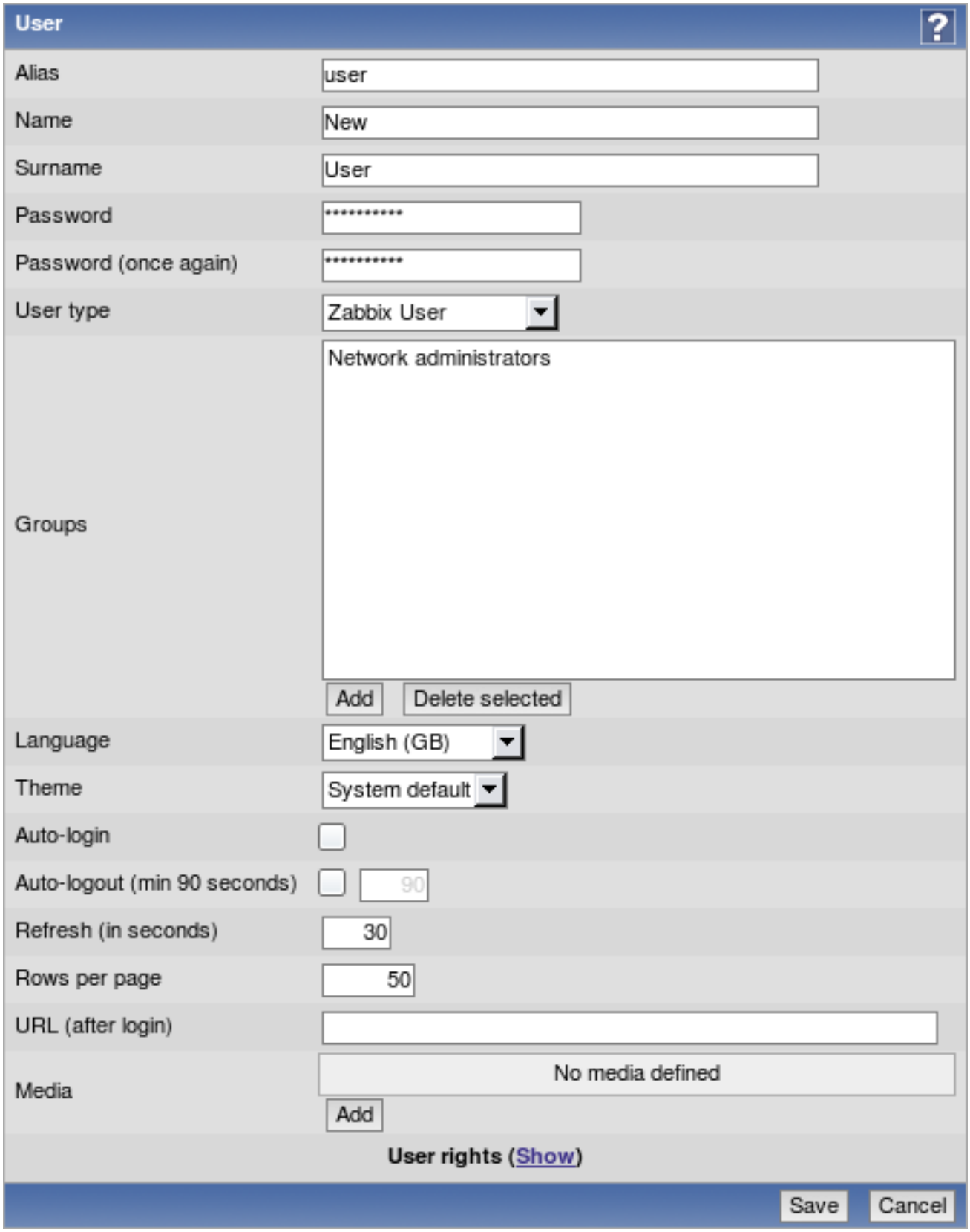
**2 Add user**

After initial installation, Zabbix has only two users defined. User ”Admin” is a Zabbix superuser, which has full permissions. User ”guest” is a special default user. If a user does not log in, the user will be accessing Zabbix with ”guest” permissions. By default, ”guest” has no permissions on Zabbix objects.

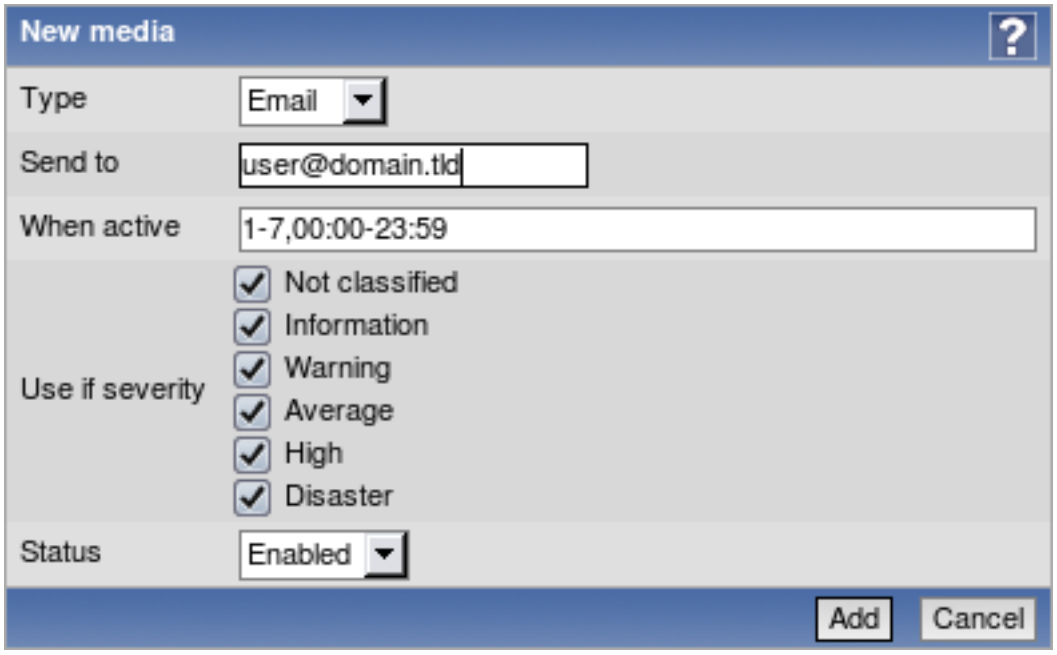
| Alias ▲ | Name | Surname | User type | Groups | Is online? | Login | GUI access | API access | Debug mode | Status |
|---|---|---|---|---|---|---|---|---|---|---|
| Admin | Zabbix | Administrator | Zabbix Super Admin | Zabbix administrators | Yes (Mon, 07 Jun 2010 10:27:18 +0300) | Ok | System default | Disabled | Disabled | Enabled |
| guest | Default | User | Zabbix User | Guests | Yes (Mon, 07 Jun 2010 10:27:00 +0300) | Ok | System default | Disabled | Disabled | Enabled |

To add a new user, navigate to Administration → Users and switch to *Users* in the dropdown, then click ”Create User”.

In the new user form, make sure to add your user to one of the existing groups, for example **Network administrators**.



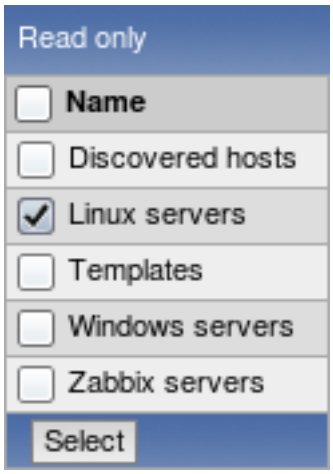

By default, new users have no media (notification methods) defined. To create one, click *Add* in the **Media** section.



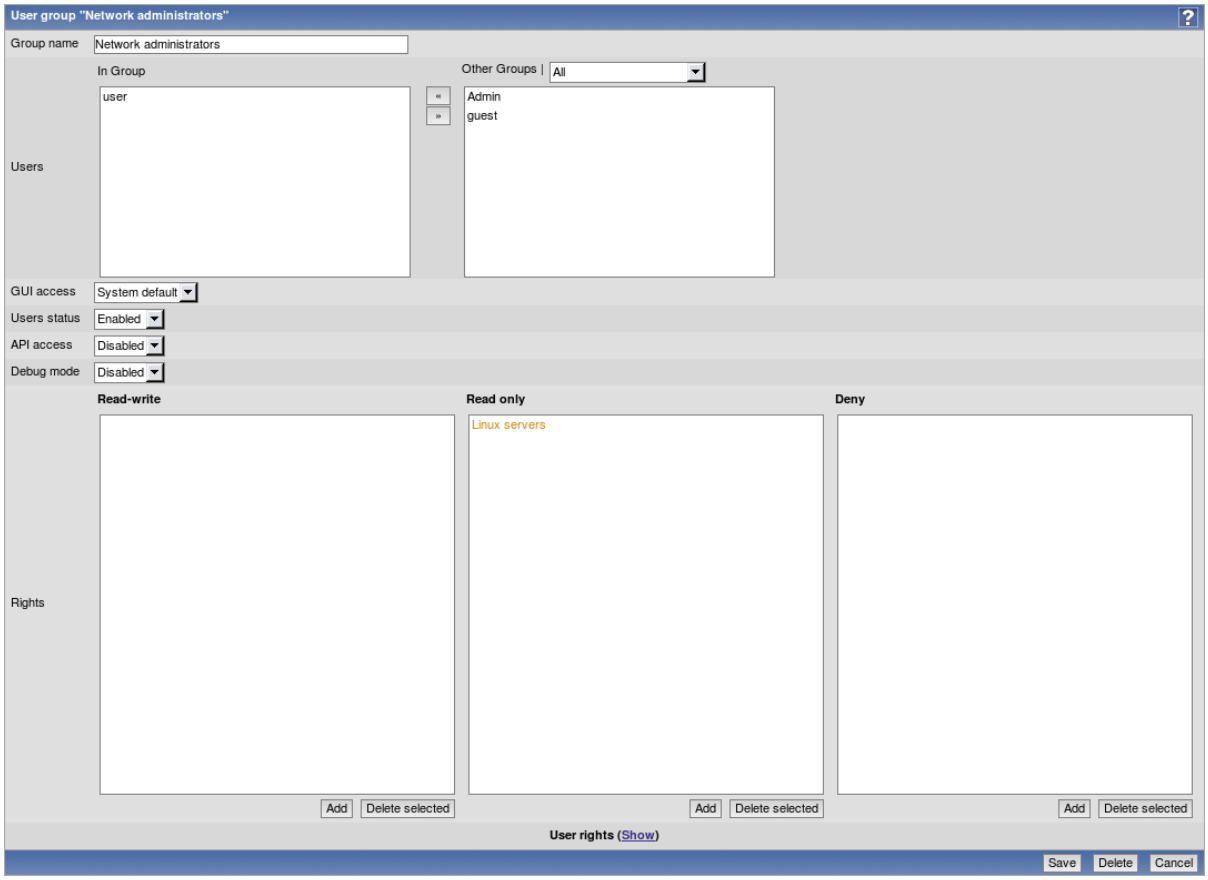

In this popup, enter an e-mail address for the user. You can specify a time period when the medium will be active (see Time period specification page for description of the format), by default a medium is always active. You can also customise severities for which the medium will be active, but leave all of them enabled for now. Click *Add*, then click *Save* in the user properties. The new user appears in the userlist.

| Alias | Name | Surname | User type | Groups | Is online? | Login | GUI access | API access | Debug mode | Status |
|---|---|---|---|---|---|---|---|---|---|---|
| Admin | Zabbix | Administrator | Zabbix Super Admin | Zabbix administrators | Yes (Mon, 07 Jun 2010 10:40:11 +0300) | Ok | System default | Disabled | Disabled | Enabled |
| guest | Default | User | Zabbix User | Guests | Yes (Mon, 07 Jun 2010 10:39:59 +0300) | Ok | System default | Disabled | Disabled | Enabled |
| user | New | User | Zabbix User | Network administrators | No | Ok | System default | Disabled | Disabled | Enabled |

By default, a new user has no permissions. To grant the user rights, click on the group in **Groups** column. This opens the group properties form. This user will have read-only access to Linux servers group, so click on *Add* below the **Read only** listbox.




In this popup, mark the checkbox next to **Linux servers**, then click *Select*.



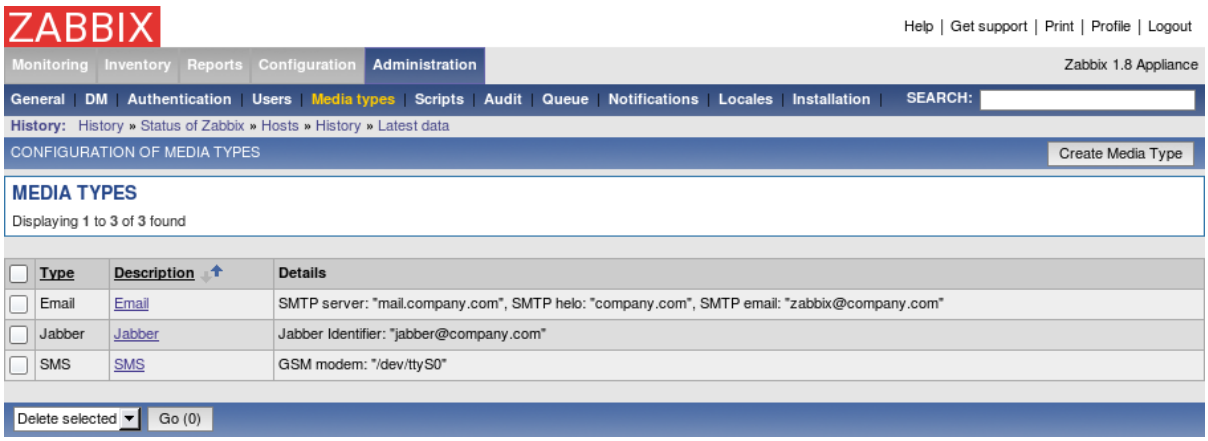

In the user group properties form, click *Save*.

**Note:**
In Zabbix, all access rights are assigned to user groups.

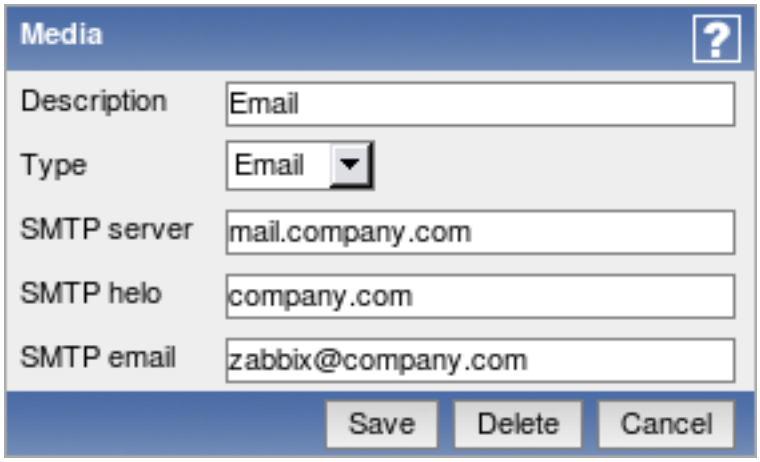
Done! You may try to log in using credentials of the new user.

### 3 Email settings

Initially, Zabbix has several predefined notification delivery methods (media types). Email is one of those. Email configuration can be found under Menu → Administration → Media types.

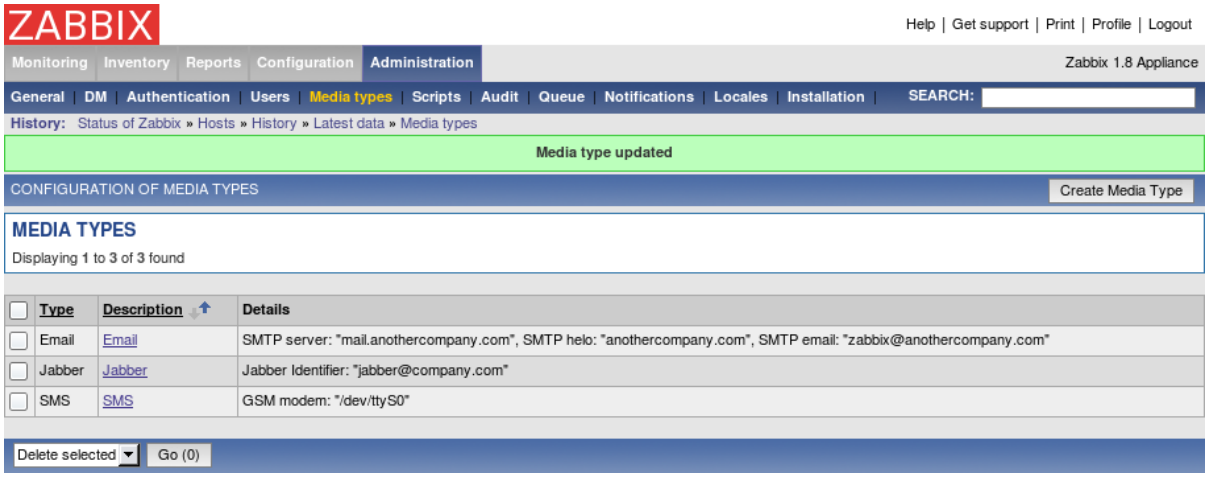



Click on *Email* in the list of pre-defined media types.




Set correct SMTP server, SMTP helo and SMTP email values. Press **Save** when ready.

**Note:**
*SMTP email* is used as the **From** address for outgoing e-mails.




Now you have media type ”Email” defined. A media type must be linked with users, otherwise it will not be used.

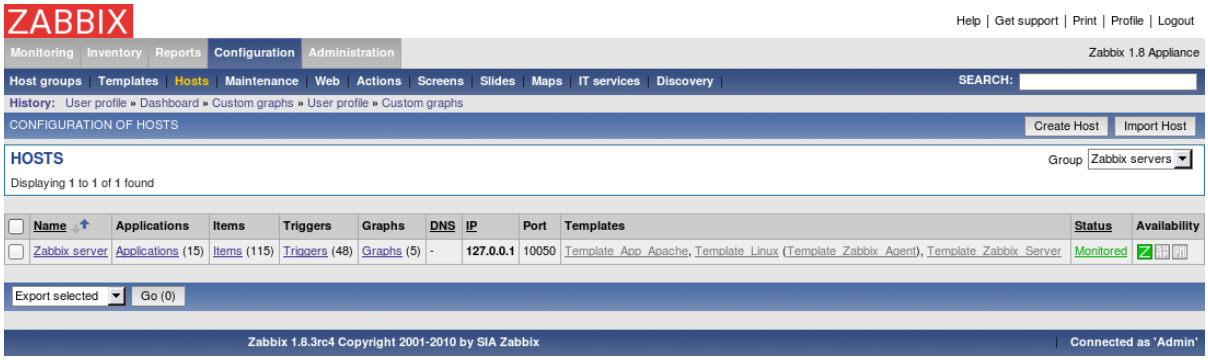
## 4 Monitoring an agent-enabled host

The section provides details about monitoring a host which has Zabbix agent running. You must have the agent installed and configured properly.

### 4.1 Monitoring default Zabbix server

Open Configuration → Hosts to see the list of currently defined hosts. The situation will be different depending on Zabbix version being used.

- If you are using Zabbix up to version 1.8.3, you will see single disabled host, *Zabbix server*.
- If you are using Zabbix appliance version 1.8.3 or later, you will see single enabled host, *Zabbix server*.



If the host is not monitored, click on **Not monitored** in the **Status** column and confirm the popup. That's it, we don't have to do anything else - if agent and server daemons are running properly, the host will be monitored from now on.

4.2 Monitoring a different server

Open Configuration → Hosts to see the list of currently defined hosts. There will be one pre-defined host, but now we want to add another one.

Click on *Create host*. As the minimum, host definition for our purposes should have the following defined:

- Host name;
- Host must belong to at least one hostgroup;
- For passive Zabbix agent monitored hosts IP address should be defined;
- For a quickstart, we will use one of the pre-defined templates as well.

Other options will suit us with their defaults.

**Host name**

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

**Groups**

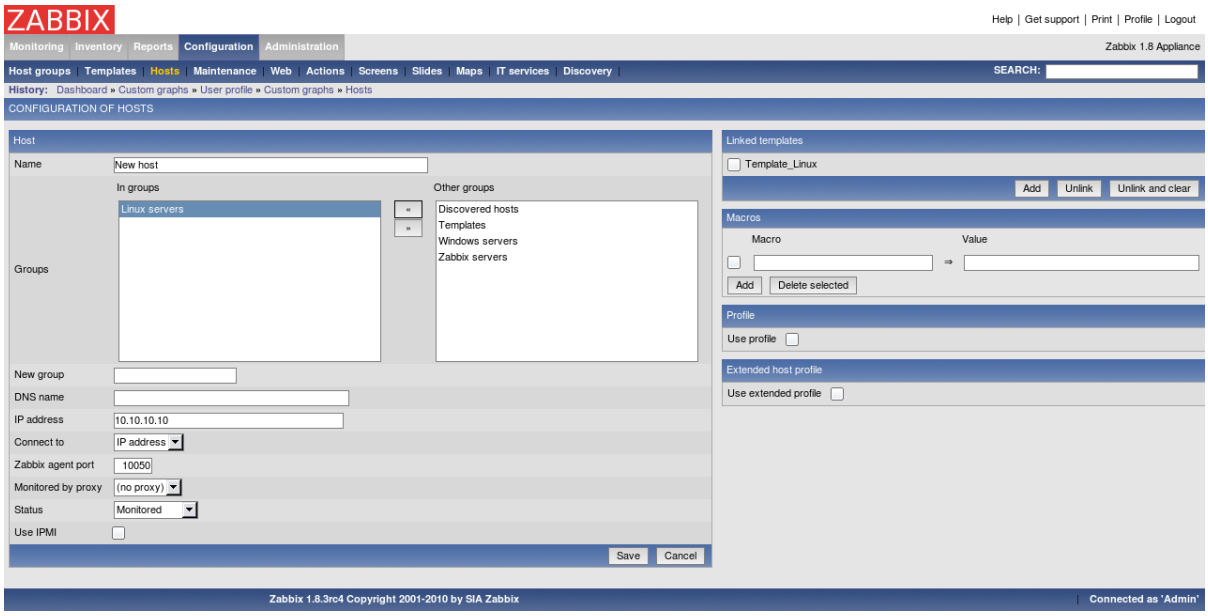
- Host must belong to at least one host group. Move groups from the right hand side box to the left hand side box and the opposite until you are satisfied with the result.

**IP address**

- Enter the IP address of the host. Note that Zabbix agent daemon must have Zabbix server IP address specified in its configuration file **Server** directive.

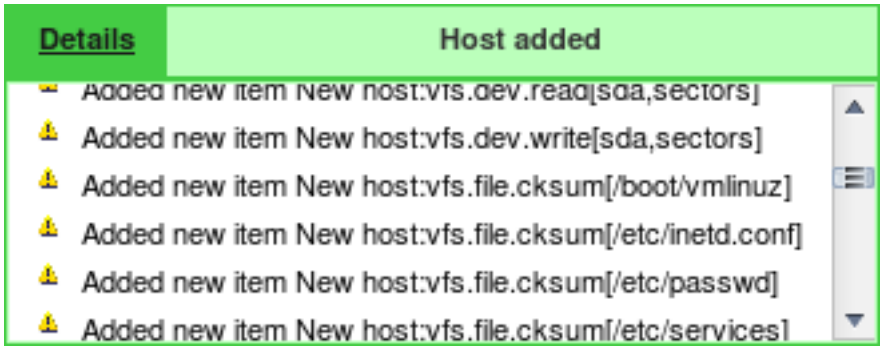
**Linked templates**

- On the right hand side block *Linked templates*, click on the *Add* button, choose *Templates* in the *Group* dropdown, then mark checkbox next to *Template_Linux* entry (assuming the newly added host is running Linux) and click on *Select*.




When done, click *Save*.

The host should be successfully created. Click on *Details* in the upper left corner of the resulting page - that should show you what actually happened.




According to the details, the effect of using a template should be that this new host now has entities from *Template_Linux* - let's verify that. In the *Group* dropdown, choose one of the groups you added your new host to. That should show a high level configuration overview of this host.

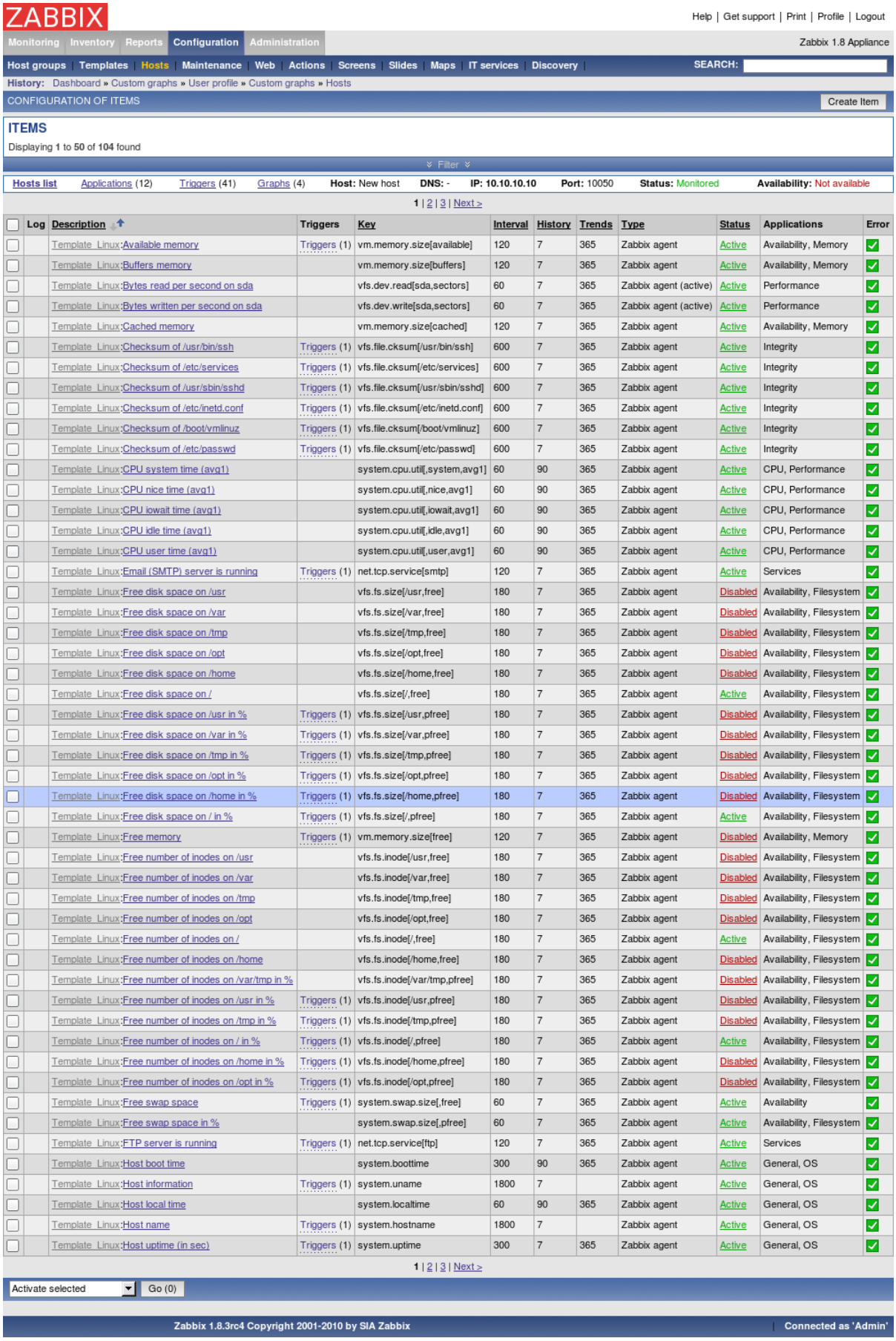
4.3 Verifying current configuration

| Name | Applications | Items | Triggers | Graphs | DNS | IP | Port | Templates | Status | Availability |
|---|---|---|---|---|---|---|---|---|---|---|
| New host | Applications (12) | Items (104) | Triggers (41) | Graphs (4) | - | 10.10.10.10 | 10050 | Template_Linux (Template_Zabbix_Agent) | Monitored | ZBX SNMP JMX IPMI |

In this list we can see that several items, triggers and graphs supposedly have been added to our new host.

**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.

Let's make sure that this host indeed has those items. Click on *Items* next to it.




Looks like items have been added successfully. Note the *Template_Linux* text in gray prefixing them, which indicates which template do the entities come from. What about triggers? Looking above the item list, there's a horizontal strip which allows to easily navigate between different entity categories of a host.

**Note:**
By default, Zabbix entity lists are limited to 50 entries per page. you can modify this in your user profile.

Hosts list | Applications (12) | Triggers (41) | Graphs (4) | **Host:** New host | **DNS:** - | **IP:** 10.10.10.10 | **Port:** 10050 | **Status:** Monitored | **Availability:** Not available

In there, click on *Triggers*.

| Severity | Status | Name | Expression | Error |
|---|---|---|---|---|
| Warning | Enabled | Template_Linux:/etc/inetd.conf has been changed on server {HOSTNAME} | {New host:vfs.file.cksum[/etc/inetd.conf].diff(0)}>0 | |
| Average | Enabled | Template_Linux:/etc/passwd has been changed on server {HOSTNAME} | {New host:vfs.file.cksum[/etc/passwd].diff(0)}>0 | |
| Average | Enabled | Template_Linux:/etc/services has been changed on server {HOSTNAME} | {New host:vfs.file.cksum[/etc/services].diff(0)}>0 | |
| Average | Enabled | Template_Linux:/usr/bin/ssh has been changed on server {HOSTNAME} | {New host:vfs.file.cksum[/usr/bin/ssh].diff(0)}>0 | |
| Average | Enabled | Template_Linux:/usr/sbin/sshd has been changed on server {HOSTNAME} | {New host:vfs.file.cksum[/usr/sbin/sshd].diff(0)}>0 | |
| Warning | Enabled | Template_Linux:/vmlinuz has been changed on server {HOSTNAME} | {New host:vfs.file.cksum[/boot/vmlinuz].diff(0)}>0 | |
| Information | Enabled | Template_Linux:Configured max number of opened files is too low on {HOSTNAME} | {New host:kernel.maxfiles.last(0)}<512 | |
| Information | Enabled | Template_Linux:Configured max number of processes is too low on {HOSTNAME} | {New host:kernel.maxproc.last(0)}<256 | |
| Average | Enabled | Template_Linux:Email (SMTP) server is down on {HOSTNAME} | {New host:net.tcp.service[smtp].last(0)}=0 | |
| Information | Enabled | Template_Linux:Host information was changed on {HOSTNAME} | {New host:system.uname.diff(0)}>0 | |
| Information | Enabled | Template_Linux:Hostname was changed on {HOSTNAME} | {New host:system.hostname.diff(0)}>0 | |
| Average | Enabled | Template_Linux:Lack of available memory on server {HOSTNAME} | {New host:vm.memory.size[available].last(0)}<20M | |

Great - triggers also seem to be in place (the above screenshot only shows part of the output, though). There was also something about graphs - using the host bar above the trigger list navigate to custom graph configuration.

| Name | Width | Height | Graph type |
|---|---|---|---|
| Template_Linux:CPU utilisation | 900 | 200 | Stacked |
| Template_Linux:Disk usage | 900 | 200 | Stacked |
| Template_Linux:Network utilisation | 900 | 200 | Normal |
| Template_Linux:Used diskspace | 400 | 200 | Pie |

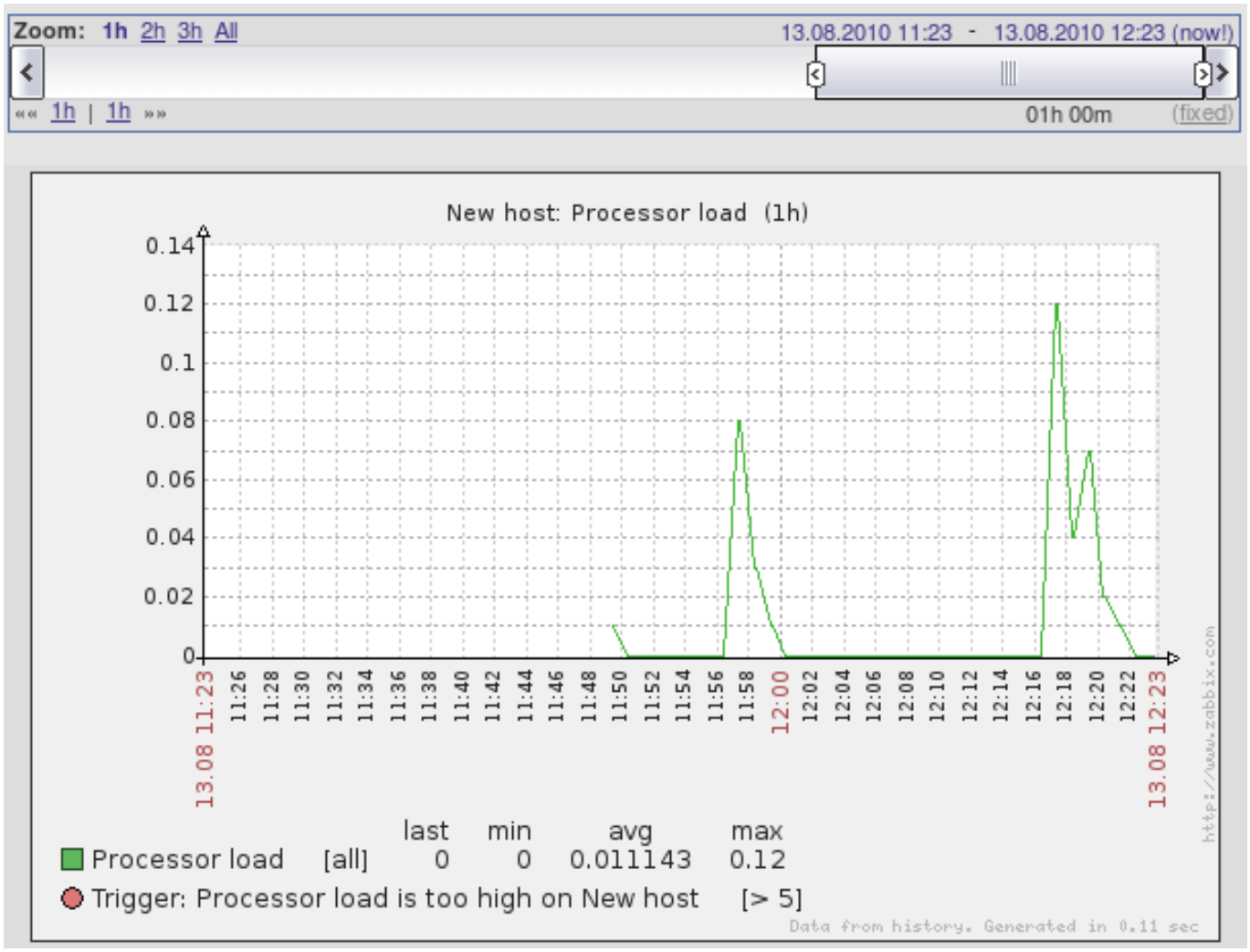
Here we can see that some templated custom graphs are available as well.

Now it is time to see what information is available. Go to Menu→Latest data and expand some category in there.

| Description | Last check | Last value | Change | History |
|---|---|---|---|---|
| **Availability** (14 Items) | | | | |
| **CPU** (6 Items) | | | | |
| CPU idle time (avg1) | 13 Aug 2010 12:19:34 | 95.91 | -2.43 | Graph |
| CPU iowait time (avg1) | 13 Aug 2010 12:19:35 | 0.764526 | +0.575458 | Graph |
| CPU nice time (avg1) | 13 Aug 2010 12:19:36 | 0 | - | Graph |
| CPU system time (avg1) | 13 Aug 2010 12:19:37 | 2.36 | +1.26 | Graph |
| CPU user time (avg1) | 13 Aug 2010 12:19:38 | 0.678771 | +0.179632 | Graph |
| Processor load | 13 Aug 2010 12:19:33 | 0.07 | +0.03 | Graph |
| **Filesystem** (7 Items) | | | | |
| **General** (5 Items) | | | | |
| **Integrity** (5 Items) | | | | |
| **Memory** (5 Items) | | | | |

The values are being gathered and displayed along with change information, if any.

In Zabbix, for all numeric items a graph can be obtained without any configuration at all - these graphs are generated on runtime. To view such a graph, click on *Graph* link next to any item.




You can change the currently displayed time period using the controls above the graph.

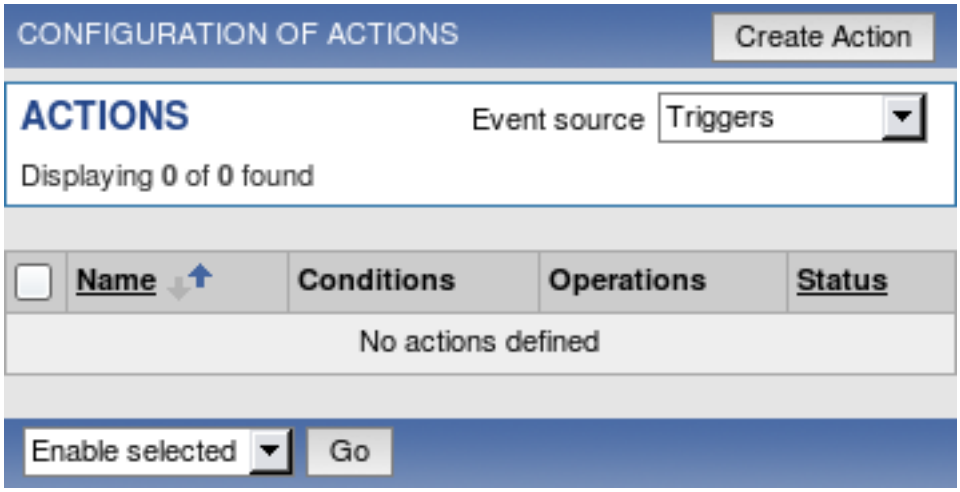
Feel free to explore other areas that display monitoring information, including:

- Monitoring → Graphs for custom graphs;
- Monitoring → Triggers for a list of currently active problems;
- Monitoring → Dashboard for a high level overview;
- Monitoring → Maps for network maps;
- Monitoring → Screens for compound pages showing several elements at once.

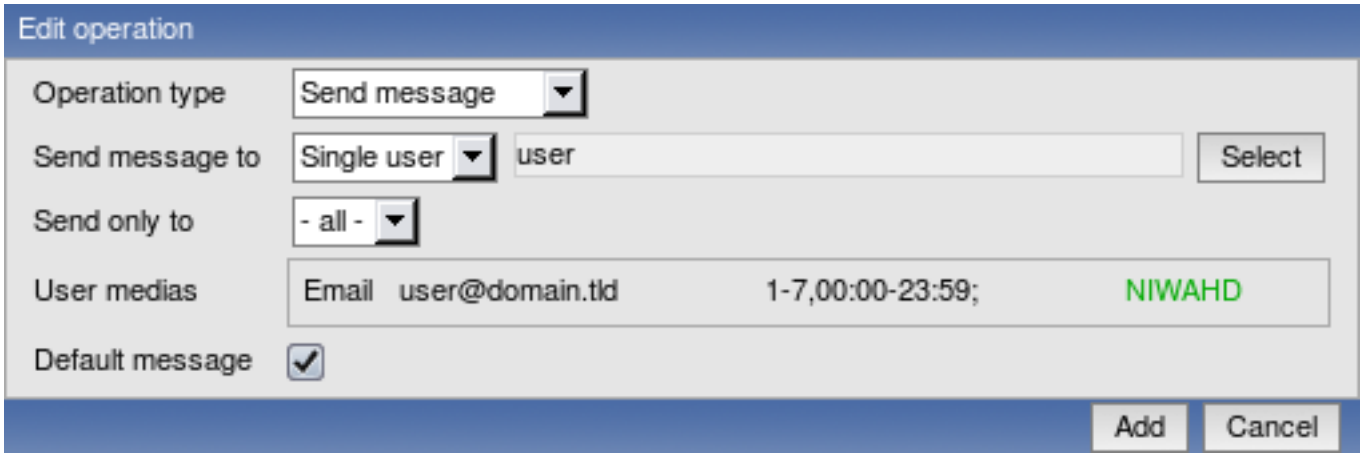
After having the basic monitoring in place, we might want to actually notify on situation changes, which we'll set up in the next section.

**5 Set up notifications**

We have a host or several hosts monitored. We can see simple and custom graphs, as well as data for individual items. We also have problem conditions, called triggers, set up, and they are changing from OK to PROBLEM state and back as situation changes. While we can look at the data to determine the current status, it is not feasible to do so all the time - which means we will want to set up notifications. To do this, open Configuration → Actions.

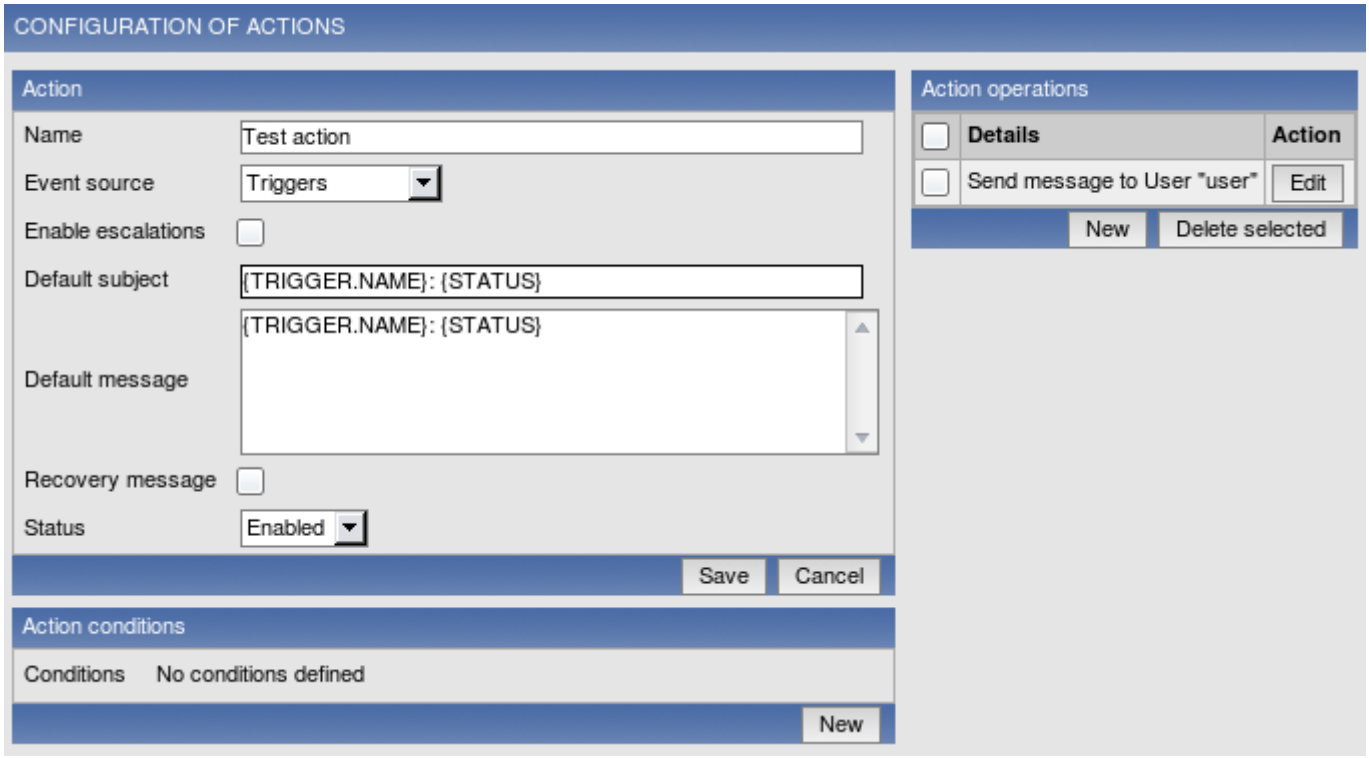



By default, there are no actions configured. To create one, click *Create Action*. In the upcoming form, enter a name for the action. In the most simple case, if we don't add any conditions, action will be used upon any trigger change from OK to PROBLEM and vice versa. We still should define what the action should do - and that is done in the *Action operations* block. Click on *New* in that block, which opens new operation configuration form. Here, choose *Single user* in the *Send message to* dropdown, then click on *Select*. In the upcoming popup, choose the user we created before.




Notice how the e-mail address we specified for that user will be used here.

Macros (or variables) **{TRIGGER.NAME}** and **{STATUS}**, currently visible in the *Default subject* and *Default message* fields, will be replaced with trigger name and trigger status, respectively. Trigger status will be either PROBLEM or OK. Click *Add* in the *Edit operation* block.




We are done with the simple action configuration, so click *Save* in the *Action* block.

Congratulations - we are done with the simple setup of monitoring some host and sending out notifications based on problem condition definitions.

**Note:**
If the notifications don't work, make sure user you created has at least read permissions on the host which generated the event, as discussed in the "Add user" step. Additionally, you can check out action log by going to Administration → Audit, and choosing *Actions* in the dropdown, located in the upper right corner.

## 6 XML Import and Export

goals overview data_export data_import map_export_import screen_export_import

#### 1 Goals

Zabbix Import/Export functionality is created to make possible effective exchange of various configuration entities.

Data is exported in XML format which is easy to read and modify.

Use cases:

- Sharing of templates or network maps

Zabbix users may share configuration parameters.

- Integration with third-party tools

Universal XML format makes integration and data import/export possible with third party tools and applications.

**Note:**
Exporting and importing network maps is supported since Zabbix version 1.8.2.

#### 2 Overview

Currently two main categories of configuration are supported for export - hosts and their associated data, and network maps.

2.1 Host import/export

Zabbix host import/export processes the following data:

- Hosts and their linkage to templates;
- Templates;
- Applications;
- Items;
- Triggers;
- Custom graphs;
- User macros.

2.2 Map import/export

Zabbix map import/export supports the following elements since version 1.8.2:

- Full map configuration;
- All map elements, including images, triggers, hosts, host groups and maps;
- All connectors with associated data, including labels and status indicators.

Additionally, since 1.8.3 used images (icons and background images) are exported as well.

2.3 Screen import/export

Zabbix screen import/export supports all screen elements.

**3 Host export**

For Zabbix versions up to 1.8.3, host and template export is available at *Configuration* → *Export/Import*. Starting with 1.8.3, import and export controls are available on corresponding configuration pages (*Configuration* → *Hosts* and *Configuration* → *Templates*).

3.1 Since Zabbix 1.8.3

3.1.1 Step 1

Navigate either to *Configuration* → *Hosts* or *Configuration* → *Templates*, depending on which ones you want to export. Mark checkboxes next to elements to be exported.

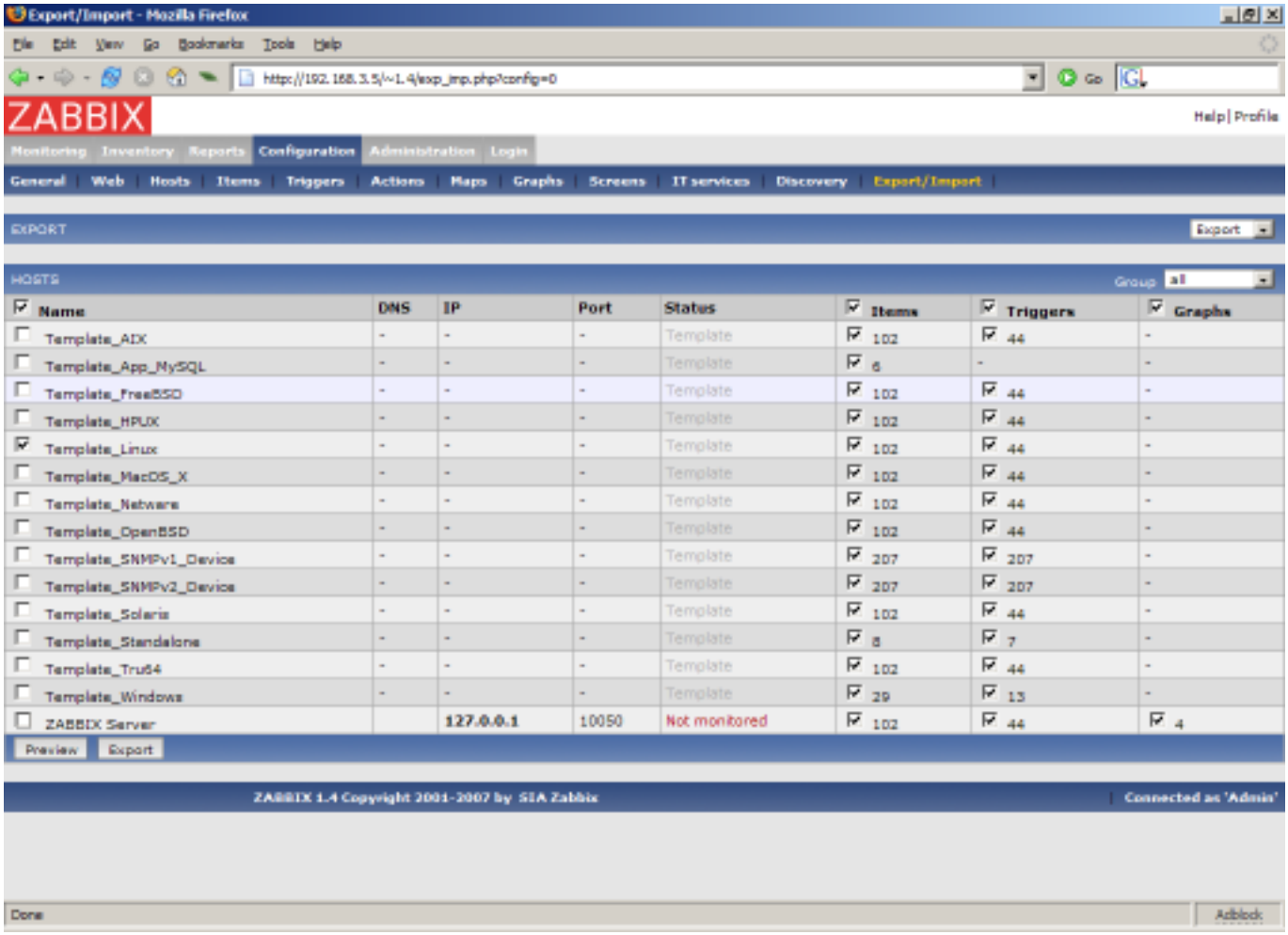
3.1.2 Step 2

Make sure that **Export selected** is chosen in the activity dropdown below host or template list, then click **Go** and save the file.

3.2 Up to Zabbix 1.8.3

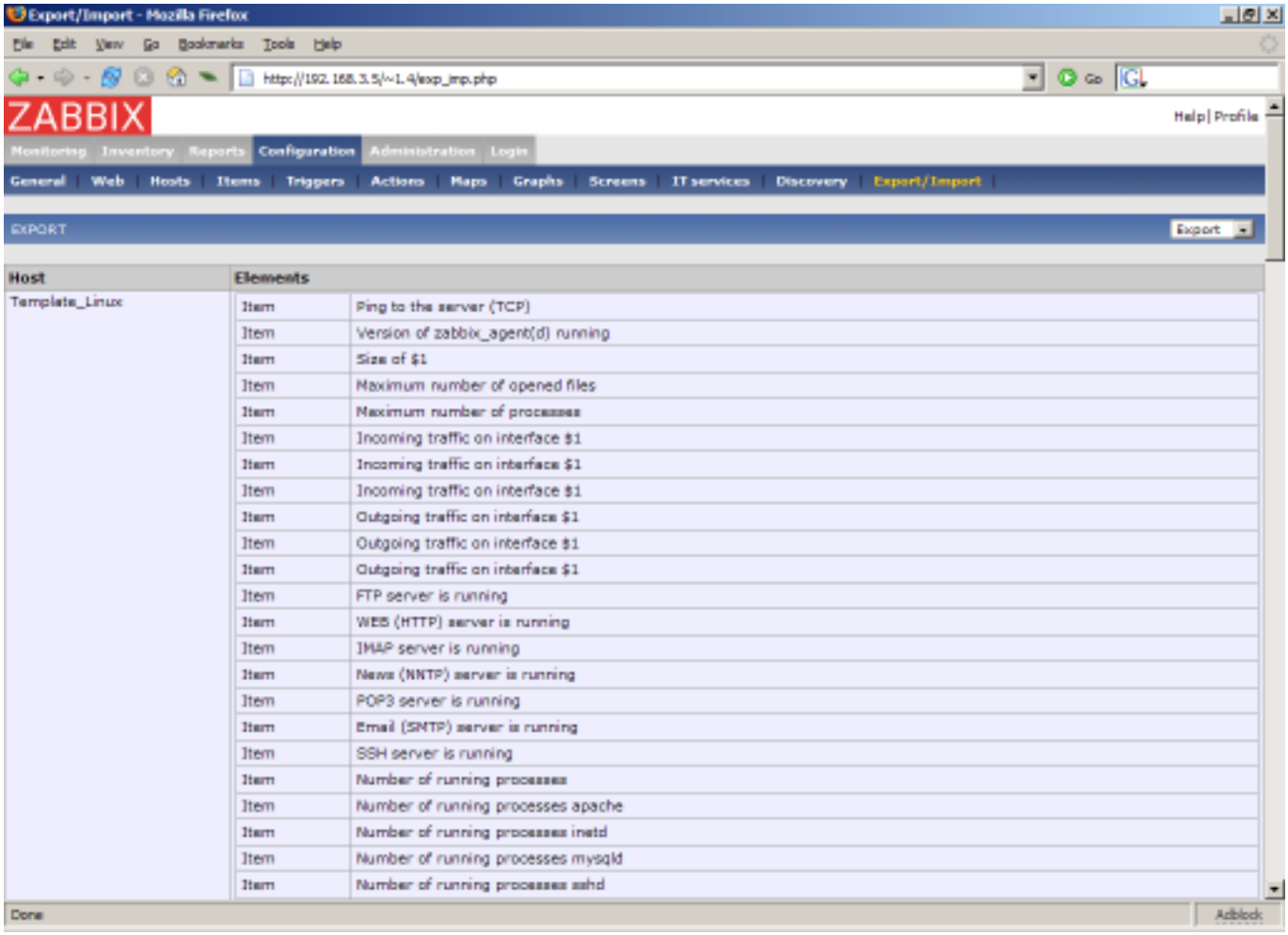
Step 1

Select elements for export




We selected host ”Template_Linux” and all its items and triggers.

Press button ”Preview” to see list of elements to be exported:




Step 2

Export data

Press button "Export" to export selected elements to a local XML file with default name *zabbix_export.xml*. The file has the following format (one element of each type is shown):

```
<?xml version="1.0"?>
<zabbix_export version="1.0" date="11.05.07" time="11.11">
  <hosts>
      <host name="ZABBIX Server">
          <useip>1</useip>
          <ip>127.0.0.1</ip>
          <port>10050</port>
          <status>1</status>
          <groups>
          </groups>
          <items>
              <item type="0" key="agent.ping" value_type="3">
                  <description>Ping to the server (TCP)</description>
                  <delay>30</delay>
                  <history>7</history>
                  <trends>365</trends>
                  <snmp_port>161</snmp_port>
                  <valuemap>Service state</valuemap>
                  <applications>
                      <application>General</application>
                  </applications>
              </item>
                  ....
          </items>
          <triggers>
              <trigger>
```

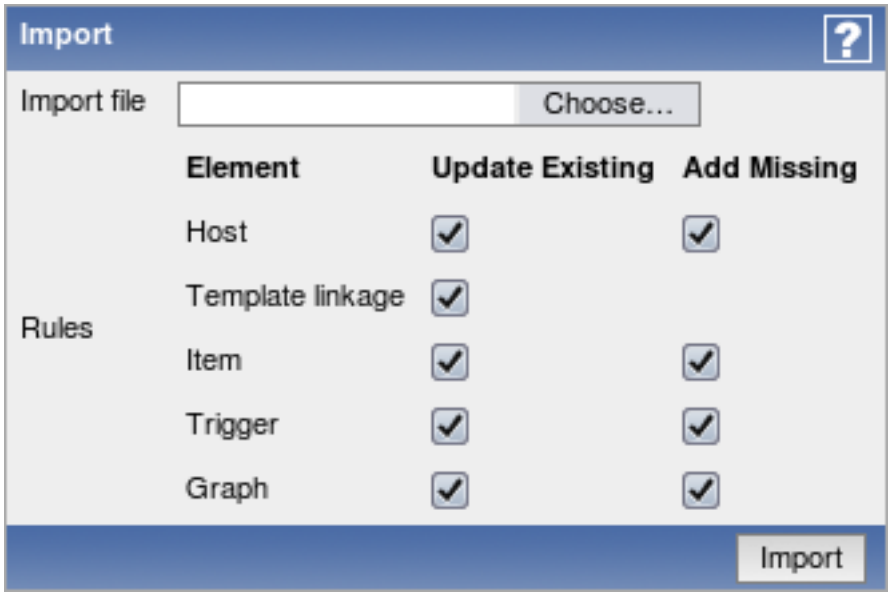
```
                <description>Version of zabbix_agent(d) was changed on {HOSTNAME}</description>
                <expression>{{HOSTNAME}:agent.version.diff(0)}>0</expression>
                <priority>3</priority>
            </trigger>
                ....
        <graphs>
            <graph name="CPU Loads" width="900" height="200">
                <show_work_period>1</show_work_period>
                <show_triggers>1</show_triggers>
                <yaxismin>0.0000</yaxismin>
                <yaxismax>100.0000</yaxismax>
                <graph_elements>
                    <graph_element item="{HOSTNAME}:system.cpu.load[,avg15]">
                        <color>990000</color>
                        <yaxisside>1</yaxisside>
                        <calc_fnc>2</calc_fnc>
                        <periods_cnt>5</periods_cnt>
                    </graph_element>
                    <graph_element item="{HOSTNAME}:system.cpu.load[,avg1]">
                        <color>009900</color>
                        <yaxisside>1</yaxisside>
                        <calc_fnc>2</calc_fnc>
                        <periods_cnt>5</periods_cnt>
                    </graph_element>
                    <graph_element item="{HOSTNAME}:system.cpu.load[,avg5]">
                        <color>999900</color>
                        <yaxisside>1</yaxisside>
                        <calc_fnc>2</calc_fnc>
                        <periods_cnt>5</periods_cnt>
                    </graph_element>
                </graph_elements>
            </graph>
                ....
        </graphs>
    </host>
                ....
  </hosts>
</zabbix_export>
```

**4 Host import**

For Zabbix versions up to 1.8.3, host and template import is available at *Configuration → Export/Import*. Starting with 1.8.3, import and export controls are available on corresponding configuration pages (*Configuration → Hosts* and *Configuration → Templates*).

Step 1

Configure settings for data import and press “Import”.




Pay attention to the following parameters of the item:

| PARAMETER | Description |
|---|---|
| **Import file** | File name of XML file. |
| **Rules** | **Element** defines element of XML file.<br>If parameter **Update** is set for **Existing** element, then the import will update it with data taken from the file. Otherwise it will not update it.<br>If parameter **Add** is set for **Missing** element, then the import will add new element with data taken from the file. Otherwise it will not add it. |

**Attention:**
Note that Zabbix versions 1.8.x place triggers before items in the export and such data can not be imported in Zabbix 1.6.x. If such a path is desired, items should be moved in front of the triggers.

**5 Map export and import**

**Note:**
Map export and import is available since Zabbix version 1.8.2.

Map export and import controls can be found under Configuration → Maps menu, where all configured maps are displayed.

5.1 Map exporting

In left bottom corner a selection box is available with two options: ”Export selected” and ”Delete selected”. To export maps:

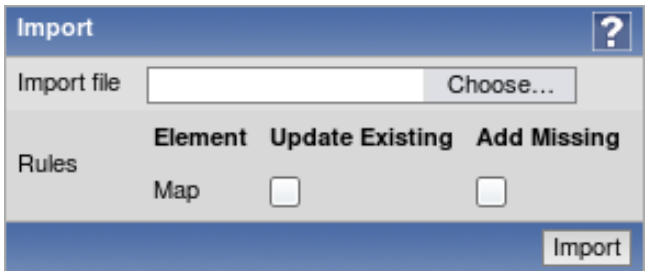
1. Mark boxes next to maps you wish to export;
2. Select ”Export selected” if it’s not selected already;
3. Press button ”Go”;
4. Select file where Zabbix should store XML data with exported maps.

5.2 Map importing

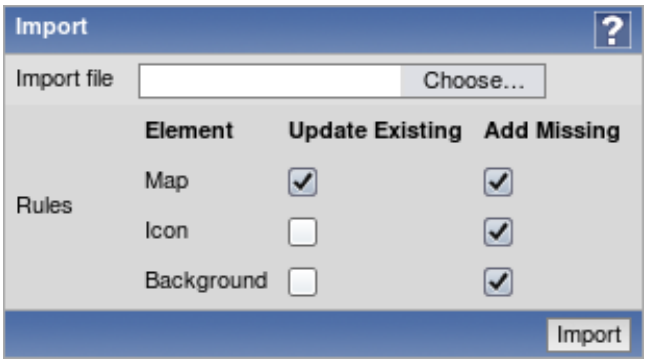
Importing maps is as easy as exporting them. On the top right corner near ”Create Map” button, you will find new button - ”Import Map”.


To import maps:

1. Press ”Import Map” button. You will get to a screen similar to what you see when importing hosts in Configuration→ Export/Import (Import) menu;
2. Press on ”Choose file” button to select XML file containing exported Zabbix maps;
3. Check box under ”Update existing” if you need to update (overwrite) existing maps;
4. Check box under ”Add missing” if you need to create a new map if it’s missing;
5. Press import to send needed data to Zabbix frontend;
6. Wait till page reloads. It can take some time if you have lots of maps to import or lots of hosts, triggers etc. Zabbix frontend will inform you about import success or failure.




Map import dialogue




Map import dialogue for Zabbix super admin, showing image importing options (available since version 1.8.3)

Click on ”Details” link on the left hand side to see more information about what was done by import, or why it failed to import.

5.3 What is exported?

Only map structure is exported. That means all map settings are exported, all contained elements with their settings are exported, so are the map links and map link status indicators.

5.4 What is not exported?

Any maps, hostgroups, hosts, triggers, images or any others elements related to the exported map are not exported. Thus if at least one of the elements map refers to is missing, import will fail.

For example, if map refers to a specific trigger on a specific node, but this trigger has been deleted in the meantime, import would fail with an error message:

```
Cannot find trigger "our_node5:Zabbix server:Disk is full" used in exported map "Small map".
```

5.5 Export format

An example empty map export with background image from a distributed setup node:

```
<sysmap>
  <selements>
  </selements>
  <links>
  </links>
  <name>TEST</name>
  <width>800</width>
  <height>600</height>
  <backgroundid>
    <node>aly_trunk</node>
    <name>Map BG</name>
  </backgroundid>
  <label_type>2</label_type>
  <label_location>0</label_location>
```

```
  <highlight>1</highlight>
  <expandproblem>1</expandproblem>
</sysmap>
```

5.5.1 Map elements

Let's take XML excerpt of some Zabbix map element as an example:

```
<selement>
  <selementid>100100000000372</selementid>
  <elementid>
    <node>noden1</node>
    <host>LocalHost</host>
    <description>DOUBLE</description>
    <expression>{TimeHost:system.localtime[local].last(0)}=0 & {TimeHost:system.localtime[local].last
  </elementid>
  <elementtype>2</elementtype>
  <iconid_off>
    <node>noden1</node>
    <name>Hub</name>
  </iconid_off>
  <label>New Element</label>
  <label_location>-1</label_location>
  <x>231</x>
  <y>122</y>
</selement>
```

- **<selement>** is the opening tag for an element (shorthand of "System map element");
- **<selementid>** is a unique element id, used for map link references;
- **<elementid>** refers to the actual Zabbix entity that is represented on the map (map/hostgroup/host etc.;
- **<node>** tag will be present if the exported map comes from a distributed setup, skipped otherwise;
- **<elementtype>** describes what type of element info is stored in **<elementid>** node;

**Note:**
When importing an XML, **selementid** values don't have to match any values in the existing dataset - they are only used to determine map link connections.

5.5.2 Element types and storage

**elementtype** tag in map export can be one of the following:

| Value | Type |
|---|---|
| 0 | Host |
| 1 | Map |
| 2 | Trigger |
| 3 | Host group |
| 4 | Image |

- **Host reference**

*DM (distributed monitoring) setup*

```
<node>noden1</node>
<host>LocalHost</host>
```

*Single server setup*

```
<host>LocalHost</host>
```

Hosts are referred to by host name.

- **Map reference**

*DM setup*

```
<node>noden1</node>
<name>Local map</name>
```

*Single server setup*

```
<name>Local map</name>
```

Maps are referred to by map name.

- **Trigger reference**

Triggers are described in a more complex way:

*DM setup*

```
<node>noden1</node>
<host>LocalHost</host>
<description>Lack of free memory on server {HOSTNAME}</description>
<expression>{LocalHost:vm.memory.size[free].last(0)}<10000</expression>
```

*Single server setup*

```
<host>LocalHost</host>
<description>Lack of free memory on server {HOSTNAME}</description>
<expression>{LocalHost:vm.memory.size[free].last(0)}<10000</expression>
```

Trigger is referred to by host name, trigger description and trigger expression.

- **Host group reference**

*DM setup*

```
<node>noden1</node>
<name>Local Host Group</name>
```

*Single server setup*

```
<name>Local Host Group</name>
```

Host groups are referred to by host group name.

- **Image reference**

> **Note:**
> For images <elementid> node can be skipped.

Nodes **<iconid_off>**, **<iconid_on>**, **<iconid_unknown>**, **<iconid_maintenance>** and **<iconid_disabled>** describes what icons should be used for the map element according to its status.

For default icon, **<iconid_off>** is used.

Inside icon block, image itself is specified: *DM setup*

```
<node>noden1</node>
<name>Local Image</name>
```

*Single server setup*

```
<name>Local Image</name>
```

To use default icon for any state, node for that state should be skipped in the **<selement>** block.

5.5.3 Element labels

- **<label>** describes map elements labels. Macros can be used in labels.
- **<label_location>** is used for positioning element's label:

| Value | Type |
|---|---|
| -1 | use map default |
| 0 | bottom |
| 1 | left |
| 2 | right |
| 3 | top |

5.5.4 Element positioning

**<x>** and **<y>** nodes are used for positioning element on the map by x and y coordinates.

5.5.5 Map links

Example:

```
<link>
  <selementid1>100100000000399</selementid1>
  <selementid2>100100000000402</selementid2>
  <drawtype>0</drawtype>
  <color>00AA00</color>
  <linktriggers>
  </linktriggers>
</link>
```

- **<selemetid1>** and **<selementid2>** nodes are used to specify map elements that link connects.
- **<drawtype>** defines default link style:

| Value | Style |
|---|---|
| 0 | line |
| 2 | bold line |
| 3 | dot |
| 4 | dashed line |

- **<color>** specifies what the default link colour is;
- **<linktriggers>** contains information about link status indicators.

Example:

```
<linktrigger>
  <triggerid>
    <node>aly_trunk</node>
    <host>Symmetra PX40 Clone2</host>
    <description>APC: Input Current (PHASE L3)</description>
    <expression>{Symmetra PX40 Clone2:upsPhaseInputCurrent.L3.last(0)}<15 | {Symmetra PX40 Clone2
  </triggerid>
  <drawtype>0</drawtype>
  <color>0</color>
</linktrigger>
```

- **<triggerid>** describes trigger used for indicating link status. Linked trigger referenced the same as map element trigger;
- **<drawtype>** and **<color>** are used to indicate how link should be drawn on the map if this trigger has the highest severity from all the active triggers that are attached to this link.

5.5.6 Images

**Note:**
Image import/export is supported since Zabbix version 1.8.3.

It is possible to export and import used images alongside maps. If exported map is using any images, they are stored in the resulting XML file. An example of how an exported image might look like:

```
<images>
  <image>
    <name>Server (small)</name>
    <imagetype>1</imagetype>
    <encodedImage>iVBORw0KGgoAA...ErkJggg==</encodedImage>
  </image>
</images>
```

**Warning:**
Value for the <encodedImage> tag is truncated in the above example.

Used tags:

- <images> - root element for images
- <image> - individual image element
- <name> - image name, unique

- <imagetype> - image type, where 1 => icon, 2 => background
- <encodedImage> - base64 encoded image

When importing, missing images can be added and existing images can be overwritten by marking appropriate checkboxes. Image importing is only available to users of Zabbix Super Admin type.

**Warning:**
Warning: if replacing an existing image, it will affect all maps that are using this image.

It is possible to import images only by unchecking both map checkboxes.

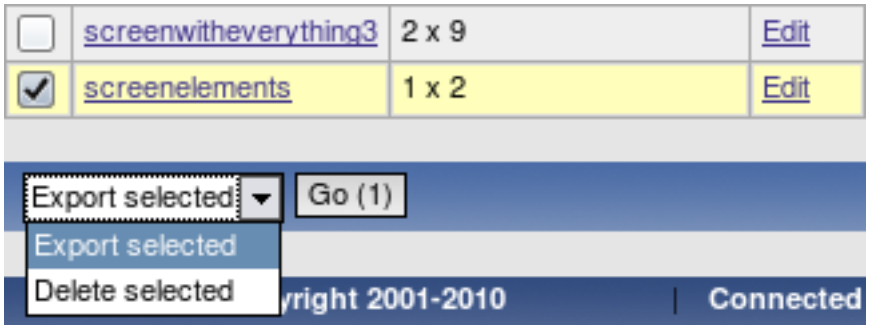
**6 Screen export and import**

**Note:**
Screen export and import is available since Zabbix version 1.8.2.

Screen export and import controls can be found under Configuration → Screens menu, where all configured screens are displayed.

6.1 Screen exporting

In left bottom corner a selection box is available with two options: ”Export selected” and ”Delete selected”.

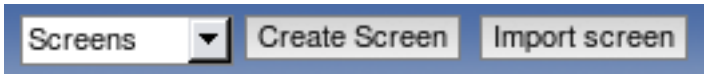



To export screens:

1. Mark checkboxes next to screens you wish to export;
2. Select ”Export selected” if it’s not selected already;
3. Press button ”Go”;
4. Select file where Zabbix should store XML data with exported screens.

6.2 Screen importing

Importing screens is as easy as exporting them. On the top right corner near ”Create Screen” button, you will find new button - ”Import Screen”.




To import screens:

1. Press ”Import Screen” button. File selection along with options to control import is shown;
2. Click the button to select XML file containing exported Zabbix screens;
3. Check box under ”Update existing” if you need to update (overwrite) existing screens;
4. Check box under ”Add missing” if you need to create new screen if it’s missing;
5. Press import to send needed data to the Zabbix frontend;
6. Wait till page reloads. It can take some time if you have lots of screens to import or lots their elements. Zabbix frontend will inform you about import success or failure. Click on ”Details” link in the left upper corner to see more detailed information about what has been done by import, or why it failed to import.

|<| |<| |-|

6.3 Exported data

Only screen structure is exported. That includes all screen settings are and all screen elements along with their configuration.

6.4 Not exported data

Anything included in the screen itself (like a host, hostgroup or any other data) is not exported.

When importing a screen, if any of the referenced elements is missing, import will fail, for example, with:

```
Cannot find trigger "child_node5: ZBXHost:DOUBLE CHECK" used in exported screen "Link Screen"
```

6.5 XML format - screen definition

Screen export consists of screen definition itself and any additional elements.

```
<screens>
  <screen>
    <name>Screen name</name>
    <hsize>1</hsize>
    <vsize>2</vsize>
    <screenitems>
      <screenitem></screenitem>
      ...
      <screenitem></screenitem>
    </screenitems>
  </screen>
</screens>
```

- **name** - screen name;
- **hsize** - rows;
- **vsize** - columns;
- **screenitem** - individual screen items, described below.

6.6 XML format - screen elements

Available screen elements and their IDs.

| Resource type | Description |
|---|---|
| 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 | #Status of hostgroup triggers |
| 15 | #System status |
| 16 | #Status of host triggers |

6.7 XML format - available element tags

**Bold text** - mandatory tag for all elements;
Normal text - tag available for all elements;
*Italic text* - tag optionally available for some elements (see below for details).

- **<resourcetype>** - identifies element type, as per the table above;
- <*resourceid*> - identifies resource, if applicable; depends of resource type;
- <*width*> - element's width in pixels, if applicable;
- <*height*> - element's height in pixels, if applicable;
- **<x>** - element location on screen table by X axis (cell of the upper left corner);
- **<y>** - element location on screen table by Y axis (cell of the upper left corner);
- <colspan> - if higher than 1, sets count of columns to merge (to the right);
- <rowspan> - if higher than 1, sets count of rows to merge (down);
- <*elements*> - amount of rows to show, if applicable;
- <valign> - vertical align: **0** - middle, **1** - top, **2** - bottom;

- <halign> - horizontal align: **0** - centre, **1** - left, **2** - right;
- <*style*> - meaning depends on resource type;
- <*dynamic*> - allows to apply the element to different hostgroups and/or hosts, if applicable.

If <resourceid> refers to an object by name, it can have subtags. If data is exported from a distributed setup installation, node will always be identified by name:

```
<node>Zabbix node</node>
```

For example, #Simple graph **<resourceid>** entry from a non-distributed setup would look like this:

```
<resourceid>
  <host>Zabbix server</host>
  <key_>system.cpu.load</key_>
</resourceid>
```

In a distributed setup, it becomes:

```
<resourceid>
  <node>Zabbix node</node>
  <host>Zabbix server</host>
  <key_>system.cpu.load</key_>
</resourceid>
```

Individual object references are listed at each element.

6.8 XML format - individual screen element details, A-Z

Each individual element must have mandatory tags from the previous section and may have tags that are available for all elements. If there are additional tags available for the specific element, they are listed here.

6.8.1 Clock

Resource type 7. Additional tags:

- <width>;
- <height>;
- <style> - Local time (0), Server time (1).

6.8.2 Data overview

Resource type 10. Additional tags:

- <resourceid> - Host group (by name);
- <width>;
- <height>.

Available <*resourceid*> contents:

```
<name>Linux servers</name>
```

6.8.3 Graph

Resource type 0. Additional tags:

- <resourceid> - Graph (by name);
- <dynamic>.

Available <*resourceid*> contents:

```
<host>Zabbix host</host>
<name>Graph name</name>
```

6.8.4 History of actions

Resource type 12. Additional tags:

- <elements> - amount of rows to show.

6.8.5 History of events

Resource type 13. Additional tags:

- <elements> - amount of rows to show.

6.8.6 Hosts info

Resource type 4. Additional tags:

- <resourceid> - Host group (by name).

Available <*resourceid*> contents:

```
<name>Linux servers</name>
```

6.8.7 Map

Resource type 2. Additional tags:

- <resourceid> - Zabbix map (by name).

Available <*resourceid*> contents:

```
<name>City map</name>
```

6.8.8 Plain text

Resource type 3. Additional tags:

- <resourceid> - Item (by key);
- <elements> - number of rows to show;
- <style> - if set, HTML code will rendered for in item data that contains strings;
- <dynamic>.

Available <*resourceid*> contents:

```
<host>Zabbix server</host>
<key_>system.cpu.load</key_>
```

6.8.9 Screen

Resource type 8. Additional tags:

- <resourceid> - Screen (by name);

Available <*resourceid*> contents:

```
<name>Application servers screen</name>
```

6.8.10 Server info

Resource type 6. No additional tags available.

6.8.11 Simple graph

Resource type 1. Additional tags:

- <resourceid> - Item (by key);
- <dynamic>.

Available <*resourceid*> contents:

```
<host>Zabbix server</host>
<key_>system.cpu.load</key_>
```

6.8.12 Status of host triggers

Resource type 16. Additional tags:

- <resourceid> - Host (by name);
- <elements> - number of rows to show.

Available <*resourceid*> contents:

```
<host>aleksei_host</host>
```

6.8.13 Status of hostgroup triggers

Resource type 14. Additional tags:

- <resourceid> - Host group (by name);
- <elements> - number of rows to show.

Available <*resourceid*> contents:

```
<name>aaa</name>
```

6.8.14 System status

Resource type 15. No additional tags available.

6.8.15 Triggers info

Resource type 5. Additional tags:

- <resourceid> - Host group (by name);

Available <*resourceid*> contents:

<name>aaa</name>

6.8.16 Triggers overview

Resource type 9. Additional tags:

- <resourceid> - Host group (by name);

Available <*resourceid*> contents:

<name>aaa</name>

6.8.17 URL

Resource type 11. Additional tags:

- <url> - fully qualified or relative URL.

6.9 XML export example

The following is a simple screen (2x2), exported to XML. It contains one custom graph in upper left cell (spanning two columns), one simple graph in the lower left cell and trigger status element, filtered for a hostgroup, in the lower right cell. Notice the encoding of **&** as **&**.

```
<?xml version="1.0" encoding="UTF-8"?>
<screens>
  <screen>
    <name>Excellent screen</name>
    <hsize>2</hsize>
    <vsize>2</vsize>
    <screenitems>
      <screenitem>
        <resourcetype>0</resourcetype>
        <resourceid>
          <host>Zabbix server</host>
          <name>CPU Load & traffic</name>
        </resourceid>
        <width>1000</width>
        <height>100</height>
        <x>0</x>
        <y>0</y>
        <colspan>2</colspan>
        <rowspan>0</rowspan>
        <elements>0</elements>
        <valign>0</valign>
        <halign>0</halign>
        <style>0</style>
        <dynamic>0</dynamic>
      </screenitem>
      <screenitem>
        <resourcetype>1</resourcetype>
        <resourceid>
          <host>Zabbix server</host>
          <key_>zabbix[uptime]</key_>
        </resourceid>
        <width>500</width>
        <height>90</height>
        <x>0</x>
        <y>1</y>
```

```
        <colspan>0</colspan>
        <rowspan>0</rowspan>
        <elements>0</elements>
        <valign>0</valign>
        <halign>0</halign>
        <style>0</style>
        <dynamic>0</dynamic>
      </screenitem>
      <screenitem>
        <resourcetype>14</resourcetype>
        <resourceid>
          <name>Linux servers</name>
        </resourceid>
        <width>500</width>
        <height>100</height>
        <x>1</x>
        <y>1</y>
        <colspan>0</colspan>
        <rowspan>0</rowspan>
        <elements>25</elements>
        <valign>0</valign>
        <halign>0</halign>
        <style>0</style>
        <dynamic>0</dynamic>
      </screenitem>
    </screenitems>
  </screen>
</screens>
```

## 7 Tutorials

This section contains step-by-step instructions for most common tasks.

extending_agent log_files remote_actions windows_services

### 1 Extending Zabbix Agents

This tutorial provides step-by-step instructions how to extend functionality of Zabbix agent.

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 this command to agent's configuration file.

Add the command to zabbix_agentd.conf:

```
UserParameter=mysql.questions,mysqladmin -uroot status|cut -f4 -d":"|cut -f1 -d"S"
```

**mysql.questions** is an unique identifier. It can be any string, for example, queries.

Test this parameter by using zabbix_get utility.

Step 3

Restart Zabbix agent.

Agent will reload configuration file.

Step 4

Add new item for monitoring.

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.

### 2 Monitoring of log files

This tutorial provides step-by-step instructions how to setup monitoring of log files. It is assumed that a host is configured already in Zabbix frontend.

Step 1

Configure Zabbix agent.

Follow standard instructions in order to install and configure agent on monitored host. Make sure that parameter **Hostname** matches host name of the host configured in Zabbix frontend.

Also make sure that parameter **DisableActive** is not set in zabbix_agentd.conf

Step 2

Add a new item for monitoring of a log file.

Pay attention to the following parameters of the item:

| PARAMETER | Description |
|---|---|
| Type | Must be set to 'Zabbix agent (active)'. |
| Key | Must be set to 'log[file<,regexp>]'. For example: log[/var/log/syslog], log[/var/log/syslog,error]. Make sure that the file has read permissions for user 'zabbix' otherwise the item status will be set to 'unsupported'. Zabbix agent will filter entries of log file by the regexp if present. |
| Type of information | Must be set to 'log'. |
| Update interval (in sec) | The parameter defines how often Zabbix agent will check for any changes in the log file. Normally must be set to 1 second in order to get new records as soon as possible. |

### 3 Remote commands

This tutorial provides step-by-step instructions on how to setup remote execution of pre-defined commands in case on an event. It is assumed that Zabbix is configured and operational.

Step 1

On Zabbix agent, enable remote commands. In *zabbix_agentd.conf* make sure that parameter **EnableRemoteCommands** is set to **1** and uncommented. Restart agent daemon if changing this parameter.

Step 2

Configure new action by going to Configuration → Actions and in the *New action* block choose operation type **Remote command**.

Pay attention to the following parameters of the action:

| PARAMETER | Description |
|---|---|
| Action type | Must be set to 'Remote command'. |
| Remote command | Each line must contain an command for remote execution. For example: host:sudo /etc/init.d/apache restart. Remote command may contain macros! |

**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**.

Syntax of remote commands:

| REMOTE COMMAND | Description |
|---|---|
| {HOSTNAME}:<command> | Command 'command' will be executed on the host where the event happened. |
| <host>:<command> | Command 'command' will be executed on host 'host'. |
| <group>#<command> | Command 'command' will be executed on all hosts of host group 'group'. |

**Note:**
Zabbix agent executes commands in background. Zabbix does not check if a command has been executed successfully.

**Attention:**
Remote commands in Zabbix < 1.4 are limited to 44 characters, in Zabbix >= 1.4 they are limited to 255 characters.

Syntax of IPMI remote commands:

| REMOTE COMMAND | Description |
|---|---|
| {HOSTNAME}:IPMI <ipmi control> [value] | The syntax is for execution of IPMI command on the host where the event happened. Supported values: "on", "off" or number (1, by default). |
| <host>:IPMI <ipmi control> [value] | The syntax is for execution of IPMI command on a single host. |
| <group>#IPMI <ipmi control> [value] | The syntax is for execution of IPMI command for all hosts of a host group. |

Access permissions

Make sure that user 'zabbix' 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*.

**Note:**
On recent systems it might be required to set **Defaults visiblepw** in */etc/sudoers*.

Example 1

Restart of Windows on certain condition.

In order to automatically restart Windows in case of a problem detected by Zabbix, define the following actions:

| PARAMETER | Description |
|---|---|
| Action type | ’Remote command’ |
| Remote command | host:c:\windows\system32\shutdown.exe -r -f<br>Replace ’host’ with Zabbix hostname of Windows server. |

Example 2

Restart the host by using IPMI control.

| PARAMETER | Description |
|---|---|
| Action type | ’Remote command’ |
| Remote command | {HOSTNAME}:IPMI reset on |

Example 3

Power off the host by using IPMI control.

| PARAMETER | Description |
|---|---|
| Action type | ’Remote command’ |
| Remote command | {HOSTNAME}:IPMI power off |

#### 4 Monitoring of Windows Services

This tutorial provides step-by-step instructions how to setup monitoring of Windows services. It is assumed that ZABBIX server and ZABBIX agent are configured and operational.

Step 1

Get service name

You can get that name by going to the services mmc and bring up the properties of the service you want to monitor it’s up/down status. In the General tab you should see a field called Service name. The value that follows that you put in the brackets above. For example, if I wanted to monitor the ”workstation” service then my service would be **lanmanworkstation**.

Step 2

Add item for monitoring of the service

Add item with a key service_state[lanmanworkstation], value type Integer, value mapping Windows service state.

## 9 WEB Monitoring

goals overview web_scenario web_step real_life_scenario

#### 1 Goals

Zabbix WEB Monitoring support is developed with the following goals:

- Performance monitoring of WEB applications
- Availability monitoring of WEB applications
- Support of HTTP and HTTPS
- Support of complex scenarios consisting of many steps (HTTP requests)

## 2 Overview

Zabbix provides effective and very flexible WEB monitoring functionality. The module periodically executes WEB scenarios and keeps collected data in the database. The data is automatically used for graphs, triggers and notifications.

The following information is collected per each step of WEB scenario:

- Response time
- Download speed per second
- Response code

Zabbix also checks if a retrieved HTML page contains a pre-defined string.

Zabbix WEB monitoring supports both HTTP and HTTPS.

When running a web scenario, Zabbix always follows redirects.

> **Note:**
> To use HTTP proxy, set environment variable **http_proxy** for Zabbix server user. For example, //http_proxy=http:%%//%%proxy_ip:proxy_port//.

## 3 WEB Scenario

Scenario is set of HTTP requests (steps), which will be periodically executed by Zabbix server. Normally a scenario is defined for one particular part of functionality of a WEB application. Scenarios are very convenient way of monitoring user experience. WEB Scenario is linked to a host application for grouping. WEB Scenario is periodically executed and consists of one or more Steps. All cookies are preserved during execution of a single scenario.

Example 1

Monitoring of Zabbix GUI

If we want to monitor availability and performance of Zabbix GUI, we have to login, check how quickly Overview and Status of Triggers screens work and then logout.

The scenario may have the following steps:

1. Login
2. Go to Overview screen
3. Go to Status of Triggers screen
4. Logout

If a step cannot be performed, execution of scenario fails.

| Parameter | Description |
|---|---|
| **Application** | WEB scenario will be linked to this application. The application must exist.<br>For example: *Zabbix server* |
| **Name** | Name of the WEB scenario.<br>The name will appear in Monitoring → Web<br>For example: Zabbix GUI |
| **Update interval** | How often this scenario will be executed, in seconds.<br>For example: 60 |
| **Agent** | Zabbix will pretend to be the selected browser. Useful for monitoring of web sites which generate different content for different web browsers.<br>For example: Opera 9.02 on Linux |
| **Status** | **Active**: active scenario, it will be executed<br>**Disabled**: disabled scenario, it will NOT be executed |

| Parameter | Description |
|---|---|
| **Variables** | List of macros to be used in configuration of the steps.<br>**Syntax**:<br>{macro}=value<br>The macro {macro} will be replaced by "variable" in step's URL and POST variables.<br>For example:<br>{user}=guest<br>{password}=guest<br>*Note*: Variables are not URL-encoded. |
| **Steps** | Steps of the scenario. |

As soon as a scenario is created, Zabbix automatically adds the following items for monitoring and links them to the selected application. Actual scenario name will be used instead of "Scenario".

| Item | Description |
|---|---|
| **Download speed for scenario 'Scenario'** | This item will collect information about download speed (bytes per second) of the whole scenario, i.e. average for all steps.<br>Item key: web.test.in[Scenario„bps]<br>Type: float |
| **Failed step of scenario 'Scenario'** | This item keeps number of failed step of the scenario. If all steps are executed successfully, 0 is returned.<br>Item key: web.test.fail[Scenario]<br>Type: integer |

**Note:**
Web monitoring items are added with 30 day history retention and 90 day trend retention periods.

These items can be used to create triggers and define notification conditions.

Example 1

Trigger "WEB scenario failed"

The trigger expression can be defined as:

```
{host: web.test.fail[Scenario].last(0)}#0
```

Do not forget to replace the *Scenario* with real name of your scenario.

Example 2

Trigger "WEB application is slow"

The trigger expression can be defined as:

```
{host: web.test.in[Scenario,,bps].last(0)}<10000
```

Do not forget to replace the *Scenario* with real name of your scenario.

#### 4 WEB Step

Step is basically a HTTP request. Steps are executed in a pre-defined order.

| Parameter | Description |
|---|---|
| **Name** | Name of the step.<br>For example: Login |
| **URL** | URL<br>For example: www.zabbix.com |

| Parameter | Description |
|---|---|
| **Post** | HTTP POST variables, if any.<br>For example:<br>id=2345&userid={user}<br>If {user} is defined as a macro of the WEB scenario, it will be replaced by its value when the step is executed.<br>The information will be sent as is, variables are not URL-encoded.. |
| **Timeout** | Do not spend more than Timeout seconds for execution of the step. 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.<br>For example: 15 |
| **Required** | The string (given as POSIX extended regular expression) must exist in retrieved content.<br>Otherwise this step fails. If empty, any content will be accepted.<br>For example: Homepage of Zabbix |
| **Status codes** | List of HTTP status codes to be considered as success. If retrieved status code is not in the list, this step fails.<br>If empty, any status code is accepted.<br>For example: 200,210 |

As soon as a step is created, Zabbix automatically adds the following items for monitoring and links them to the selected application. Actual scenario and step names will be used instead of ”Scenario” and ”Step” respectively.

| Item | Description |
|---|---|
| **Download speed for step ’Step’ of scenario ’Scenario’** | This item will collect information about download speed (bytes per second) of the step.<br>Item key: web.test.in[Scenario,Step,bps]<br>Type: **float** |
| **Response time for step ’Step’ of scenario ’Scenario’** | This item will collect information about response time of the step in seconds. Response time is counted from the beginning of the request until all information has been transferred.<br>Item key: web.test.time[Scenario,Step,resp]<br>Type: **float** |
| **Response code for step ’Step’ of scenario ’Scenario’** | This item will collect response codes of the step.<br>Item key: web.test.rspcode[Scenario,Step]<br>Type: **integer** |

**Note:**
Web monitoring items are added with 30 day history retention and 90 day trend retention periods.

These items can be used to create triggers and define notification conditions.

Example 1

Trigger ”Zabbix GUI login is too slow”

The trigger expression can be defined as:

```
{zabbix: web.test.time[ZABBIX GUI,Login,resp].last(0)}>3
```

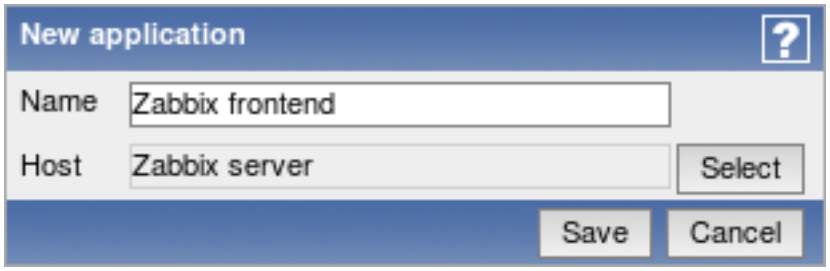
**5 Real life scenario**

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. First we must log in with our user name and password.

Step 1

Add a new host application.

Go to Configuration → Hosts, then click on Applications next to the host you want to use for web monitoring. In the application section click on *Create application*.



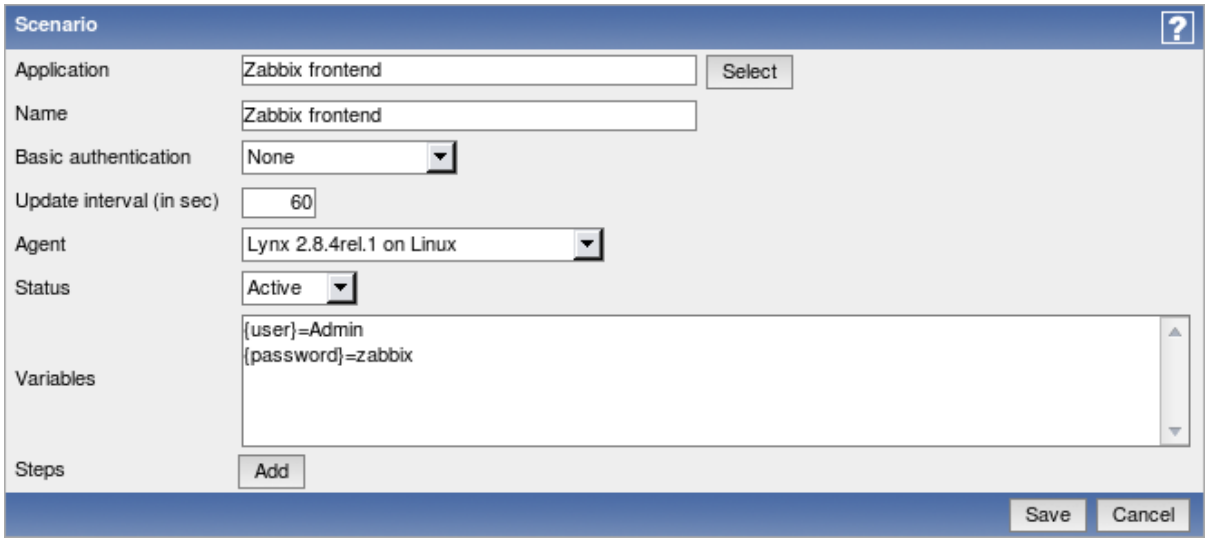

This step is not required if you already have a suitable application. You may also want to create a host if one does not exist.

Step 2

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 → Web, select the host in the dropdown, then click on *Create scenario*.




In the new scenario form, click on *Select* next to the Application field to choose the application we just created.

Note that we also create two macros, {user} and {password}.

Step 3

Define steps for the scenario.

Click on *Add* button in the *Steps* section 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 ”SIA Zabbix”.


When done configuring the step, click *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:

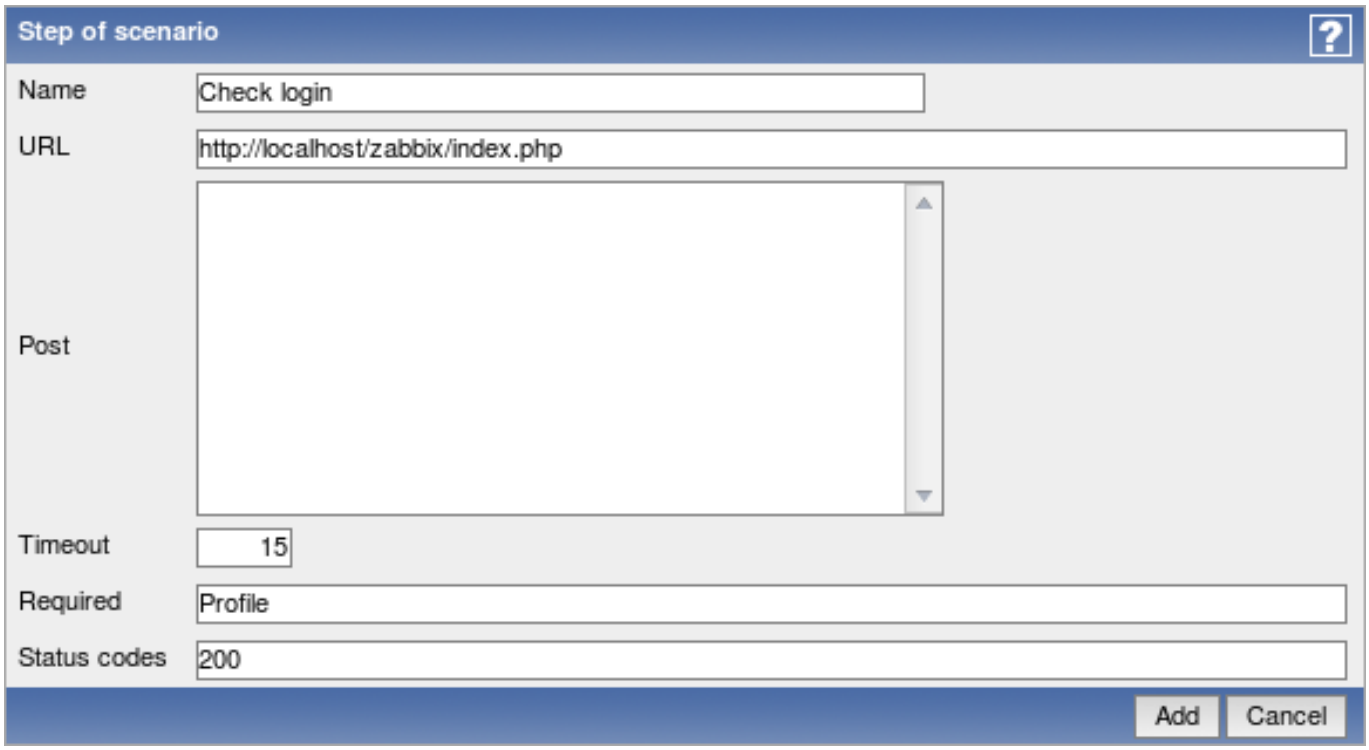
```
name=Admin&password=zabbix&enter=Enter
```

If using the macros as in this example, login string becomes:

```
name={user}&password={password}&enter=Enter
```

*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.

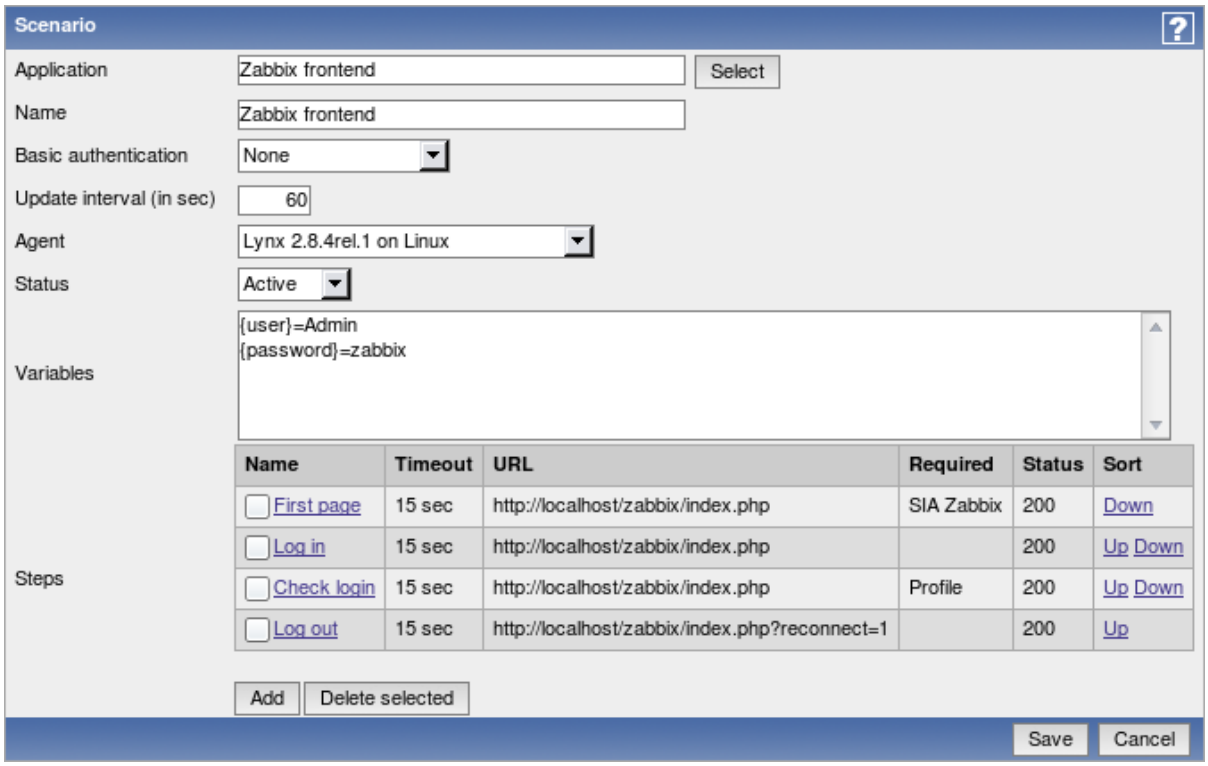

*Complete configuration of steps*

A complete configuration of web scenario steps should look like this:

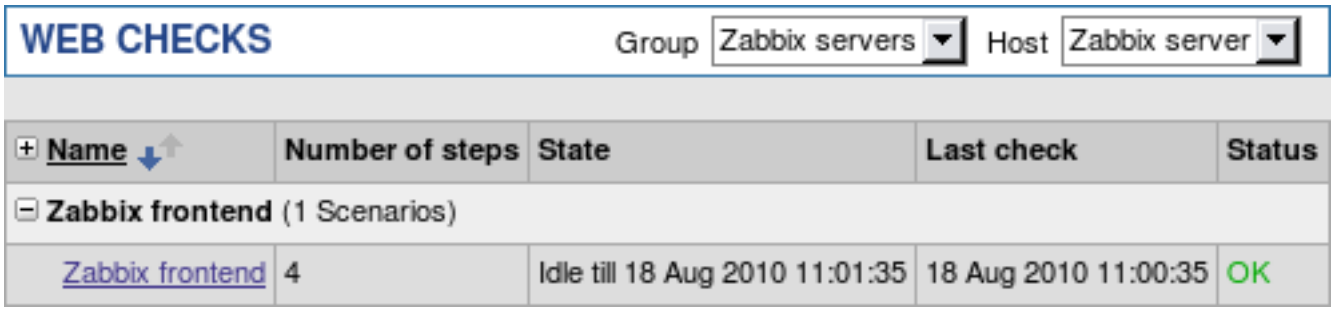
| Name | Timeout | URL | Required | Status | Sort |
|---|---|---|---|---|---|
| First page | 15 sec | http://localhost/zabbix/index.php | SIA Zabbix | 200 | Down |
| Log in | 15 sec | http://localhost/zabbix/index.php | | 200 | Up Down |
| Profile | 15 sec | http://localhost/zabbix/index.php | Profile | 200 | Up Down |
| Log out | 15 sec | http://localhost/zabbix/index.php?reconnect=1 | | 200 | Up |

Step 4

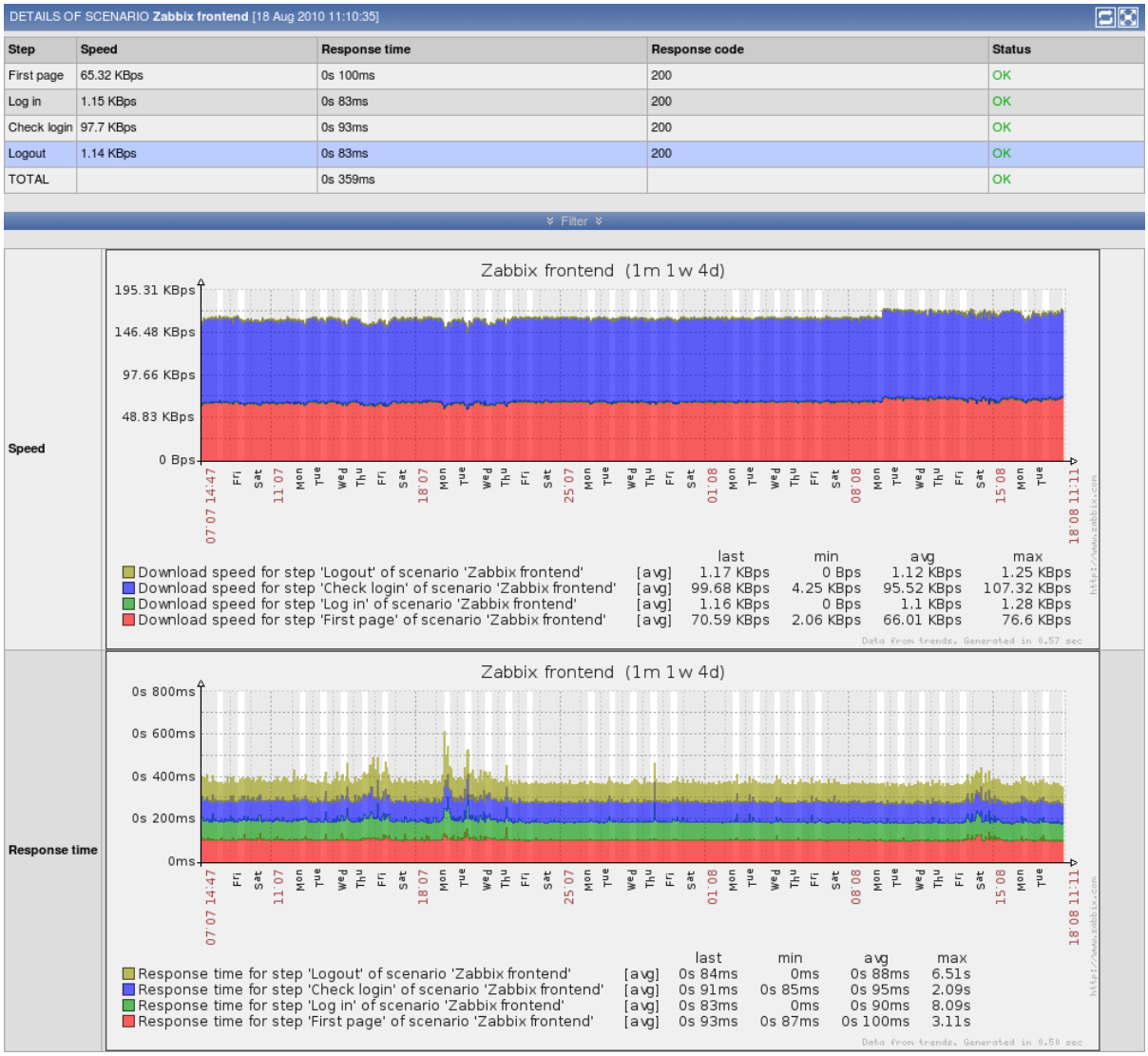
Save the finished web monitoring scenario.




The list of applications and linked scenarios will appear in Monitoring → Web:




Click on the scenario name to see more detailed statistics:




## 10 Log File Monitoring

overview how_it_works

### 1 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.

### 2 How it works

Monitoring of log files requires Zabbix Agent running on a host. An item used for monitoring of a log file must have type *Zabbix Agent (Active)*, its value type must be *Log* and key set to *log[file,<pattern>,<encoding>,<max lines>]* or *logrt[path to log file with filename format,<pattern>,<encoding>,<max lines>]*.

For example:

```
log["/home/user/file.log","pattern_to_match","UTF-8",100]
or
logrt["/home/user/filelog_.*_[0-9]{1,3}","pattern_to_match","UTF-8",100]
```

The last one will collect data from files such ”filelog_abc_1” or ”filelog__001”.

Important notes:

- The server and agent keep a trace of the monitored log's size and last modification time (for logrt) in two counters.
- 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 the last modification time (the time counter) are stored in the Zabbix database and are sent to the agent, to make sure it starts reading the log file from this point.
- Whenever the log file becomes smaller than the log size counter known by the agent, the counter is reset to zero and the agent starts reading the log file from the beginning taking the time counter into account.
- All files matching the filename format in the provided directory are analyzed every cycle the agent tries to get the next line from the log (for logrt).
- If there are several matching files with the same last modification time in the directory, then the agent will read lexicographically the smallest one.
- Zabbix Agent processes new records of a log file once per *Update interval* seconds.
- Zabbix Agent does not send more than **maxlines** of a log file per second. The limit prevents overloading of network and CPU resources and overrides the default value provided for **MaxLinesPerSecond** parameter in the configuration file of the agent.
- Special note for "\" path separators: if file_format is "file\.log", then there should not be directory "file", since it is not possible to unambiguously define whether "." is escaped or is the first symbol of the file name.

## 11 Discovery

goals overview how_it_works auto-discovery_rule real_life_scenario

### 1 Goals

There are several goals of Zabbix network discovery module:

- Simplify deployment

Network discovery can be used to significantly simplify and speed up Zabbix deployment. It also makes possible creation of user friendly appliances.

- Simplify administration

Properly configured network discovery can simplify administration of Zabbix system a lot.

- Support of changing environments

Network discovery makes possible use of Zabbix in rapidly changing environments with no excessive administration.

### 2 Overview

Zabbix provides effective and very flexible network discovery functionality. 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

Every service and host (IP) checked by Zabbix network discovery module generates events which may be used to create rules for the following actions:

- Generating user notifications
- Adding and removing hosts
- Enabling and disabling hosts
- Adding hosts to a group
- Removing hosts from a group

- Linking hosts to a template
- Unlinking hosts from a template
- Executing remote scripts

The actions can be configured to respect host or service uptime and downtime.

**Warning:**
If Zabbix server is compiled with IPv6 support and **fping6** utility is missing, ICMP checks will fail for IPv4 devices as well. Only since Zabbix 1.8.2 IPv4 addresses are still processed by located **fping**.

### 3 How it works

Network discovery basically consists of two phases: Discovery and Actions.

First, we discover a host or a service, and generate discovery event or several events.

Then we process the events and apply certain actions depending of type of discovered device, IP, its status, up/down time, etc.

3.1 Discovery

Zabbix periodically scans IP ranges defined in network discovery rules. 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 defines set of service checks to be performed for IP range.

Events generated by network discovery module have Event Source "Discovery".

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. |

3.2 Actions

For a description of all conditions available for network discovery based events see action conditions.

For a description of all operations available for network discovery based events see operations.

### 4 Network discovery rule

Network discovery rule is a rule used by Zabbix to discover hosts and services.

Parameters of network discovery rule:

| Parameter | Description |
|---|---|
| **Name** | Name of the rule. For example, "Local network". |

| Parameter | Description |
|---|---|
| **IP range** | Range of IP addresses for discovery. It may have the following formats:<br>Single IP: 192.168.1.33<br>Range of IP addresses: 192.168.1.1-255<br>IP mask: 192.168.4.0/24<br>supported IP masks:<br>/16 - /30 for IPv4 addresses<br>/112 - /128 for IPv6 addresses<br>List:<br>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 should execute this rule. |
| **Checks** | Zabbix will use this list of checks for discovery of hosts and services.<br>List of supported checks: SSH, LDAP, SMTP, FTP, HTTP, POP, NNTP, IMAP, TCP, ZABBIX Agent, SNMPv1 Agent, SNMPv2 Agent, SNMPv3 Agent<br>Parameter Ports may be one of following:<br>Single port: 22<br>Range of ports: 22-45<br>List: 22-45,55,60-70 |
| **Device uniqueness criteria** | Uniqueness criteria may be:<br>**IP address** (no processing multiple-IP devices)<br>**One of discovery check** of the rule. Will be based either on a SNMP or Zabbix Agent check. |
| **Status** | **Active** – the rule is active and will be execute by Zabbix server<br>**Disabled** – the rule is not active. It won't be executed. |

**Warning:**
Each IP address should be included only once, having multiple rules for a single IP address can have unexpected behaviour such as having deadlocks and/or duplicate hosts in the database. The same could happen if two hosts having the same DNS name are included in separate discovery rules.

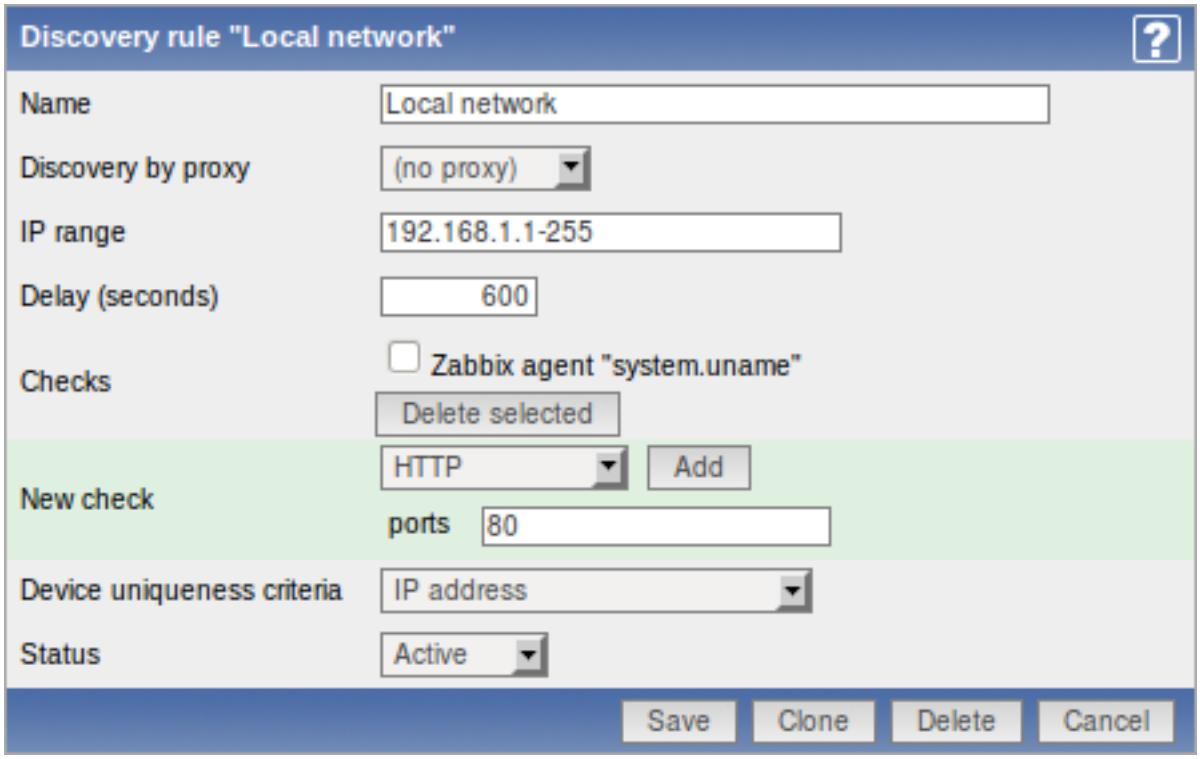
#### 5 Real life scenario

Suppose we would like to set up network discovery for local network having 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 host to monitoring if host uptime is more than 1 hour
- remove hosts if host downtime is more than 24 hours
- use Template_Windows for Windows hosts
- use Template_Linux for Linux hosts
- add Linux hosts to "Linux servers" group
- add Windows hosts to "Windows servers" group

Step 1

Define a network discovery rule for our IP range (*Configuration* → *Discovery* → *Create rule* button)



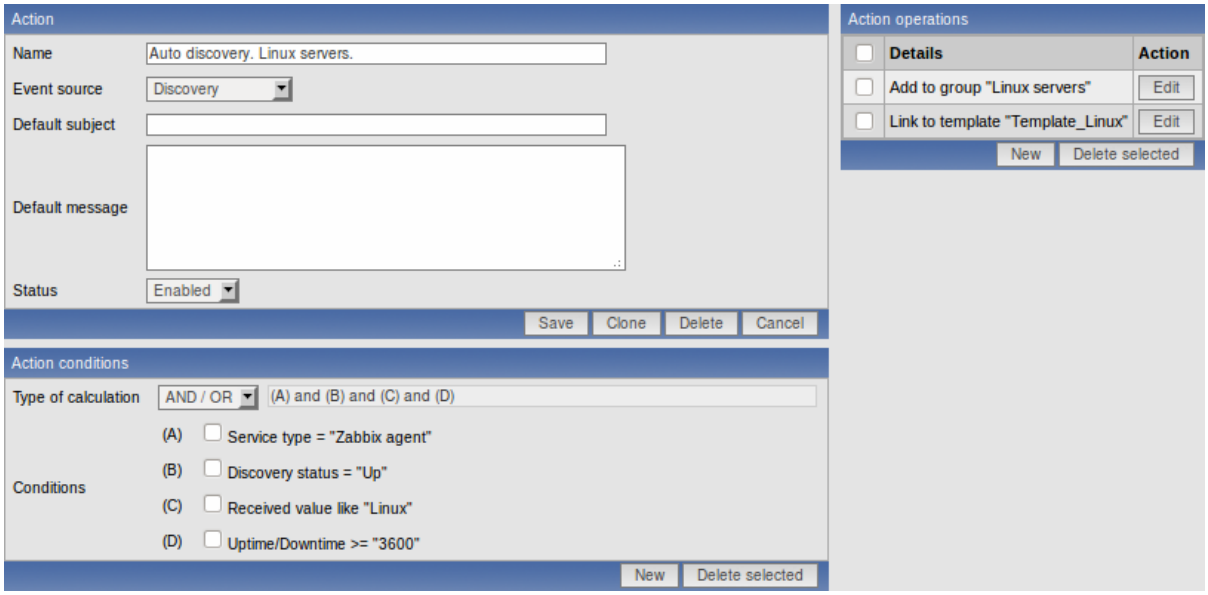

Zabbix will try to discover hosts in IP range of 192.168.1.1-192.168.1.255 by connecting to Zabbix Agents and getting value from **system.uname** key. A value received from an agent can be used to apply different actions for different operating systems. For example, link Windows servers to Template_Windows, Linux servers to Template_Linux.

The rule will be executed every 10 minutes (600 seconds).

When the rule is added, Zabbix will automatically start discovery and generation of discovery-based events for further processing.

Step 2

Define an action for adding newly-discovered Linux servers to the respective group/template. (*Configuration* → *Actions* → *Create Action* button)




The action will be activated if:

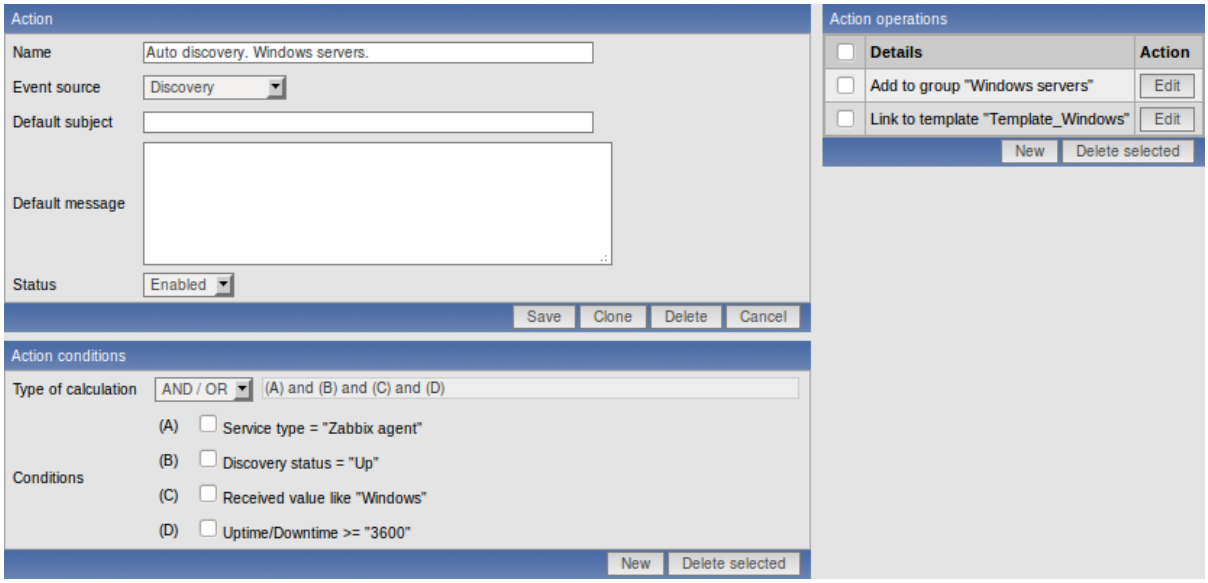
- ”Zabbix agent” service is ”Up”
- value of system.uname (the Zabbix Agent’s key we used in rule definition) contains ”Linux”
- Uptime is more than 1 hour (3600 seconds)

The action will execute the following operations:

- add the newly discovered host to ”Linux servers” group (also add host if it wasn’t added previously)
- link host to ”Template_Linux” template. Zabbix will automatically start monitoring the host using items and triggers from ”Template_Linux”.

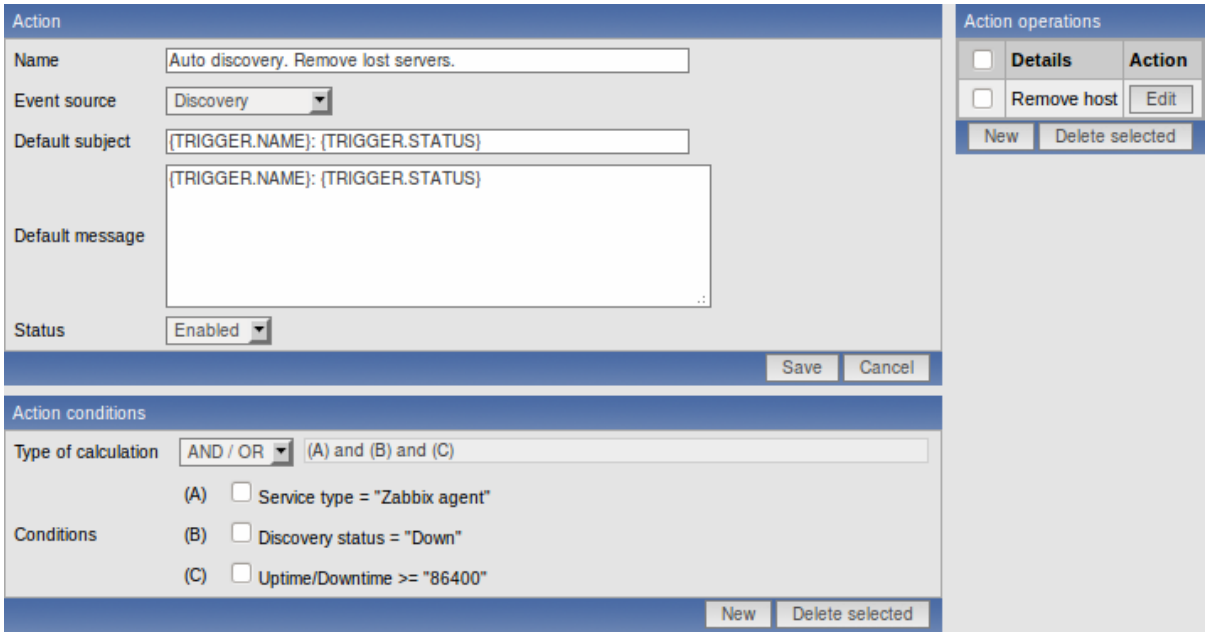
Step 3

Define an action for adding newly-discovered Windows servers to the respective group/template.




Step 4

Define an action for removing lost servers.




A server will be removed if ”Zabbix agent” service is ”Down” for more than 24 hours (86400 seconds).

## 12 Advanced SNMP Monitoring

special_mibs dynamic_indexes

#### 1 Special OIDs

Some of the most used SNMP OIDs are translated automatically to a numeric representation by Zabbix. For example, **ifIndex** is translated to **1.3.6.1.2.1.2.2.1.1**, **ifIndex.0** is translated to **1.3.6.1.2.1.2.2.1.1.0**.

The table contains list of the special OIDs.

| Special OID | Identifier | Description |
|---|---|---|
| ifIndex | 1.3.6.1.2.1.2.2.1.1 | A unique value for each interface. |
| ifDescr | 1.3.6.1.2.1.2.2.1.2 | A textual string containing information about the interface.This string should include the name of the manufacturer, the product name and the version of the hardware interface. |
| ifType | 1.3.6.1.2.1.2.2.1.3 | The type of interface, distinguished according to the physical/link protocol(s) immediately ’below’ the network layer in the protocol stack. |
| ifMtu | 1.3.6.1.2.1.2.2.1.4 | The size of the largest datagram which can be sent / received on the interface, specified in octets. |
| ifSpeed | 1.3.6.1.2.1.2.2.1.5 | An estimate of the interface’s current bandwidth in bits per second. |
| ifPhysAddress | 1.3.6.1.2.1.2.2.1.6 | The interface’s address at the protocol layer immediately ‘below’ the network layer in the protocol stack. |
| ifAdminStatus | 1.3.6.1.2.1.2.2.1.7 | The current administrative state of the interface. |
| ifOperStatus | 1.3.6.1.2.1.2.2.1.8 | The current operational state of the interface. |
| ifInOctets | 1.3.6.1.2.1.2.2.1.10 | The total number of octets received on the interface, including framing characters. |
| ifInUcastPkts | 1.3.6.1.2.1.2.2.1.11 | The number of subnetwork-unicast packets delivered to a higher-layer protocol. |
| ifInNUcastPkts | 1.3.6.1.2.1.2.2.1.12 | The number of non-unicast (i.e., subnetwork- broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol. |
| ifInDiscards | 1.3.6.1.2.1.2.2.1.13 | The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space. |
| ifInErrors | 1.3.6.1.2.1.2.2.1.14 | The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol. |
| 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. |

| Special OID | Identifier | Description |
|---|---|---|
| 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). |

**2 Use of dynamic indexes**

**Note:**
Dynamic indexes are supported since Zabbix version 1.5.

A special syntax for item OID can be used in order to deal with dynamic data (random IDs of network interfaces, etc). The syntax:

**<base OID of data>["index","<base OID of index>","<string to search for>"]**

For example, to get the **ifInOctets** value for the **GigabitEthernet0/1** interface on a Cisco device, use the following OID:

```
ifInOctets["index","ifDescr","GigabitEthernet0/1"]
```

| Parameter | Description |
|---|---|
| base OID of data | Base OID to use for data retrieval. |
| index | Method of processing. Currently one method is supported<br>**index** – search for index and append it to the base OID |
| base OID of index | The OID will be used to make a lookup for the string. |
| string to search for | The string is used for exact match with a value when doing lookup. Case sensitive. |

Another example, getting memory usage of apache process:

```
HOST-RESOURCES-MIB::hrSWRunPerfMem["index","HOST-RESOURCES-MIB::hrSWRunPath", "/usr/sbin/apache2"]
...

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 index, 5388. The index will be appended to the Data OID in order to receive value we are interested in:

```
HOST-RESOURCES-MIB::hrSWRunPerfMem.5388 = INTEGER: 31468 KBytes
```

**Note:**
Dynamic indexes are cached since Zabbix version 1.6.3.

**Note:**
Using dynamic indexes leads to more SNMP queries in Zabbix versions up to 1.7. Dynamic index lookup and data retrieval is performed in single connection since Zabbix version 1.7.

## 13 Monitoring of IPMI devices

goals ipmi_parameters ipmi_actions

### 1 Goals

There are several goals of Zabbix IPMI monitoring:

- Monitoring of health and availability of IPMI devices
- Remote IPMI based management functions

Remote restart, shutdown, halt, and other commands can be executed either automatically or manually from Zabbix front-end.

### 2 IPMI parameters

Zabbix IPMI monitoring works only for devices having IPMI support (HP iLO, Sun hardware, etc).

In order to use IPMI monitoring, a host must be configured to process IPMI commands. IPMI agent's IP address, port number, user name and password must be configured properly.

See configuration of hosts for more details.

### 3 IPMI actions

Two types of actions can be defined:

- automatic actions, which are executed automatically
- IPMI scripts, can be executed manually from Zabbix GUI

See corresponding sections of the Manual for more details.

## 14 Use of Proxies

Zabbix Proxies may greatly simplify maintenance of Zabbix environment and increase performance of the central Zabbix server.

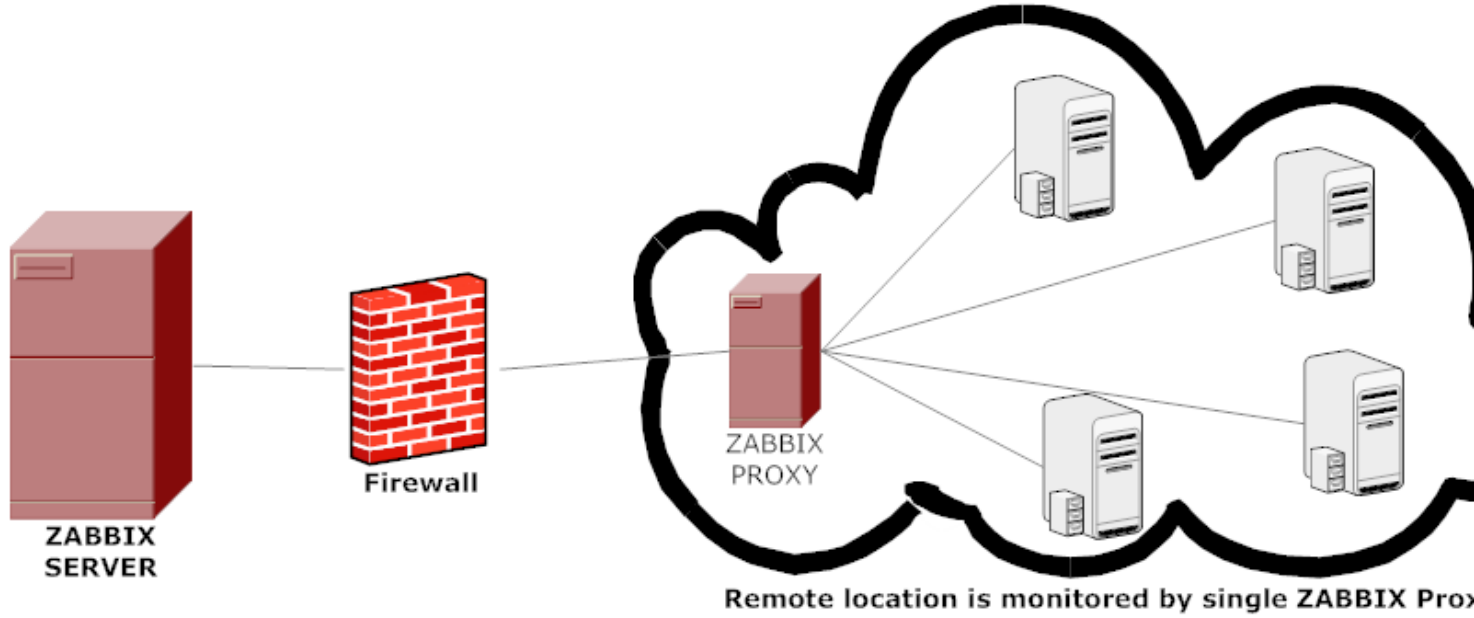
Also, use of Zabbix Proxies is the easiest way of implementing centralized and distributed monitoring, when all Agents and Proxies report to one Zabbix server and all data is collected centrally.

why_use_proxy proxy_vs_node configuration

### 1 Why use Proxy?

Zabbix Proxy can be used for many purposes:

- Offload Zabbix Server when monitoring thousands of devices
- Monitor remote locations
- Monitor locations having unreliable communications
- Simplify maintenance of distributed monitoring




**2 Proxy v.s. Node**

When making a choice between use of a Proxy or a Node, several considerations must be taken into account.

| | Lightweight | Works independently | Easy maintenance | Automatic DB creation[1] | Local administration | Ready for embedded hardware | One way TCP connections | Centralised configuration | Generates notifications |
|---|---|---|---|---|---|---|---|---|---|
| Node | No | **Yes** | No | No | **Yes** | No | Yes | No | **Yes** |
| Proxy | **Yes** | **Yes** | **Yes** | **Yes** | No | **Yes** | **Yes** | **Yes** | No |

**Note:**
[1] Automatic DB creation feature only works with SQLite. Other databases require manual setup.

**3 Configuration**

3.1 Managing proxies

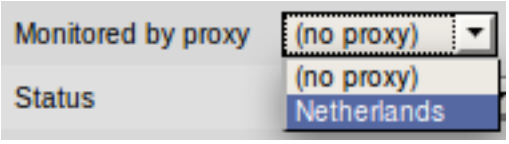
To open Zabbix proxy management, go to Administration → DM and select **Proxies** in the dropdown in the upper right corner. Here you can create, edit and delete proxies. For each proxy the last time when it contacted the server (either to send in new data or because of the heartbeat connection) is displayed.

Opening the proxy properties form allows you to select the hosts that should be monitored by that proxy.

**Attention:**
Zabbix proxy must use a separate database. Pointing it to the Zabbix server database will break the configuration.

3.2 Monitoring a host by a proxy

Each host can be monitored either by Zabbix Server or by Zabbix Proxy. Monitoring by a proxy is set up in Configuration → Hosts → open the host definition form:




If a host is configured to be monitored by a Proxy, the Proxy will perform gathering of performance and availability data for the host. The data will be collected by the Proxy and sent to Zabbix Server for further processing.

# 15 Distributed Monitoring

Zabbix can be configured to support hierarchical distributed monitoring.

goals overview configuration platform_independence configuring_single_node switching_between_nodes data_flow performance

## 1 Goals

There are several goals of the distributed monitoring:

- Get control of whole monitoring from a single or several locations

Zabbix administrator may control configuration of all Nodes from a single Zabbix WEB front-end.

- Hierarchical monitoring

This is for monitoring of complex multi-level environments.

- Monitor large complex environments

This is especially useful when monitoring several geographical locations.

- Offload the overhead from busy Zabbix server

Monitoring thousands of hosts using single Zabbix server? This may be for you!

## 2 Overview

Zabbix provides effective and reliable way of monitoring distributed IT infrastructure. Configuration of the whole distributed setup can be done from a single location via common WEB interface.

Zabbix supports up-to **1000** (one thousand) Nodes in a distributed setup. Each Node is responsible for monitoring of its own Location. Node can be configured either locally or by its Master node which has a copy of configuration data of all Child Nodes. Configuration of Child Nodes can be done in off line mode, i.e. when there are no connectivity between Master and Child Node.

Hierarchical distributed monitoring allows having tree-like structure of Nodes. Each Node reports to its Master Node only.

All Nodes may work even in case of communication problems. Historical information and events are stored locally. When communication is back, Child Nodes will optionally send the data to Master Node.

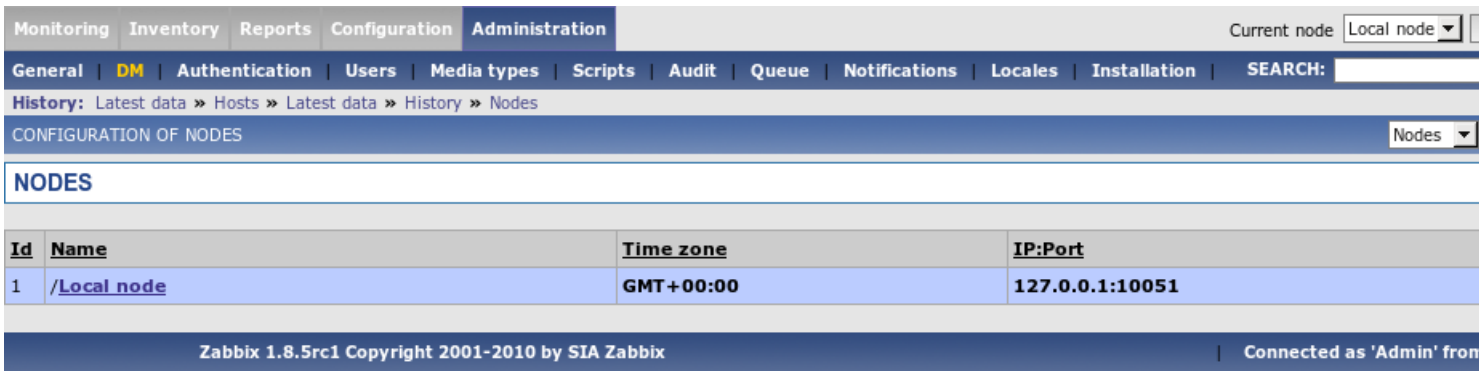
New Nodes can be attached to and detached from the Zabbix distributed setup without any loss of functionality of the setup. No restart of any Node required.

Each Node has its own configuration and works as a normal Zabbix Server.

## 3 Configuration

### 3.1 Configuration of Nodes

Node configuration is performed in *Administration* → *DM* section.



Parameters of a Node:

| Parameter | Description |
|---|---|
| **Name** | Unique node name. |
| **Id** | Unique Node ID. |
| **Type** | **Local** – Local node<br>**Remote** – Remote node |
| **Time zone** | Time zone of the Node. Zabbix automatically converts time stamps to local timezone when transferring time related data across nodes. |
| **IP** | Node IP address. Zabbix trapper must be listening on this IP address. |
| **Port** | Node Port number. Zabbix trapper must be listening on this port number. Default is 10051. |
| **Do not keep history older than (in days)** | For non local historical data only. Zabbix won't keep history of the node longer than N days. |
| **Do not keep trends older than (in days)** | For non local trend data only. Zabbix won't keep trends of the node longer than N days. |

3.2 Simple configuration

Our simple configuration consists of a Central Node and a Child Node.

Central Node will have total control over configuration of Child Node. Child Node will report to central node events, history and trends.

Central Node will have **NodeID**=1, while Child Node's **NodeID**=2.

Central Node IP: 192.168.3.2
Child Node IP: 192.168.3.5

For Central Node

**Step 1** Install Zabbix.

Follow standard installation instructions to create database, install Zabbix frontend and binaries.

**Step 2** Setup **NodeID** in server configuration file.

In file zabbix_server.conf:

```
NodeID=1
```

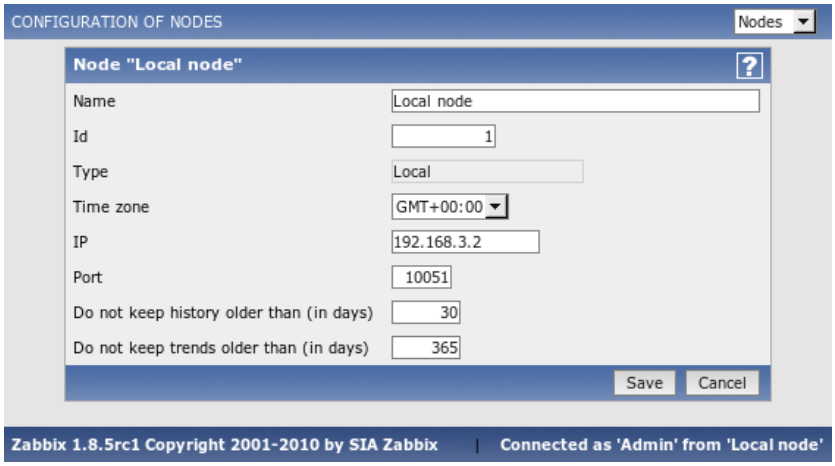
**Step 3** Convert database data.

Zabbix server has to be executed to convert unique IDs for use by first node.

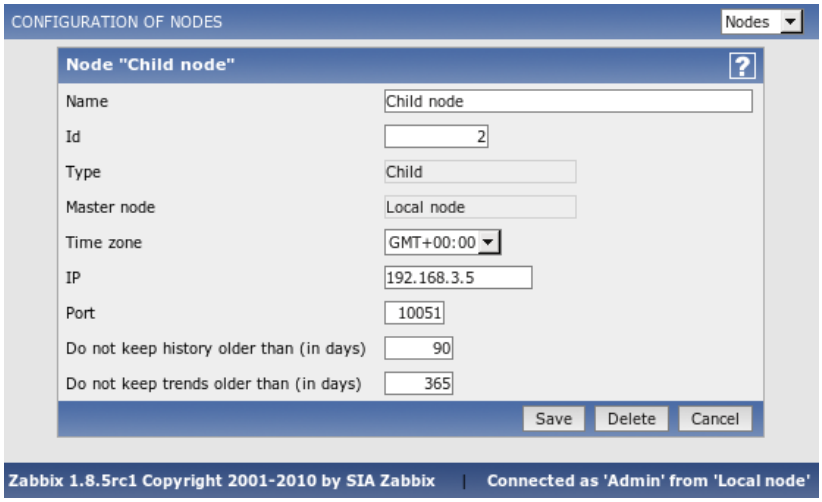
```
cd bin
./zabbix_server -n 1 -c /etc/zabbix/zabbix_server.conf
Converting tables ..................................................... done.

Conversion completed.
```

**Note:**
This should be executed only once. This option is not required to start Zabbix server! Running Zabbix server with the **-n** option does not start the server process.

**Step 4** Configure Node parameters.




**Step 5** Add child node.




**Step 6** Start Master Node.

We should see **NodeID** in startup messages of server log file:

```
31754:20070629:150342 server #16 started [Node watcher. Node ID:1]
```

For Child Node

**Step 1** Install Zabbix.

Follow standard installation instructions to create database, install Zabbix frontend and binaries.

**Step 2** Setup **NodeID** in server configuration file.

In file zabbix_server.conf:

```
NodeID=2
```

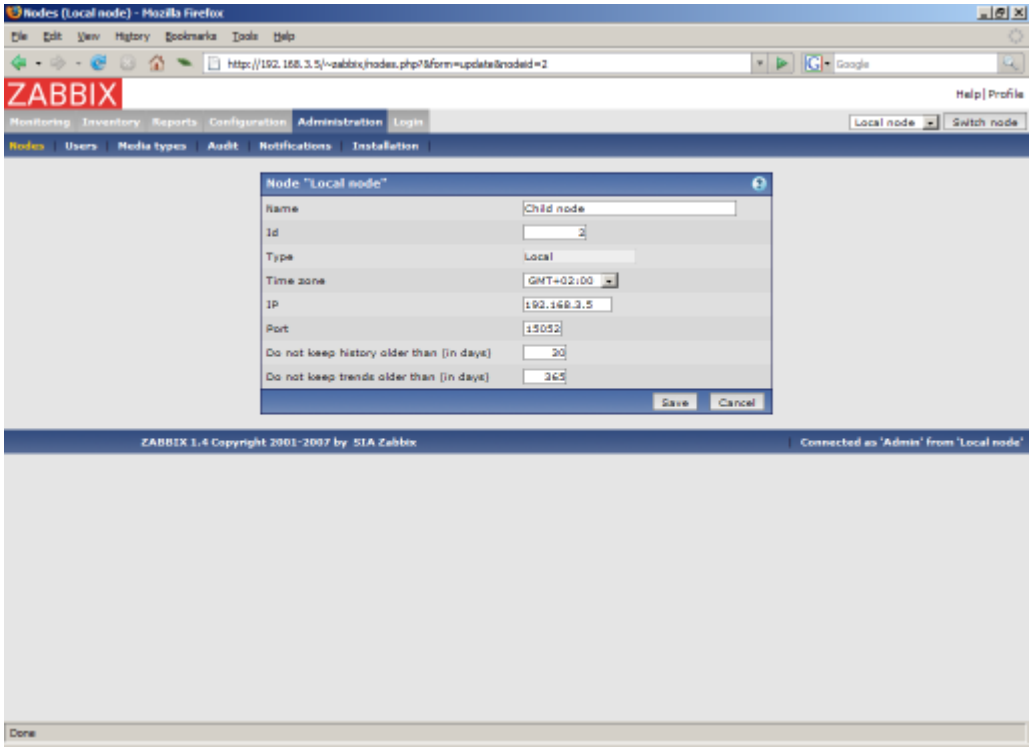
**Step 3** Convert database data.

Zabbix server has to covert all IDs to unique ones for the second node.

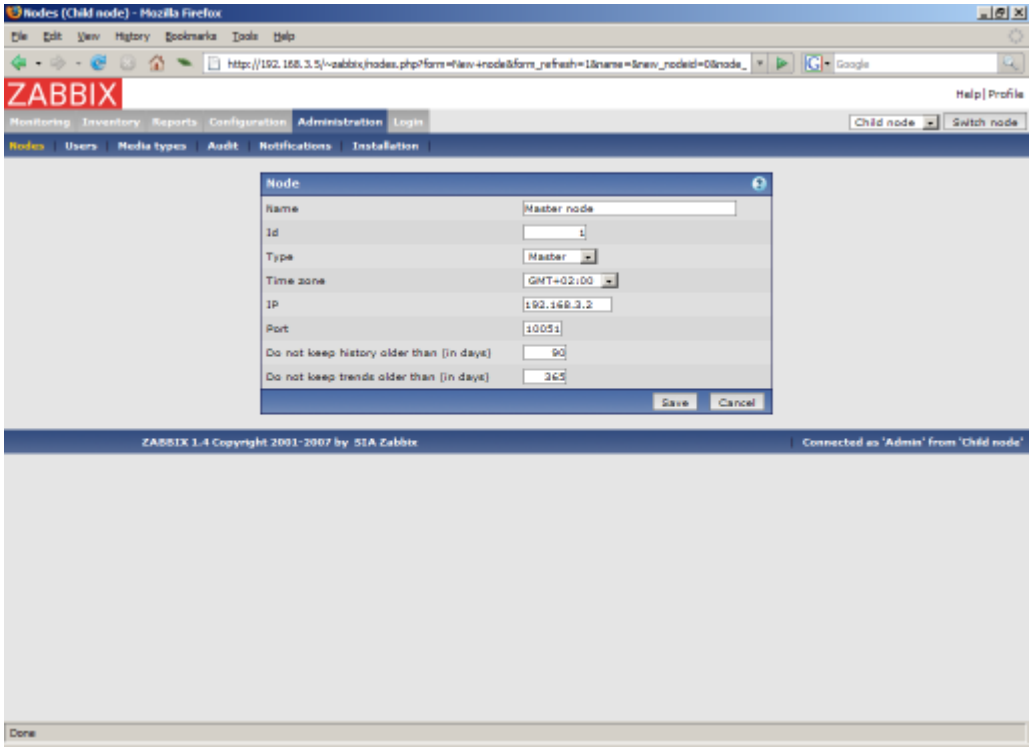
```
cd bin
./zabbix_server -n 2 -c /etc/zabbix/zabbix_server.conf
Converting tables ..................................................................... done.

Conversion completed.
```

**Note:**
This should be executed only once. This option is not required to start Zabbix server!

**Step 4** Configure Node parameters.




**Step 5** Add master node.



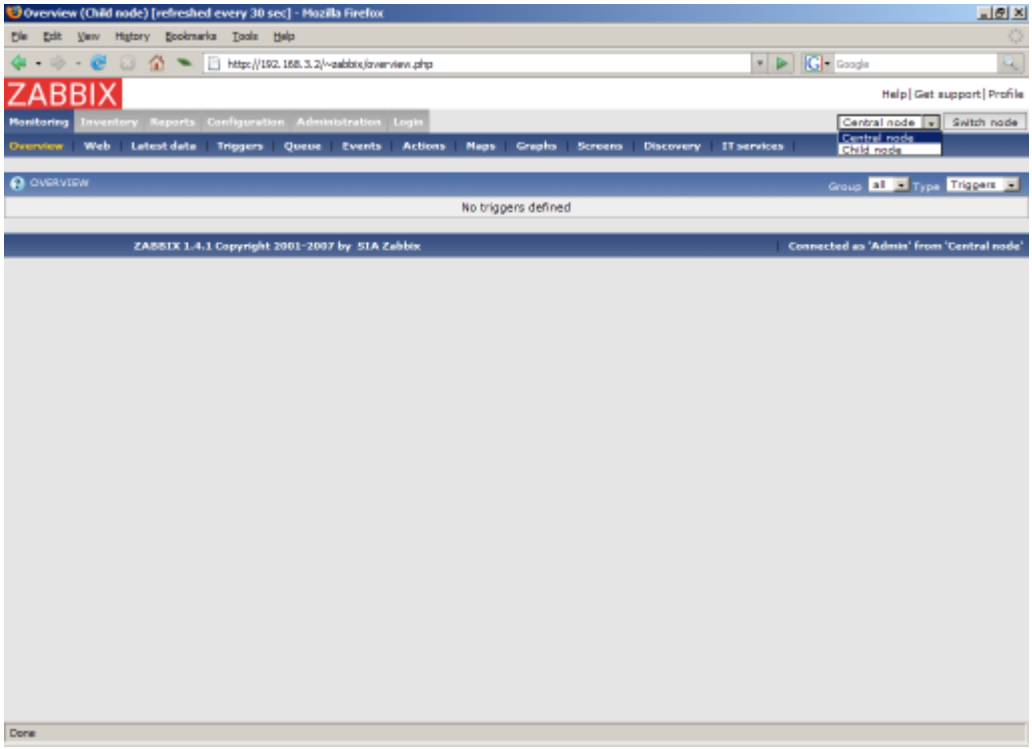

**Step 6** Start Child Node.

We should see **NodeID** in startup messages of server log file:

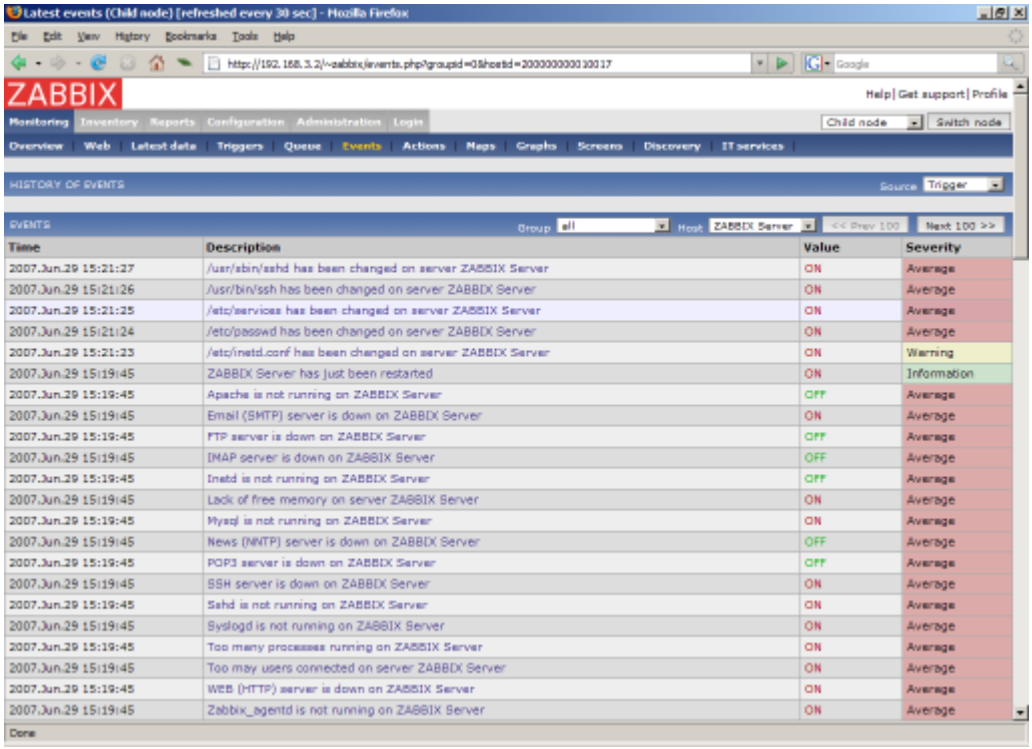
```
27524:20070629:150622 server #9 started [Node watcher. Node ID:2]
```

Does it work?

Selection of active nodes will appear automatically after nodes are defined:

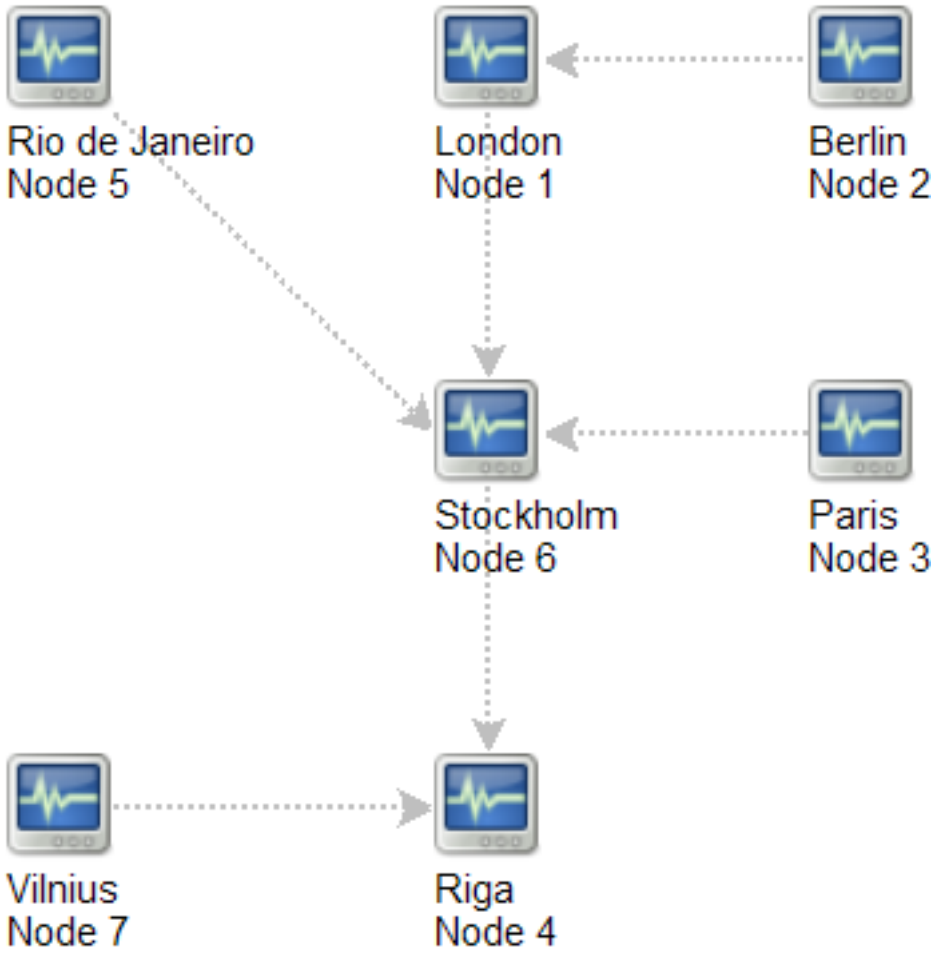



Add host for monitoring for Child Node node and see events coming to Master Node:




3.3 More complex setup

The setup consists of seven Nodes.  Each Node may be configured either locally (using local WEB interface) or from one of its Master Nodes.




In this example, Riga (node 4) will collect events from all child nodes. It may also optionally collect historical information as well.

**4 Platform independence**

A node may use its own platform (OS, hardware) and database engine independently of other nodes. Also child nodes can be installed without Zabbix frontend.

It may be practical to use less powerful hardware with Zabbix server running SQLite or MySQL MyISAM while nodes of higher levels may use combination of a better hardware with MySQL InnoDB, Oracle or PostgreSQL backend.

**5 Configuration of a single Node**

Every Node in distributed environment must be properly configured to have a unique Node ID. Additional steps

Step 1

Follow standard installation procedure.

Follow standard installation procedure but do not start Zabbix Server. Zabbix front end must be installed and configured. Zabbix database must be created and populated with data from **data.sql**.

Step 2

Configure zabbix_server.conf.

Add **NodeID** to Zabbix Server configuration file. **NodeID** must be a unique Node ID.

Step 3

Configure Master and Child Nodes.

Use Zabbix Frontend to configure details of Nodes having direct communication with the Node. Make sure that all IP addresses and port numbers are correct.

Step 4

Start Zabbix Node.

Start Zabbix Server:

```
shell> ./zabbix_server
```

If everything was configured properly, Zabbix node will automatically start configuration and data exchange with all nodes in distributed setup. You may see the following messages in server log file:

```
...
11656:20061129:171614 NODE 2: Sending data of node 2 to node 1 datalen 3522738
11656:20061129:171614 NODE 2: Sending data of node 2 to node 1 datalen 20624
...
```

**6 Switching between nodes**

When connecting to a node in distributed setup, a list of available child nodes is accessible in right-upper corner of the GUI. It displays current node.

All information available in the GUI belongs to the selected node.

**7 Data flow**

7.1 Child to Master

Each Child Node periodically sends configuration changes, historical data and events to its Master Node.

| Data | Frequency |
|---|---|
| **Configuration changes** | Every 120 seconds. |
| **Events** | Every 10 seconds. |
| **History** | Every 10 seconds. |

Child Node will resend data in case of communication problems.

Trends are calculated locally based on received historical data.

Zabbix does not send operational data across the nodes. For example, item-related information (last check, last value, etc) exists only locally.

**Note:**
Sending of Events and History can be controlled by configuration parameters **NodeNoEvents** and **NodeNoHistory**.

7.2 Master to Child

Each Master Node (a node with at least one child) periodically sends configuration changes to Child Nodes either directly or via other Child Nodes directly connected to the Master Node.

| Data | Frequency |
|---|---|
| **Configuration changes** | Every 120 seconds. |

Zabbix does not send configuration of a Master Node to Childs.

7.3 Firewall settings

Inter-node communications use TCP protocol only.

| Data flow | Source port | Destination port |
|---|---|---|
| **Child to Master** | Any | 10051 |

This is default port used by Zabbix trapper process.

#### 8 Performance considerations

Any node requires more processing resources in a distributed setup. Master Node must be powerful enough to process and store not only local data but also data received from its all Child Nodes. Network communications must be also fast enough for timely transfer of new data.

## 16 Maintenance mode for Zabbix GUI

Zabbix GUI can be temporarily disabled in order to prohibit access to the front-end. This can be useful for protection of Zabbix database from any changes initiated by users, thus protecting integrity of database.

Zabbix database can be stopped while Zabbix GUI is in the maintenance mode.

goals configuration how_it_looks_like

#### 1 Goals

There are several goals of the maintenance mode:

- Protect Zabbix database from any changes initiated by users
- Perform database maintenance
- Inform users about reason of the maintenance work
- Users from a range of IP addresses will be able to work with the GUI during the maintenance mode normally
- Automatic return to normal mode when maintenance is over

#### 2 Configuration

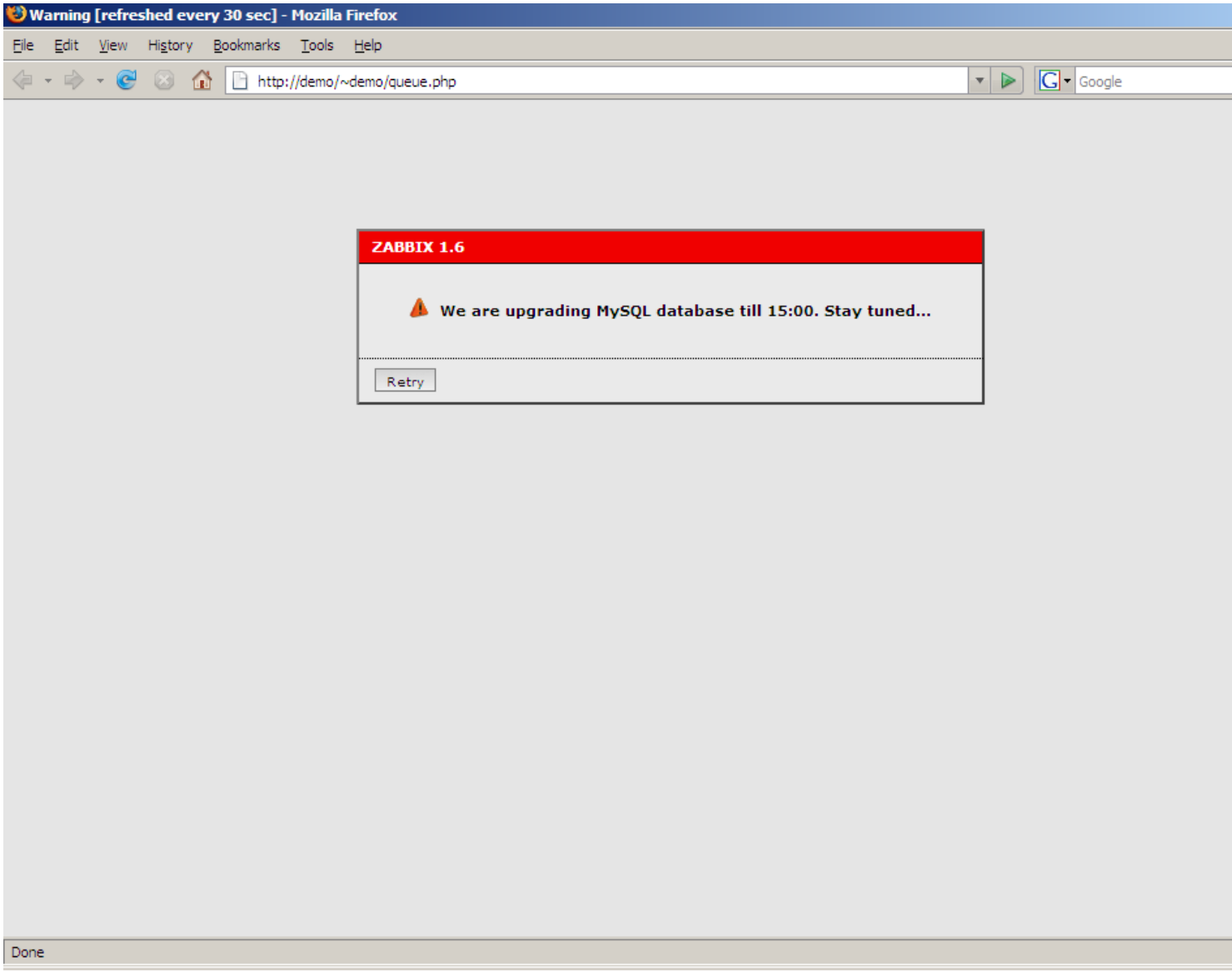
In order to enable maintenance mode, file conf/maintenance.conf.php must be modified to uncomment the following lines:

```
// 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'] = 'Zabbix is under maintenance.';
```

| Parameter | Details |
|---|---|
| **ZBX_DENY_GUI_ACCESS** | Enable maintenance mode:<br>1 – maintenance mode is enabled, disabled otherwise |
| **ZBX_GUI_ACCESS_IP_RANGE** | Connections from these IP addresses will be allowed with no maintenance mode.<br>For example:<br>192.168.1.1-255 |
| **warning_msg** | Informative message. |

#### 3 How it looks like

The following screen will be displayed while in maintenance mode. The screen is refreshed every 30 seconds in order to return to normal state withiout user intervention when maintenance is over.




## 17 WEB Interface

There are several useful features of ZABBIX WEB interface:

- almost all screens support full-screen mode
- Ctrl + Mouse click make possible selection of multiple list elements (hosts, items, triggers, etc)
- sound alarm can be switched on and off in Status of Triggers view
- a new theme can be created to match your preferences or a company color schema

creating_own_theme configuration administration page_parameters

### 1 Creating your own theme

By default, Zabbix provides number of predefined themes. You may follow this step-by-step procedure in order to create your own. Feel free to share result of your work with Zabbix community if you created something nice.

Step 1

Create your own CSS file.

The file can be based on existing CSS files coming with Zabbix. For example, you may take Black&Blue CSS file from styles/css_bb.css and create new css_new.css.

Step 2

Place the new CSS file into correct location.

The file you created, css_new.css, into directory styles/.

Step 3

Edit include/forms.inc.php.

Open this file for editing, search for css_bb.css. There are two pieces of code that have to be amended.

Original code:

```
$cmbTheme = new CComboBox('theme',$theme);
$cmbTheme→AddItem(ZBX_DEFAULT_CSS,S_SYSTEM_DEFAULT);
$cmbTheme→AddItem('css_ob.css',S_ORIGINAL_BLUE);
$cmbTheme→AddItem('css_bb.css',S_BLACK_AND_BLUE);
```

Modified code:

```
$cmbTheme = new CComboBox('theme',$theme);
$cmbTheme→AddItem(ZBX_DEFAULT_CSS,S_SYSTEM_DEFAULT);
$cmbTheme→AddItem('css_ob.css',S_ORIGINAL_BLUE);
$cmbTheme→AddItem('css_bb.css',S_BLACK_AND_BLUE);
$cmbTheme→AddItem('css_new.css','MY_COOL_THEME');
```

**Attention:**
Note that original themes use constants, but the new example uses string (enclosed in apostrophes). You should not omit apostrophes, as that will result in warnings. If you want your theme name to be translatable, you must add the constant used for name in locale files - in that case make sure to prefix it with **S_**.

Step 4

You should also add your new theme to the **config.php** file:

```
$combo_theme→addItem('css_new.css','MY_COOL_THEME');
```

Step 5

Activate new theme.

In Zabbix GUI, you may either set this theme to be a default one or change your theme in user profile.

Enjoy new look and feel!

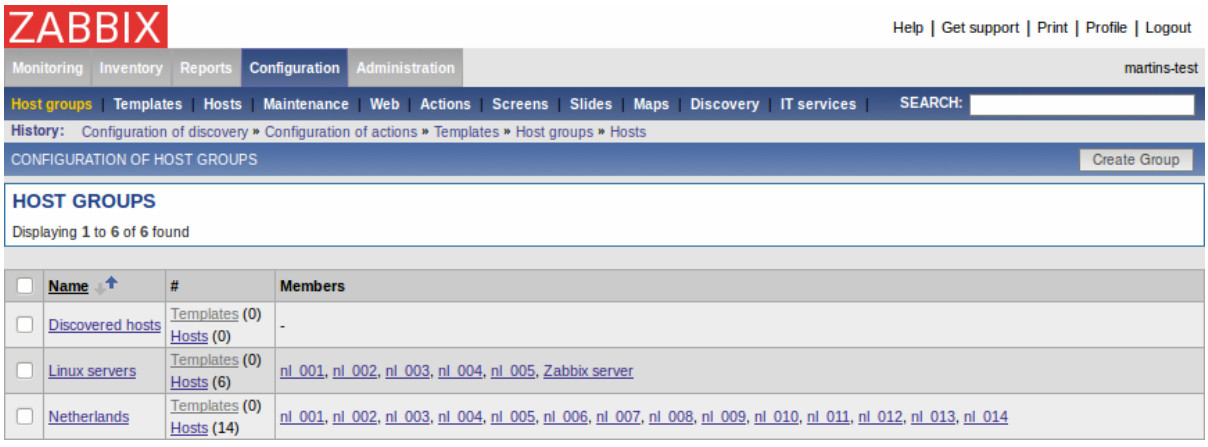
## 2 Configuration

### 2.1 Host groups

*Configuration → Host groups*

On this screen you can set up host groups and manage host group information.

A list of existing groups is displayed.

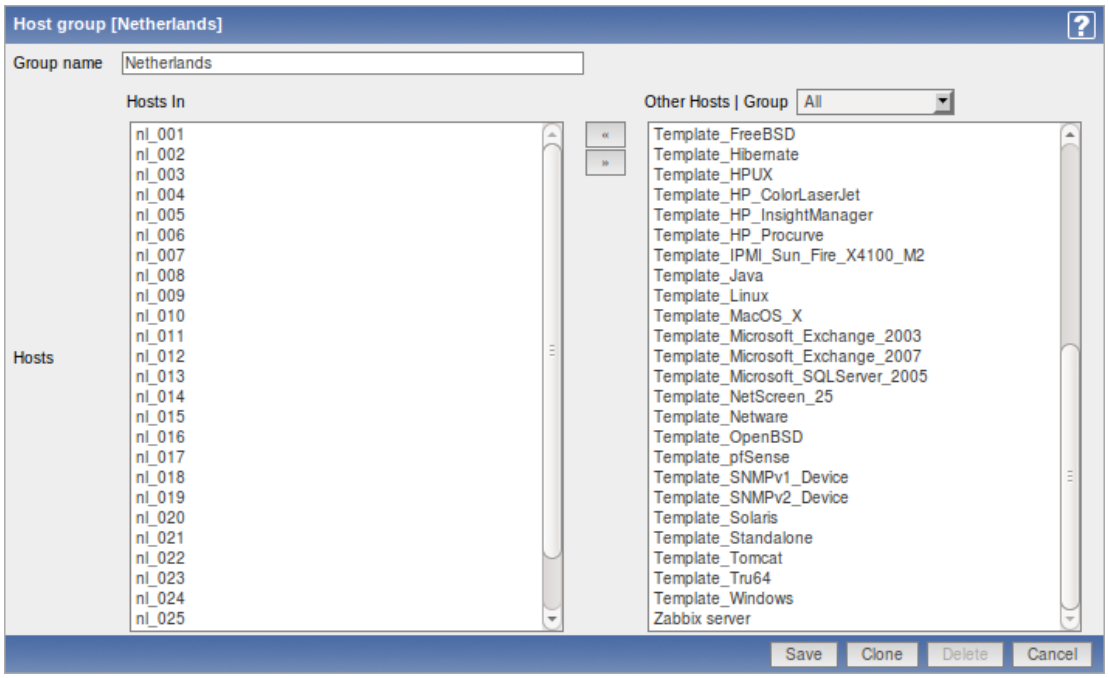



Displayed data:

| Parameter | Description |
|---|---|
| **Name** | Host Group name. |
| **#** | Number of group members (hosts). |
| **Members** | List of host group members. |

Click on *Create Group* in the upper right corner of the screen if you wish to add a group. If you wish to edit an existing group, click on its name in the list. A form is displayed where you can edit details of a host group.

Configuring a host group




Configuration parameters:

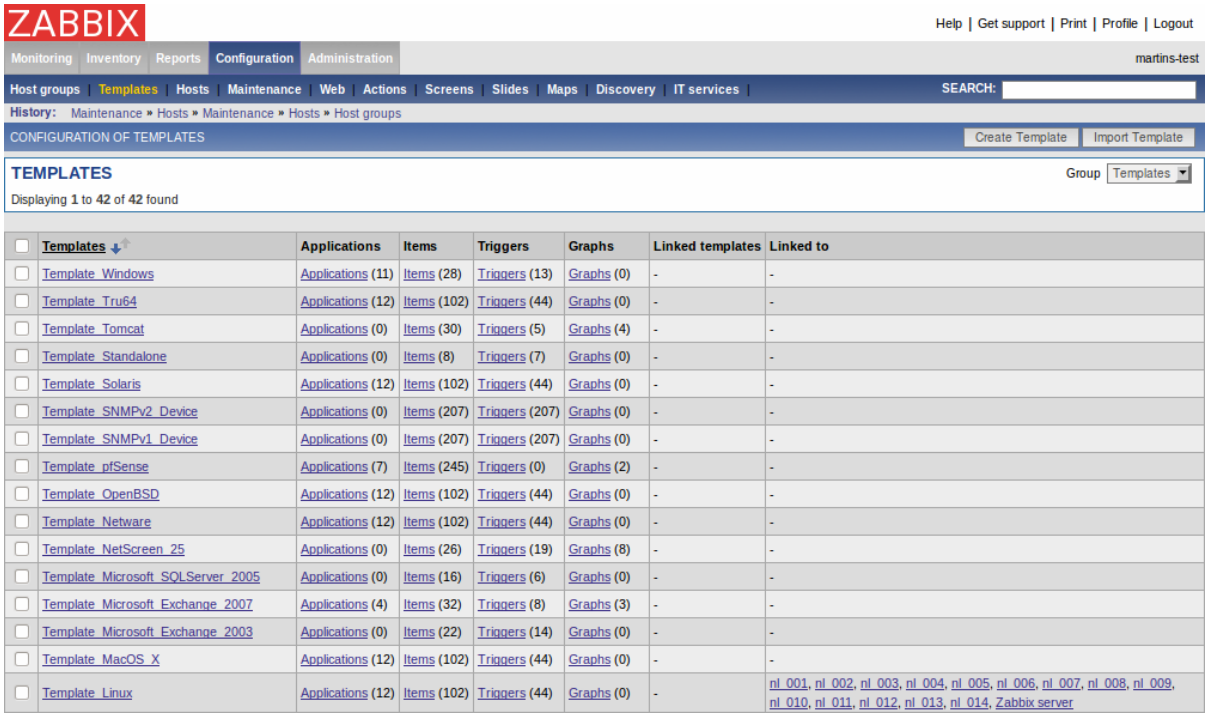
| Parameter | Description |
|---|---|
| **Group name** | Unique host group name. |
| **Hosts** | List of hosts, members of the group. |

2.2 Templates

*Configuration → Templates*

On this screen you can set up and manage host templates.

A list of existing templates is displayed.

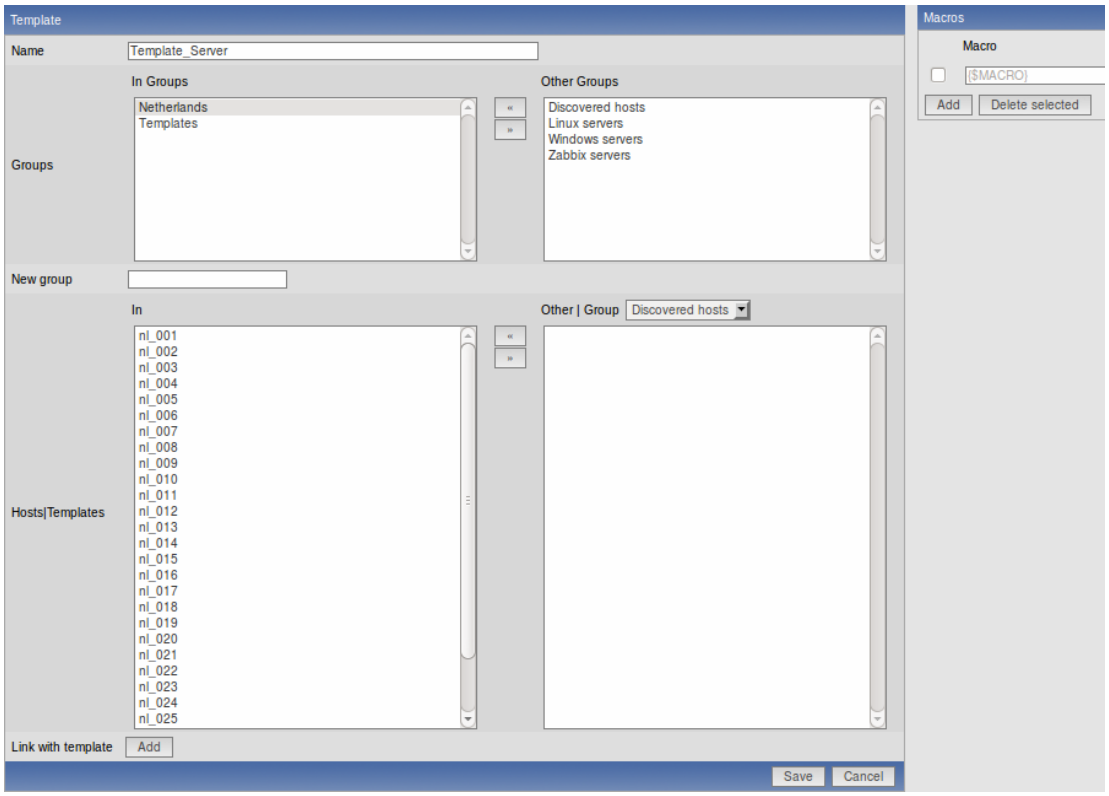



Displayed data:

| Parameter | Description |
|---|---|
| **Name** | Template name. |
| **Templates** | List of hosts linked to this template. |

Click on *Create Template* in the upper right corner of the screen if you wish to add a template. If you wish to edit an existing template, click on its name in the list. A form is displayed where you can edit details of a template.

Configuring a template




Configuration parameters:

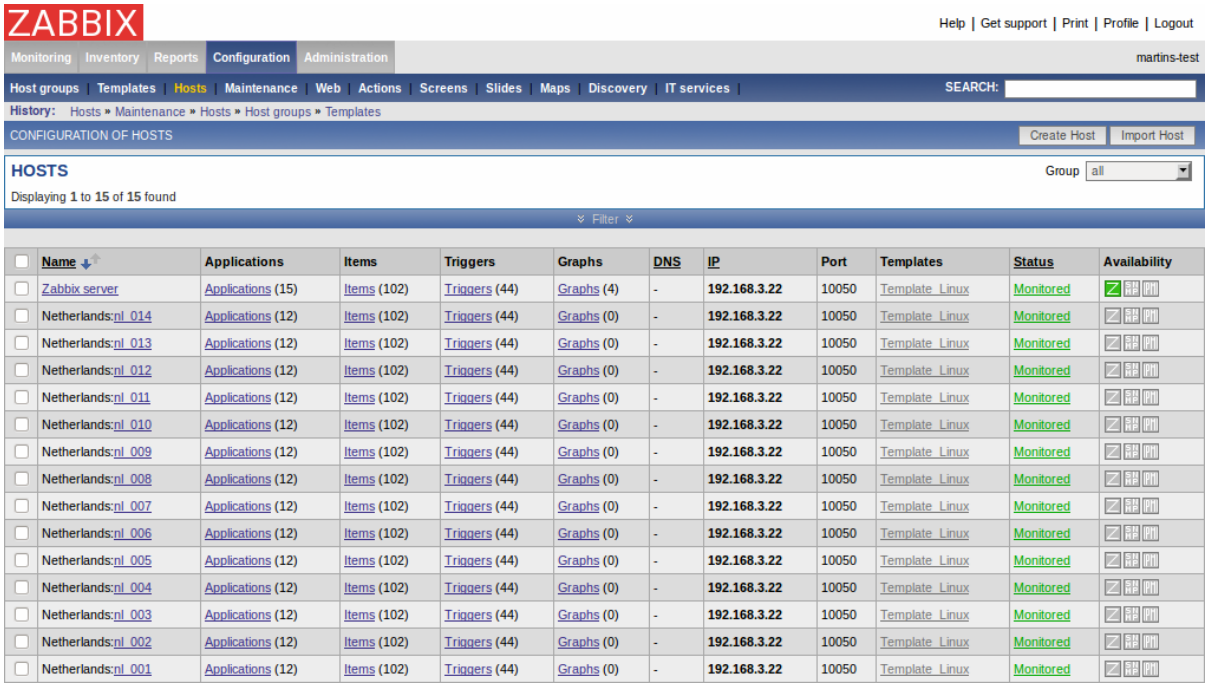
| Parameter | Description |
|---|---|
| **Name** | Unique template name. |
| **Groups** | List of host groups the template belongs to. |
| **New group** | New group can be created and linked to the template.<br>Ignored, if empty. |
| **Hosts/Templates** | List of hosts/templates linked to the template. |
| **Link with template** | Link template with one or more templates.<br>Information about items, triggers and graphs will be inherited from the templates. |

2.3 Hosts

*Configuration → Hosts*

On this screen you can set up hosts and manage host-related information.

A list of monitored hosts is displayed.

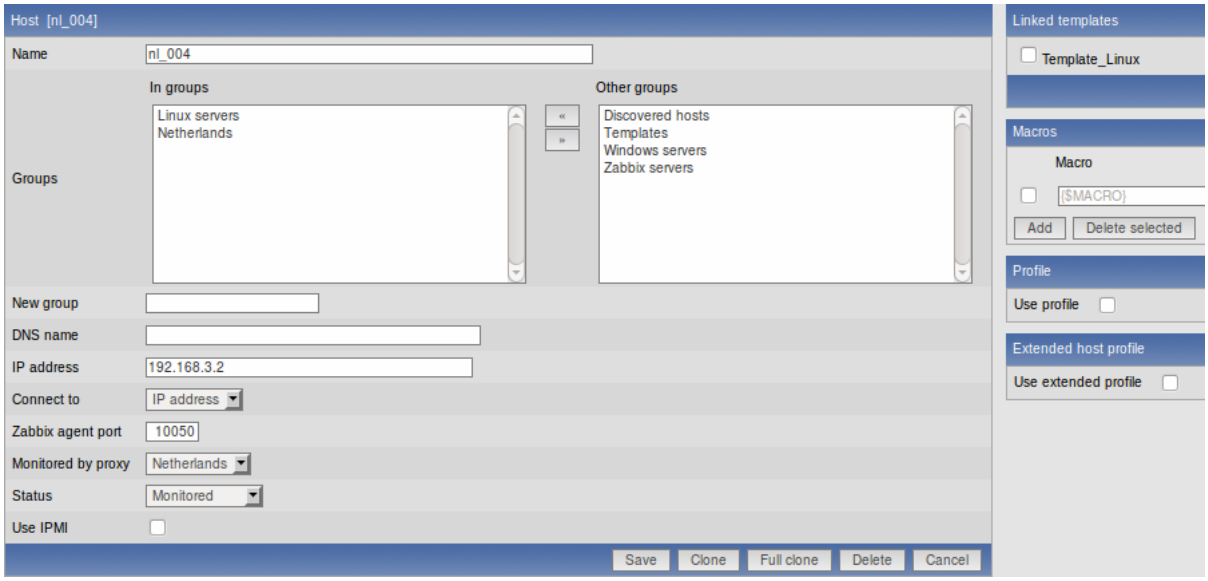



Displayed data:

| Parameter | Description |
|---|---|
| **Name** | Unique host name. |
| **DNS** | Host DNS name if used. |
| **IP** | Host IP address if used. |
| **Port** | Zabbix agent port number. Ignored if Zabbix agent items are not used. |
| **Templates** | List of first and second level (in parenthesis) templates linked to the host. |
| **Status** | Host Status:<br>**Monitored** - Host is active and being monitored<br>**Disabled** - Host disabled |
| **Availability** | Agent (Zabbix, SNMP, IPMI) availability<br>With the default theme:<br>green icon - agent is up and running<br>grey icon - availability is not known<br>red icon - agent is not available |
| **Error** | Any errors related to use of agent based checks. |

Click on *Create Host* in the upper right corner of the screen if you wish to add a host. If you wish to edit an existing host, click on its name in the host list. A form is displayed where you can edit details of a host.

Configuring a host




Configuration parameters:

| Parameter | Description |
|---|---|
| **Name** | Unique host name. |
| **Groups** | List of host groups the host belongs to. |
| **New group** | New group can be created and linked to the host. Ignored, if empty. |
| **DNS name** | Optional host DNS name. |
| **IP address** | Optional host IP address. |
| **Connect to** | Zabbix server will use this setting to retrieve data from agents:<br>**DNS** name - Connect to host DNS name<br>**IP address** - Connect to host IP (recommended) |
| **Port** | Zabbix agent TCP port number. Default value is 10050. |
| **Monitored by proxy** | The host can be monitored either by Zabbix server or one of Zabbix proxies:<br>**(no proxy)** - host is monitored by Zabbix server<br>**Proxy name** – host is monitored by Zabbix proxy "Proxy name" |
| **Status** | Host status:<br>**Monitored** – Host is active, ready to be monitored<br>**Not monitored** – Host is not active, thus not monitored |
| **Link with template** | Link host with one or more templates. Information about items, triggers and graphs will be inherited from the templates.<br>**Unlink** – unlink from template, but preserve information about items, triggers and graphs<br>**Unlink and clear** – unlink from template and remove all information inherited from the template |
| **Use IPMI** | Enable IPMI management functionality for this host. |
| **IPMI IP address** | IP address of IPMI management device. |
| **IPMI port** | Port number of the IPMI device. |
| **IPMI privilege level** | Keep default setting here, User. |
| **IPMI username** | User name for authentication. |
| **IPMI password** | Password for authentication. |
| **Use profile** | Enable or disable use of Host profile. |
| **Use extended profile** | Enable or disable use of extended Host profile. |

Both host and template definition forms include buttons "Clone" and "Full clone".

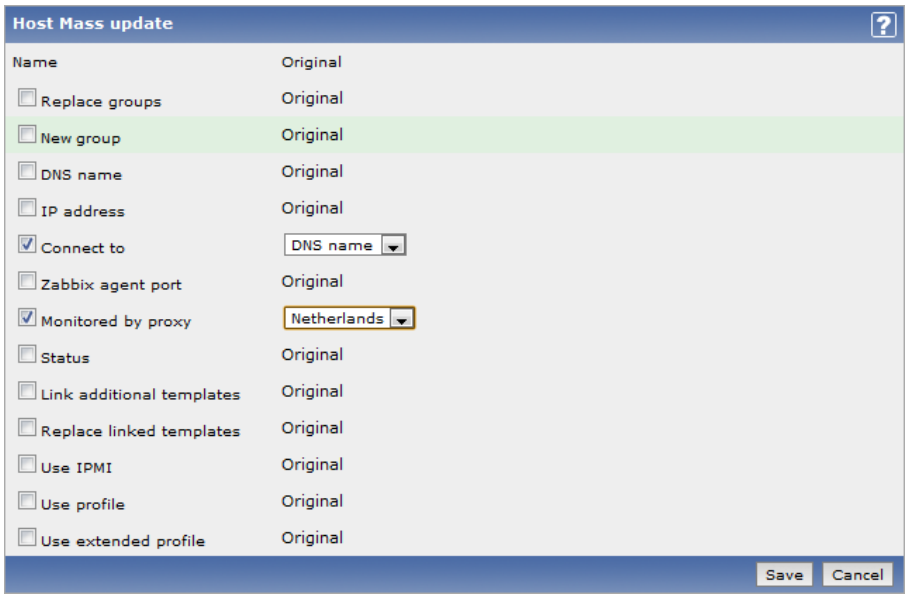
"Clone" will add a new host or template based on the configuration parameters of the existing host/template and that will include template linkage (thus also all templated item, trigger, graph and application information from those templates). "Full clone" in addition to that will also clone directly attached items, triggers, graphs and applications.

*Note*: When a new 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 (changed item interval, history period) will not be cloned to the new host; instead they will be as on the template.

Mass-updating hosts

Mass update is a very effective way of changing attributes for a number of hosts at once.

To update some hosts, check them in the host list, then select ”Mass update” in the dropdown below the host list and click on ”Go”. A form is displayed where you can select what attributes for the hosts you want to update.



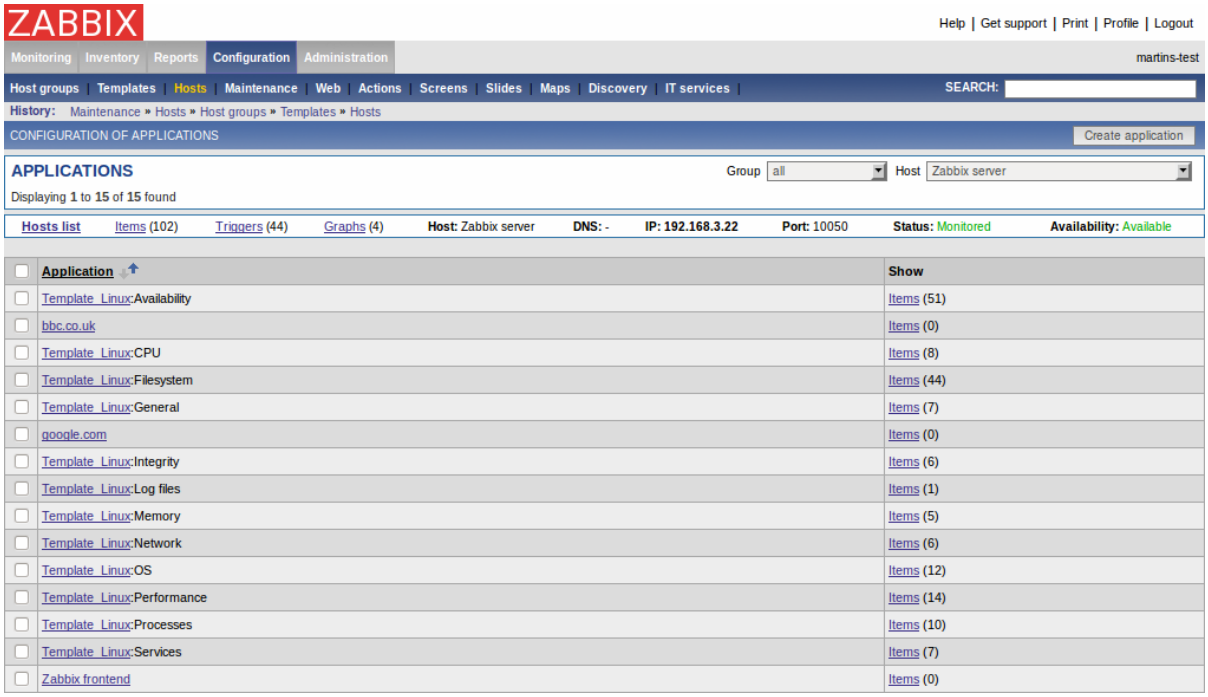

2.3.1 Applications

*Configuration → Hosts*
*Configuration → Templates*

From the list of hosts or templates you can access the applications that are linked to a host or template by clicking in the *Applications* column.

On the applications screen you can view and manage applications.

A list of applications linked to the host/template is displayed first. To view the other existing applications, select Group/Host display options in the dropdown menus above.



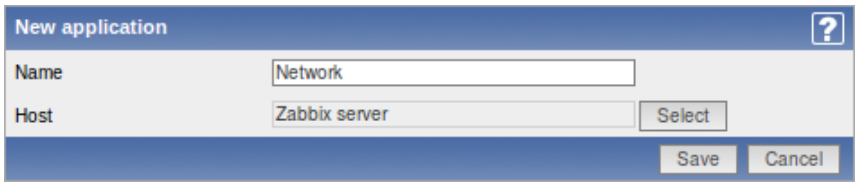

Displayed data:

| Parameter | Description |
|---|---|
| **Application** | Application name. |

| Parameter | Description |
|---|---|
| **Show** | Link to host items, also displays number of items (members of the application). |

Click on *Create application* in the upper right corner of the screen if you wish to add an application. If you wish to edit an existing application, click on its name in the list. A form is displayed where you can edit details of an application.

Configuring an application




Configuration parameters:

| Parameter | Description |
|---|---|
| **Name** | Application name. Must be unique within one host. |
| **Hosts** | Host name the application is linked to. |

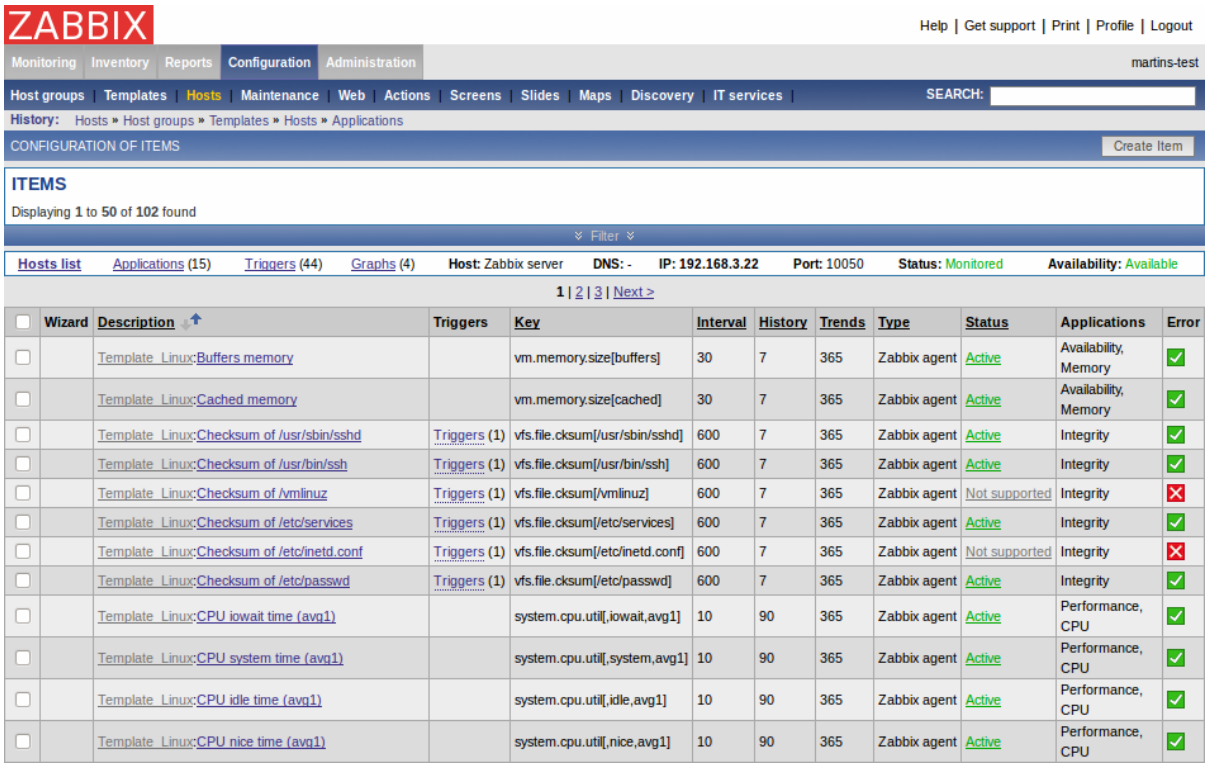
2.3.2 Items

*Configuration → Hosts*
*Configuration → Templates*

From the list of hosts or templates you can access the items that are linked to a host or template by clicking in the *Items* column.

On the items screen you can view and manage items.

A list of existing items is displayed.



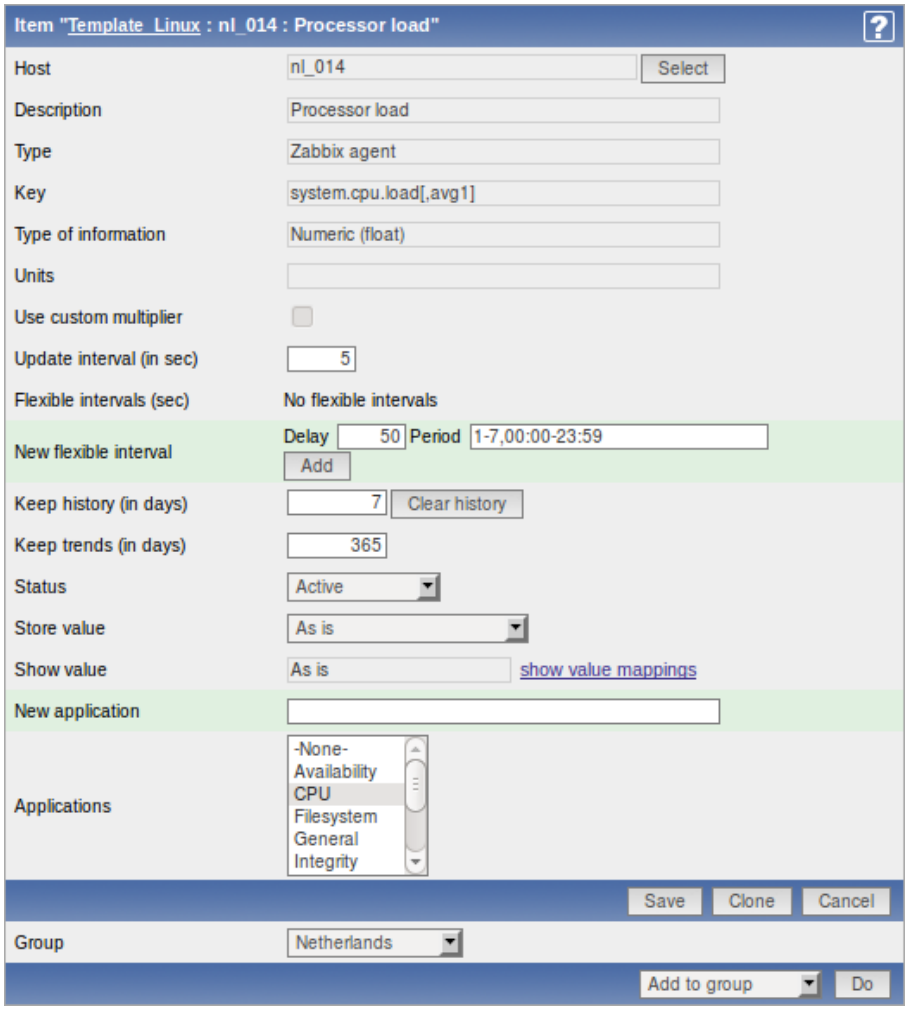

Displayed data:

| Parameter | Description |
|---|---|
| **Description** | Item description (name). |
| **Key** | Unique item key. |
| **Update interval** | Frequency of the check. |

| Parameter | Description |
|---|---|
| **History** | Number of days Zabbix keeps detailed historical data. |
| **Trends** | Number of days Zabbix keeps trends data. |
| **Type** | Item type. |
| **Status** | Item status. |
| **Applications** | List of applications the item belongs to. |
| **Error** | Any errors related to this item. |

Click on *Create Item* in the upper right corner of the screen if you wish to add an item. If you wish to edit an existing item, click on its name in the list. A form is displayed where you can edit details of an item.

Configuring an item




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

Item attributes:

| Parameter | Description |
|---|---|
| **Description** | Item description.<br>It may contain these macros:<br>**$1,$2...$9** - first, second... ninth parameter of item key<br>For example: Free disk space on $1<br>If item key is “vfs.fs.size[/,free]”, the description will be automatically changed to ”Free disk space on /” |
| **Type** | Item type. See sections below for detailed description of each type. |
| **Key** | Item key. The key must be unique within a single host.<br>The key value must be supported by the agent or Zabbix server if key type is ’Zabbix Agent’, ’Zabbix Agent (active)’, ’Simple check’ or ’Zabbix aggregate’. |

| Parameter | Description |
|---|---|
| **Type of information** | Type of data as stored in the database after performing conversions, if any.<br>**Numeric (unsigned)** – 64bit unsigned integer<br>**Numeric (float)** – floating point number<br>**Character** – character (string) data limited to 255 bytes<br>**Log** – log file. Must be set for keys log[].<br>**Text** – text of unlimited size |
| **Data type** | Data type is used for integer items in order to specify the expected data type:<br>**Decimal** – data in decimal format<br>**Octal** – data in octal format<br>**Hexadecimal** – data in hexadecimal format<br>Zabbix will automatically perform conversion to numeric.<br>This is supported starting from version 1.8. |
| **Units** | If set, Zabbix will add the unit postfix to all received values.<br>Till Zabbix 1.8.2, default multiplier is 1024, and some units have special processing:<br>**b, bps** - 1000 is 1K, special processing for bits.<br>Since Zabbix 1.8.2, default multiplier is 1000, and special processing is used for units **B**, where multiplier is 1024.<br>For example, if units are set to **B**, Zabbix will display:<br>1 as 1B<br>1024 as 1KB<br>1536 as 1.5KB<br>**unixtime** – translated to "yyyy.mm.dd hh:mm:ss". To translate correctly, the received value must be a *Numeric (unsigned)* type of information.<br>**uptime** – translated to "hh:mm:ss" or "N days, hh:mm:ss"<br>**s** – translated to "yyy mmm ddd hhh mmm sss ms", parameter is treated as number of seconds. 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. |
| **Use multiplier** | Pre-process received values.<br>**Do not use** - do not pre-process received values<br>**Custom multiplier** – multiply received values by value defined in Custom multiplier<br>Use this option to convert values received in KB, MBps, etc into B, Bps. Otherwise Zabbix cannot correctly set prefixes (K, M, G etc). |
| **Custom multiplier** | Multiply all received value by this integer or floating-point value. |
| **Update interval (in sec)** | Refresh this item every N seconds.<br>*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** | List of exceptions for Update Interval. For example:<br>Delay: 10 Period: 1-5,09:00-18:00 – refresh set to 10 seconds for working hours. Otherwise default update interval will be used.<br>If multiple flexible intervals overlap, the smallest *Delay* value is used for the overlapping period.<br>See Time period specification page for description of Period format.<br>*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. |
| **Keep history (in days)** | Keep detailed history for N days in the database. Older data will be removed by Housekeeper. |
| **Keep trends (in days)** | Keep aggregated (hourly min, max, avg, count) detailed history for N days in the database. Older data will be removed by Housekeeper. |

| Parameter | Description |
|---|---|
| **Status** | **Active** - active (normal) status. Zabbix will process this item.<br>**Disabled** – item is disabled. This item will not be processed.<br>**Not supported** – item is not supported by Zabbix or SNMP agent. This item will not be processed, however Zabbix may try to periodically set status of such items to Active if configured. |
| **Store value** | **As is** – no pre-processing<br>**Delta (speed per second)** – evaluate value as (**value**-**prev_value**)/(**time**-**prev_time**), where<br>*value* – current value<br>*value_prev* – previously received value<br>*time* – current timestamp<br>*prev_time* – timestamp of previous value<br>This setting is extremely useful to get speed per second based on constantly growing value.<br>*Note*: 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.<br>**Delta (simple change)** – evaluate as (**value**-**prev_value**), where<br>*value* – current value<br>*value_prev* – previously received value |
| **Show value** | Apply value mapping to this item. Value mapping does not change received values, it is for displaying data only.<br>It works with integer items only.<br>For example, ”Windows service states”. |
| **Log time format** | Available for items of type **Log** only. Supported placeholders:<br>* **y**: *Year (0001-9999)*<br>* **M**: *Month (01-12)*<br>* **d**: *Day (01-31)*<br>* **h**: *Hour (00-23)*<br>* **m**: *Minute (00-59)*<br>* **s**: *Second (00-59)*<br>Leaving this field blank means don’t try to parse the timestamp.<br>For example, consider the following line from the Zabbix agent log file:<br>” 23480:20100328:154718.045 Zabbix Agent started. Zabbix 1.8.2 (revision 11211).”<br>It begins with six character positions for PID, followed by date, time, and the rest of the line.<br>Log time format for this line would be ”ppppppp:yyyyMMdd:hhmmss”.<br>Note that ”p” and ”:” chars are just placeholders and can be anything but ”yMdhms”. |
| **Applications** | Link item to one or more applications. |

Up to version 1.8.1 Zabbix supports the following unit prefixes:

- K (Kilo);
- M (Mega);
- G (Giga);
- T (Tera);

Since version 1.8.2, additionally supported prefixes include:

- P (Peta);
- E (Exa);
- Z (Zetta);
- Y (Yotta);

See more details about items in other sections of the Manual.

** Unit blacklist **

By default, specifying a unit for an item will result in 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:

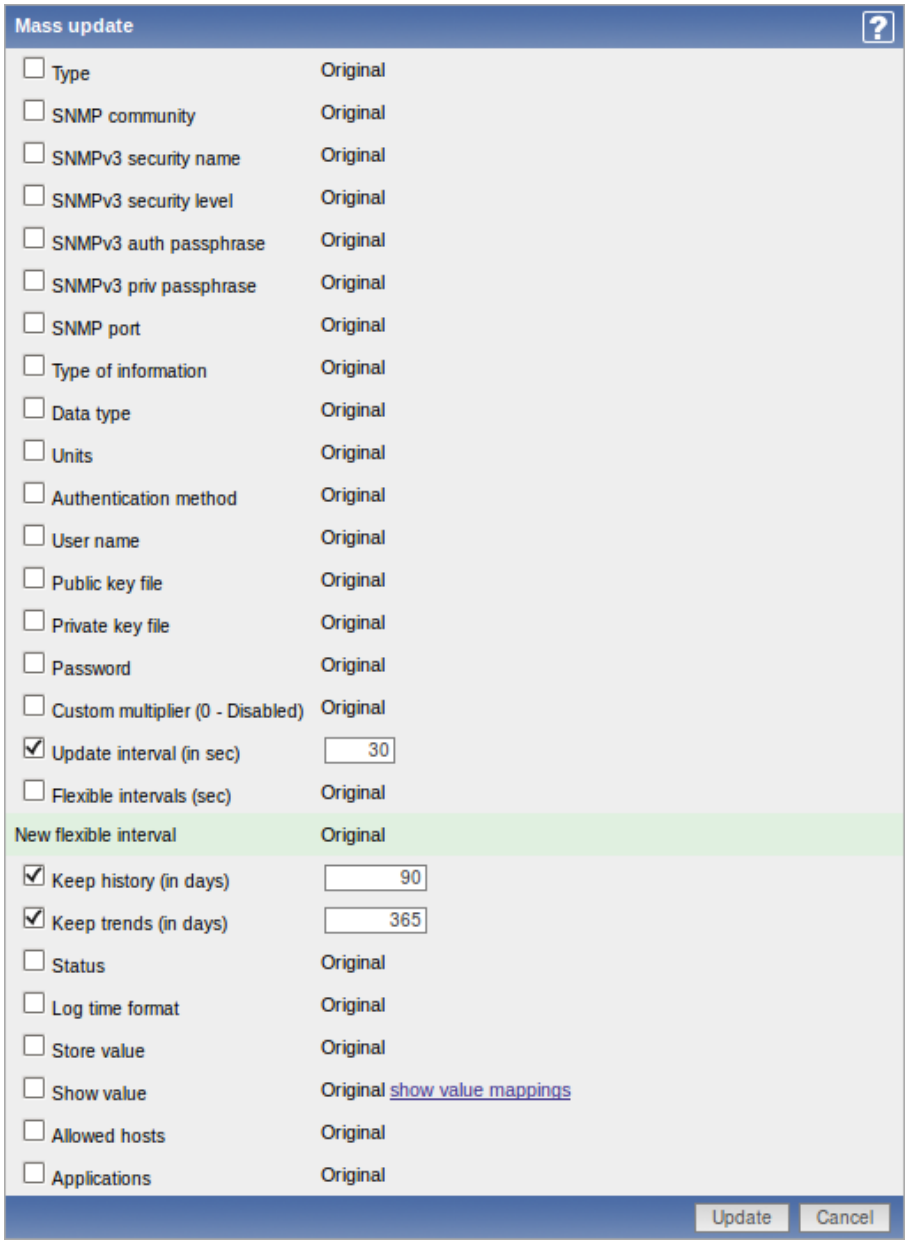
- ms
- RPM
- rpm
- %

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

Mass-updating items

Mass update is a very effective way of changing attributes for a number of items at once.

To update some items, check them in the item list, then select ”Mass update” in the dropdown below the item list and click on ”Go”. A form is displayed where you can select what attributes for the items you want to update.

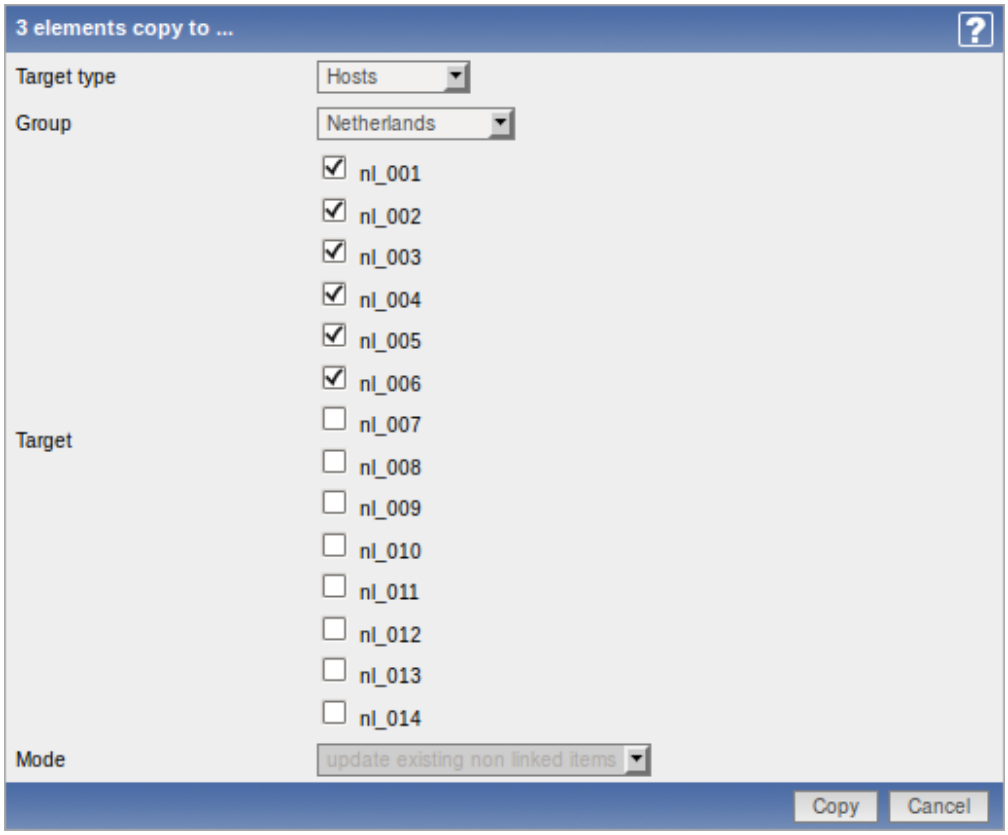



Check any parameter that you would like to change, enter a new value for it and press ”Save”.

Copy selected to...

The function makes it possible to copy a selected item to a number of hosts.

To do so, mark the item in the list, then select ”Copy selected to...” in the dropdown below the list and click on ”Go”. A form is displayed where you can select the hosts to copy items to.




Select the hosts you would like to copy the items to and press ”Copy”.

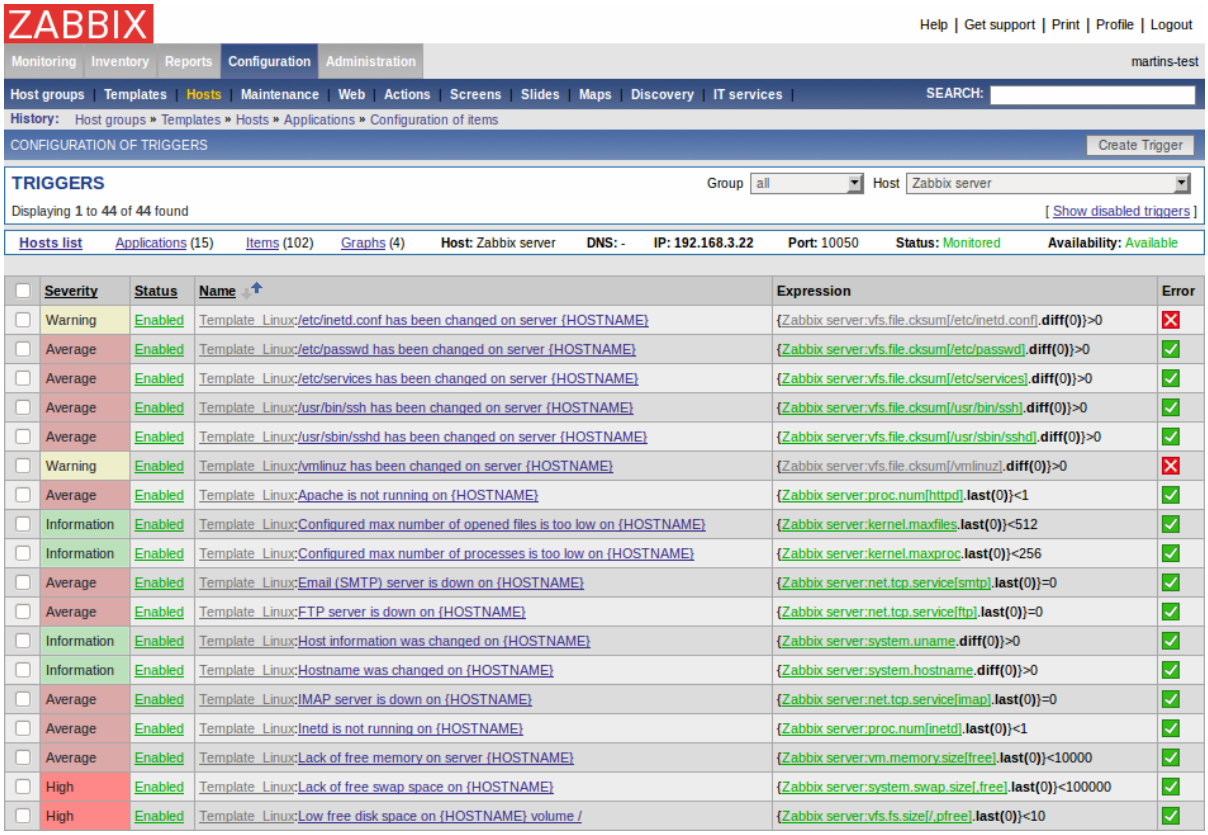
2.3.3 Triggers

*Configuration → Hosts*
*Configuration → Templates*

From the list of hosts or templates you can access the triggers that are linked to a host or template by clicking in the *Triggers* column.

On the triggers screen you can view and manage triggers.

A list of existing triggers is displayed.

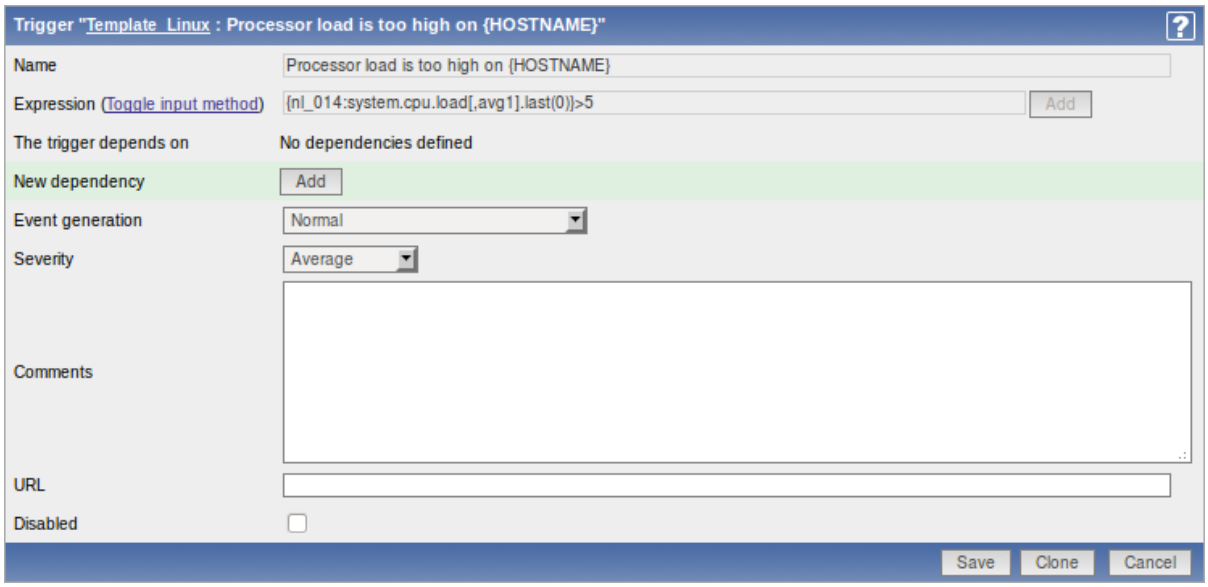



Displayed data:

| Parameter | Description |
|---|---|
| **Severity** | Coloured trigger severity. |
| **Status** | Trigger status. Note that Disabled triggers are hidden by default. |
| **Name** | Trigger name. |
| **Expression** | Trigger expression. |

Click on *Create Trigger* in the upper right corner of the screen if you wish to add a trigger. If you wish to edit an existing trigger, click on its name in the list. A form is displayed where you can edit details of a trigger.

Configuring a trigger




You can also create a new trigger from the existing one by pressing the *Clone* button and then saving under a different name.

Trigger attributes:

| Parameter | Description |
|---|---|
| **Name** | Trigger name. The name may contain macros. |
| **Expression** | Logical expression used for calculation of trigger state. |
| **The trigger depends on** | List of triggers the trigger depends on. |
| **New dependency** | Add new dependency. |
| **Event generation** | **Normal** – events are generated normally, on trigger status change<br>**Normal + Multiple PROBLEM events** (**Multiple TRUE events in 1.8.2 and before**) – events are also generated on every PROBLEM evaluation of the trigger |
| **Severity** | Trigger severity. |
| **Comments** | Text field used to provide more information about this trigger. May contain instructions for fixing specific problem, contact detail of responsible staff, etc. |
| **URL** | If not empty, the URL is used in the screen 'Status of Triggers'. |
| **Disabled** | Trigger can be disabled if required. |

See also more information about triggers.

**Mass-updating triggers**

Mass update is a very effective way of changing attributes for a number of triggers at once.

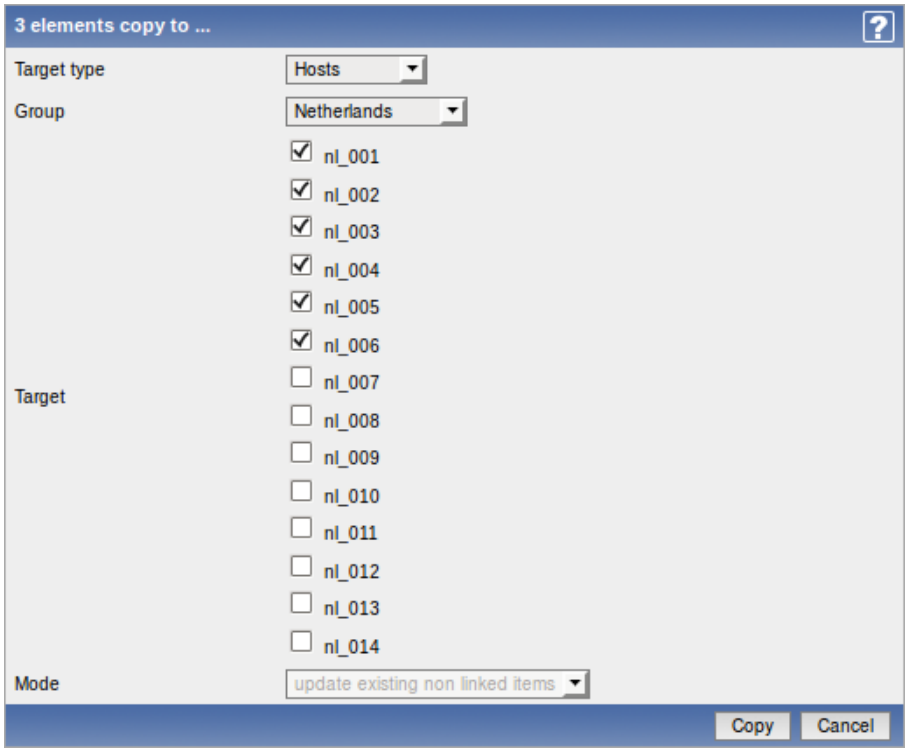
To update some triggers, check them in the list, then select "Mass update" in the dropdown below the trigger list and click on "Go". A form is displayed where you can select what attributes for the triggers you want to update.


Check any parameter that you would like to change, enter a new value for it and press "Save".

**Copy selected to...**

The function makes it possible to copy a selected trigger to a number of hosts.

To do so, mark the trigger in the list, then select "Copy selected to..." in the dropdown below the list and click on "Go". A form is displayed where you can select the hosts to copy triggers to.




Select the hosts you would like to copy triggers to and press ”Copy”.

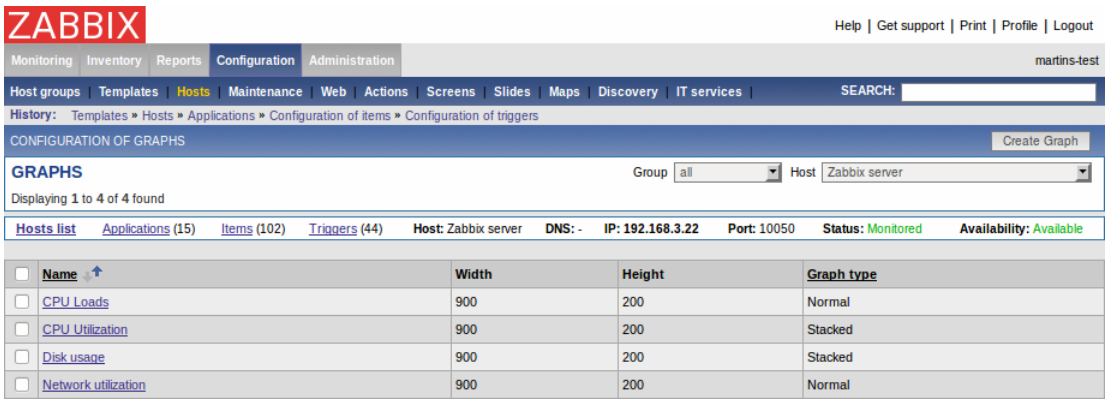
2.3.4 Graphs

*Configuration → Hosts*
*Configuration → Templates*

From the list of hosts or templates you can access the graphs that are linked to a host or template by clicking in the *Graphs* column.

On the graphs screen you can manage custom graphs.

A list of existing custom graphs is displayed.

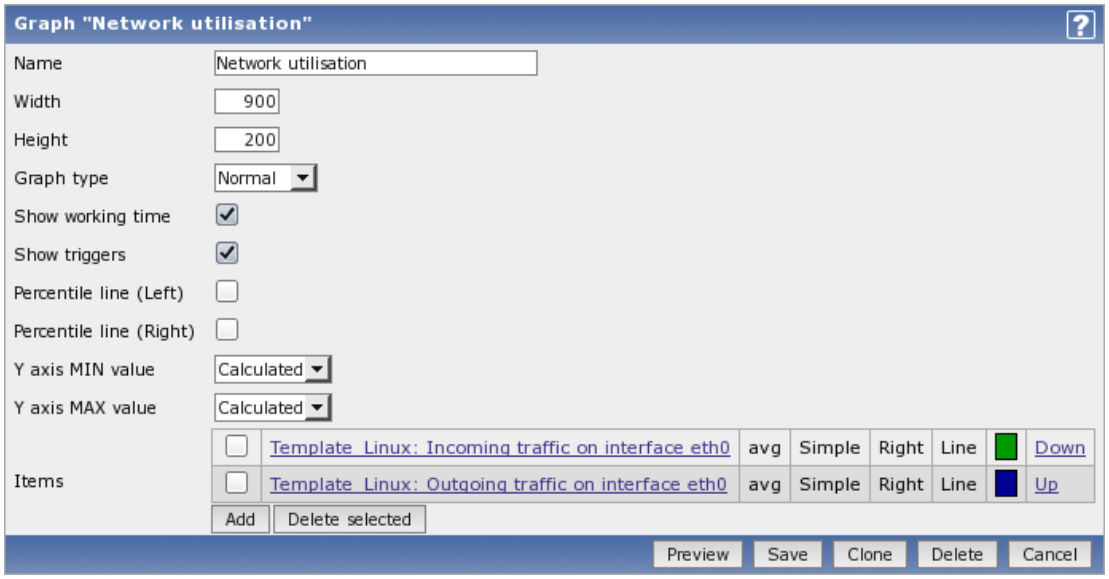



Displayed data:

| Parameter | Description |
|---|---|
| **Name** | Graph name. |
| **Width** | Graph width in pixels. |
| **Height** | Graph height in pixels. |
| **Graph type** | Graph type:<br>**Normal**<br>**Stacked**<br>**Pie**<br>**Exploded** |

Configuring a graph

Click on *Create Graph* in the upper right corner of the screen in the graph list if you wish to add a graph. If you wish to edit an existing graph, click on its name in the list. A form is displayed where you can configure a graph.

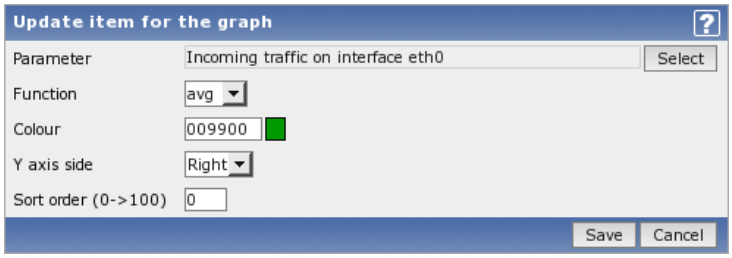



You can also create a new graph from the existing one by pressing the *Clone* button and then saving under a different name.

Graph attributes:

| Parameter | Description |
|---|---|
| **Name** | Unique graph name. |
| **Width** | Graph width in pixels. |
| **Height** | Graph height in pixels. |
| **Graph type** | Graph type:<br>**Normal** – normal graph, values displayed as lines.<br>**Stacked** – stacked graph.<br>**Pie** – pie graphs.<br>**Exploded** – exploded pie graph. |
| **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. Normally used for displaying 95% percentile. Only available for normal graphs. |
| **Percentile line (Right)** | Display percentile for right Y axis. Normally used for displaying 95% percentile. Only available for normal graphs. |
| **Y axis MIN value** | Type of Y axis:<br>**Calculated** – Y axis value will be automatically calculated<br>**Calculated [min=0]** – Y min value is set to 0, maximum value will be automatically calculated.<br>**Fixed** – fixed min and max value for Y axis. Not available for pie and exploded pie graphs. |
| **Y axis MAX value** | Type of Y axis:<br>**Calculated** – Y axis value will be automatically calculated<br>**Calculated [min=0]** – Y min value is set to 0, maximum value will be automatically calculated.<br>**Fixed** – fixed min and max value for Y axis. Not available for pie and exploded pie graphs. |
| **3D view** | Enable 3D style. For pie and exploded pie graphs only. |
| **Legend** | Display legend. For pie and exploded pie graphs only. |
| **Items** | List of graph elements (items) to be displayed for this graph. |

Graph element:




Attributes of a graph element:

| Parameter | Description |
|---|---|
| **Parameter** | Selection of host item, which will be displayed. |
| **Type** | Type (only available for normal graphs):<br>Simple<br>Aggregated |
| **Function** | What values will be displayed when more than one value exists for a single pixel (X-coordinate):<br>**all** – all (minimum, average and maximum)<br>**min** – minimum only<br>**avg** – average only<br>**max** – maximum only |
| **Draw style** | Draw style (only available for normal graphs; for stacked graphs filled region is always used):<br>**Line** – draw lines<br>**Filled region** – draw filled region<br>**Bold line** – draw bold lines<br>**Dot** – draw dots<br>**Dashed line** – draw dashed line |
| **Colour** | RGB colour in HEX notation. |
| **Aggregated periods count** | |
| **Y axis side** | Which Y axis side the element is assigned to. |
| **Sort order (0→100)** | Draw order, 0 will be processed first. |

Below the graph preview is displayed. Note that it will not show any data for template items.

**Note:**
3 triggers is the hard-coded limit for the number of triggers displayed in the graph legend.

2.3.5 Template linkage

**Warning:**
Starting with Zabbix 1.8, template linkage to hosts can be managed in Configuration→Templates.

2.3.6 Proxies

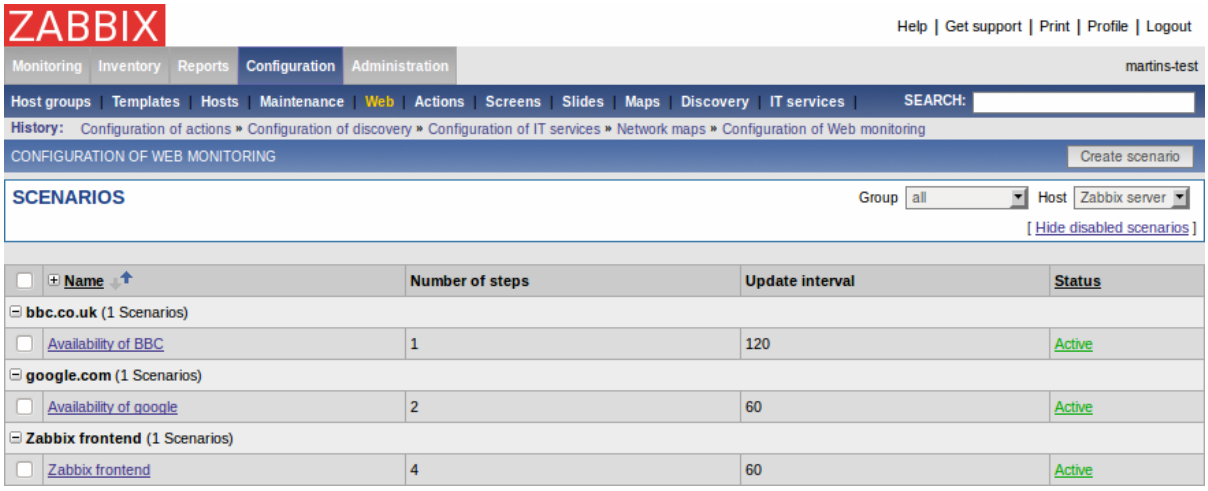
**Warning:**
Starting with Zabbix 1.8, proxy management is available at Administration→DM.

2.4 Web monitoring

*Configuration → Web*

On this screen you can manage scenarios for web monitoring.

A list of the active scenarios is displayed.

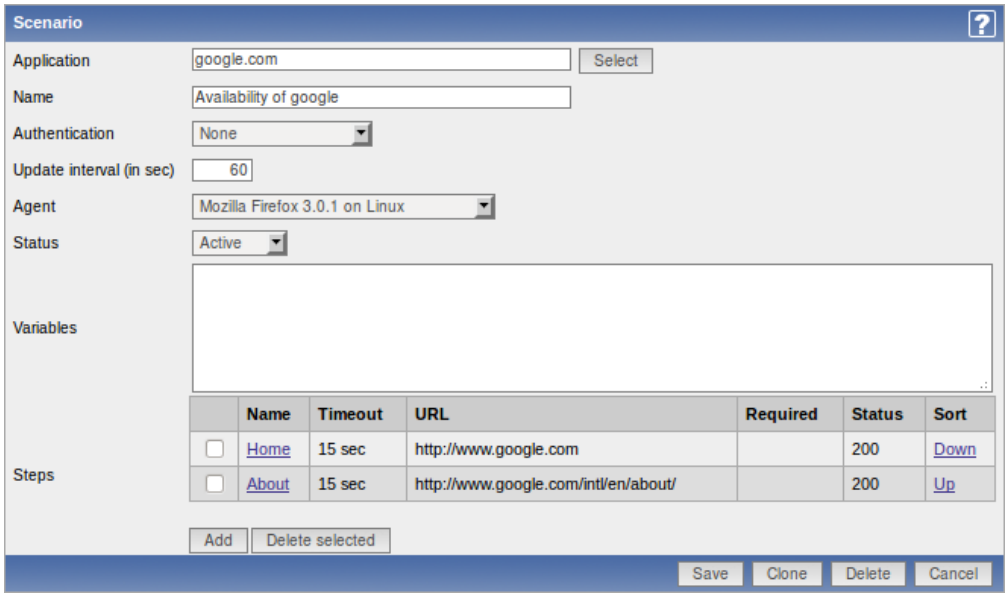



Displayed data:

| Parameter | Description |
|---|---|
| **Name** | Unique name of the web scenario. |
| **Number of steps** | Number of individual steps (HTTP requests) the scenario consists of. |
| **Update interval** | Frequency of execution of the web scenario. |
| **Status** | Status of the scenario:<br>**Active** - the scenario is active<br>**Disabled** - the scenario is disabled. Note that disabled scenarios are not displayed by default in the main screen. |

Click on *Create scenario* in the upper right corner of the screen if you wish to add a scenario. If you wish to edit an existing scenario, click on its name in the list. A form is displayed where you can edit the parameters of a web scenario.

Configuring a web scenario



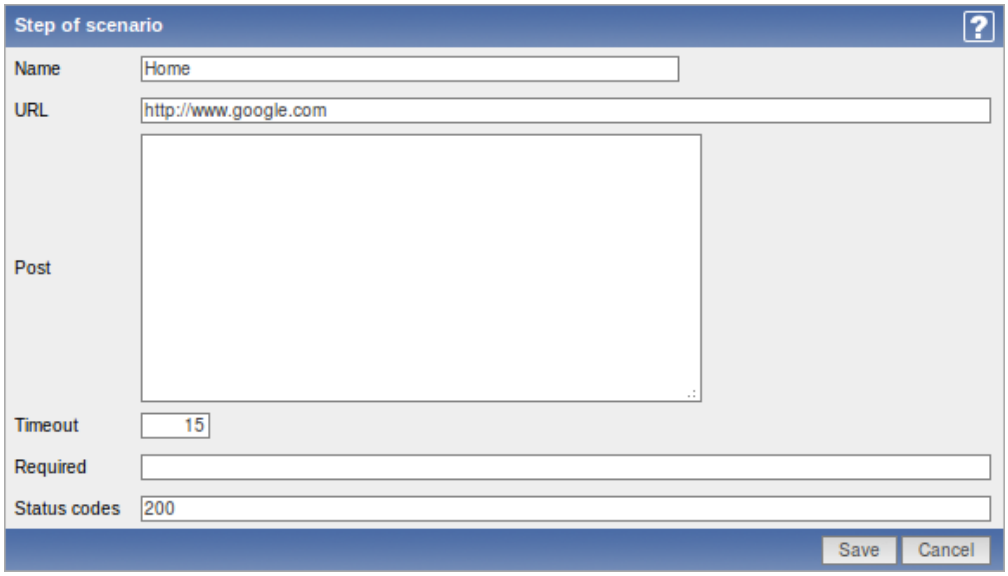

Configuration parameters:

| Parameter | Description |
|---|---|
| **Application** | Host application the scenario is linked to. |
| **Name** | Unique name of the web scenario. |
| **Update interval (in sec)** | Frequency of execution of the web scenario. |
| **Agent** | Client agent string. Zabbix will pretend that it is Firefox, MS Explorer or any other application.<br>Useful when a website returns different content for different browsers. |

| Parameter | Description |
|---|---|
| **Status** | Status of the scenario:<br>**Active** - the scenario is active<br>**Disabled** - the scenario is disabled. Note that disabled scenarios are not displayed by default in the main screen. |
| **Variables** | List of variables (macros) to be used in scenario steps (URL and Post variables).<br>They have the following format:<br>**{macro1}**=value1<br>**{macro2}**=value2<br>For example:<br>username=Alexei<br>password=kj3h5kJ34bd<br>The macros can be referenced as {username} and {password}.<br>Zabbix will automatically replace them with actual values. |
| **Steps** | List of steps executed by the scenario, displaying:<br>**Name** - step name<br>**Timeout** - timeout<br>**URL** - location to connect to<br>**Required** - required string<br>**Status** - step status |

To add a web scenario step, click on the *Add* button next to Steps. A form will open where you can define the parameters of an individual web scenario step.

Configuring a web scenario step




Configuration parameters:

| Parameter | Description |
|---|---|
| **Name** | Unique step name. |
| **URL** | URL to connect to and retrieve data. For example:<br>http://www.zabbix.com<br>https://www.google.com |
| **Post** | List of POST variables. GET variables can be passed in the URL parameter. |
| **Timeout** | Zabbix will not spend more than the set amount of seconds on processing the URL. |
| **Required** | Required string. Retrieved content (HTML) must contain this string, otherwise the step will fail.<br>If empty, no check is performed. |

| Parameter | Description |
|---|---|
| **Status codes** | List of expected HTTP codes. If Zabbix gets a code which is not in the list, the step will fail.<br>If empty, no check is performed.<br>For example: 200,201,210-299 |

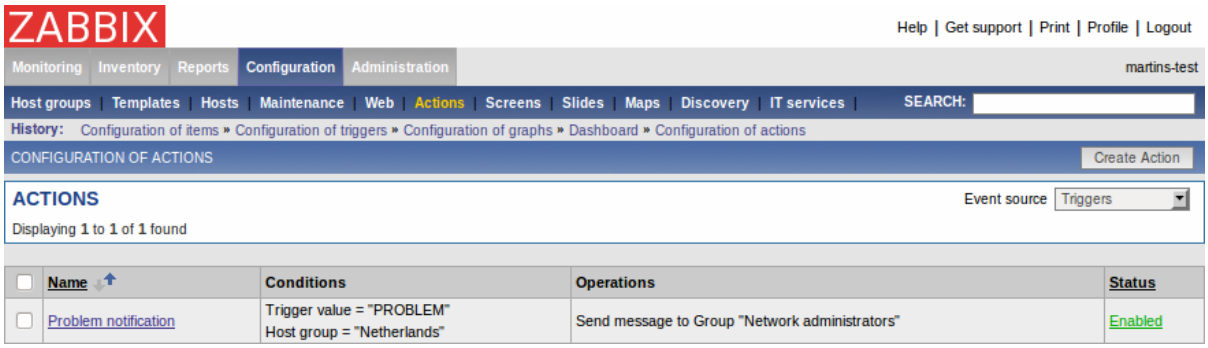
See also this section for more information about web monitoring.

2.5 Actions

*Configuration → Actions*

On this screen you can set up and manage actions.

A list of existing actions is displayed. Actions are displayed by the event source: Triggers/Discovery/Auto registration. Use the dropdown menu in the upper right corner to switch between various sources.

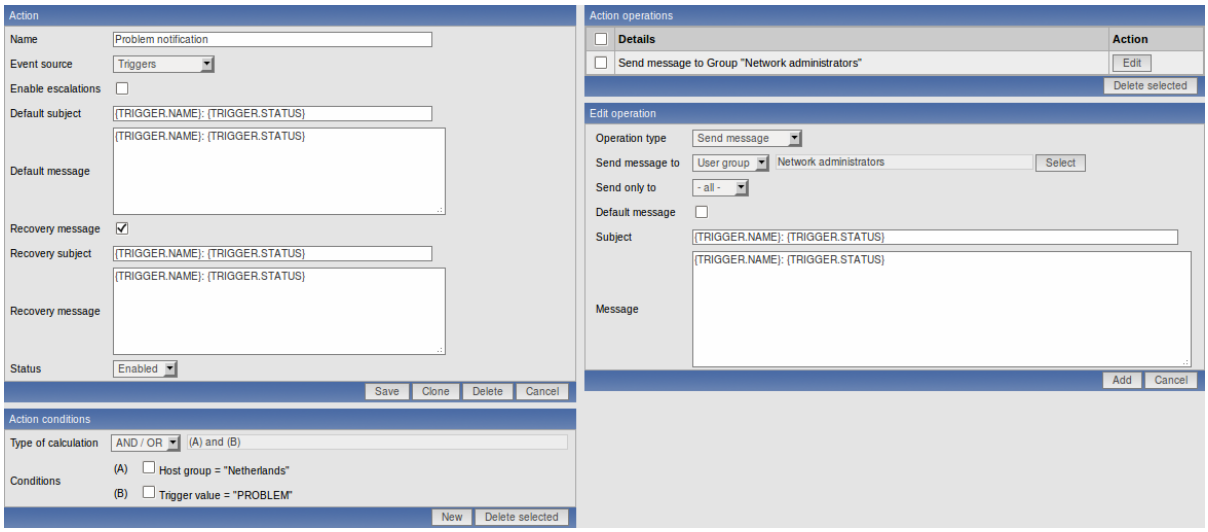



Displayed data:

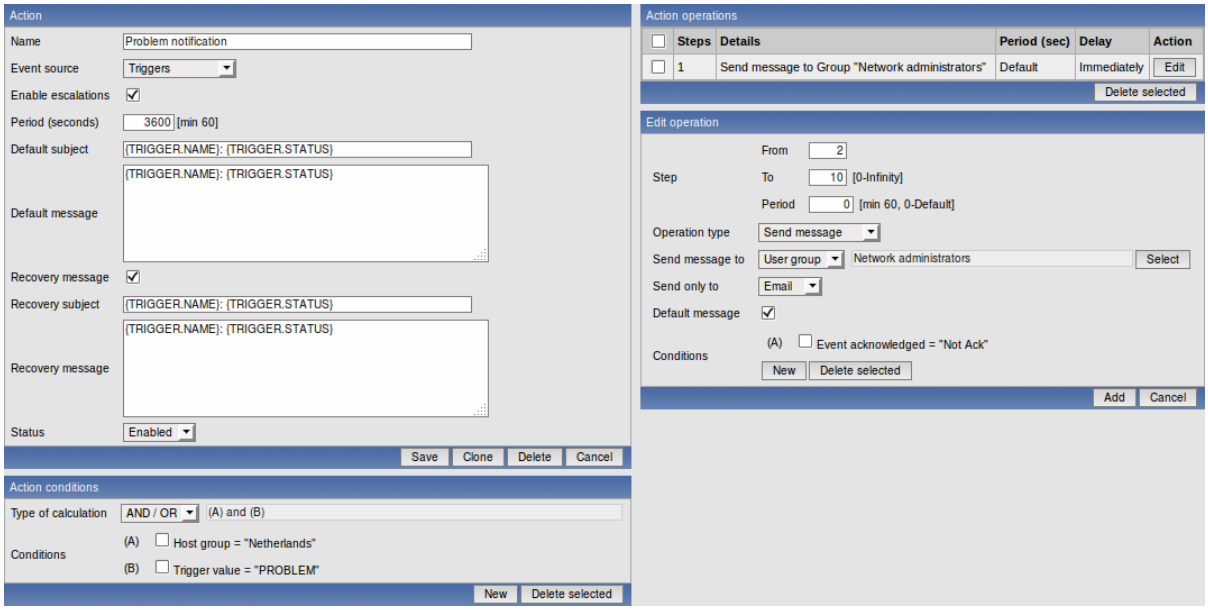
| Parameter | Description |
|---|---|
| **Name** | Action name. |
| **Conditions** | List of conditions for this action. |
| **Operations** | List of operations for execution. |
| **Status** | Status of the action. |

Click on *Create Action* in the upper right corner of the screen if you wish to add an action. If you wish to edit an existing action, click on its name in the host list. A form is displayed where you can edit details of an action.

Configuring an action




More configuration options are available if escalation is enabled:




See more details about configuration of actions, conditions and operations in other sections of the Manual.

2.6 Screens

*Configuration → Screens*

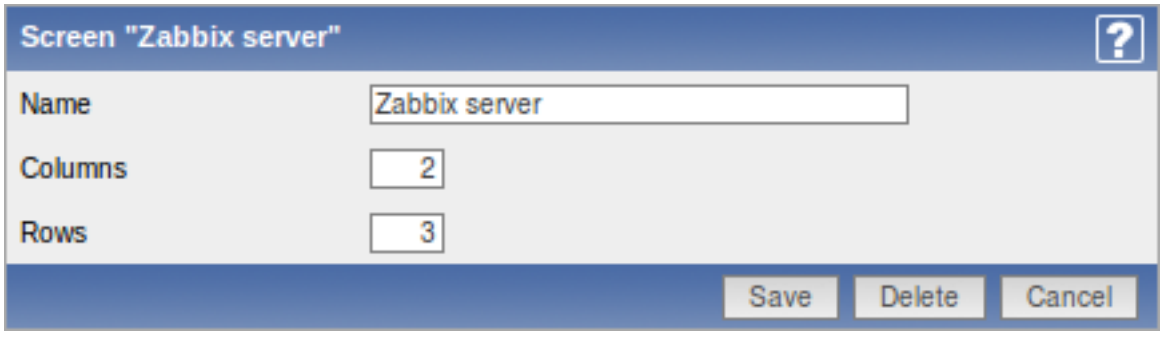
On this screen you can set up and manage screens.

A list of existing screens is displayed.


| Parameter | Description |
|---|---|
| **Name** | Screen name. |
| **Dimension (cols x rows)** | Screen size, number of columns and rows. |

Click on *Create Screen* in the upper right corner of the screen if you wish to add a screen. If you wish to edit the elements of an existing screen, click on its name in the list. If you wish to edit high-level information of an existing screen, click on the Edit button. A form is displayed where you can edit a screen.

Configuring a screen (high-level)

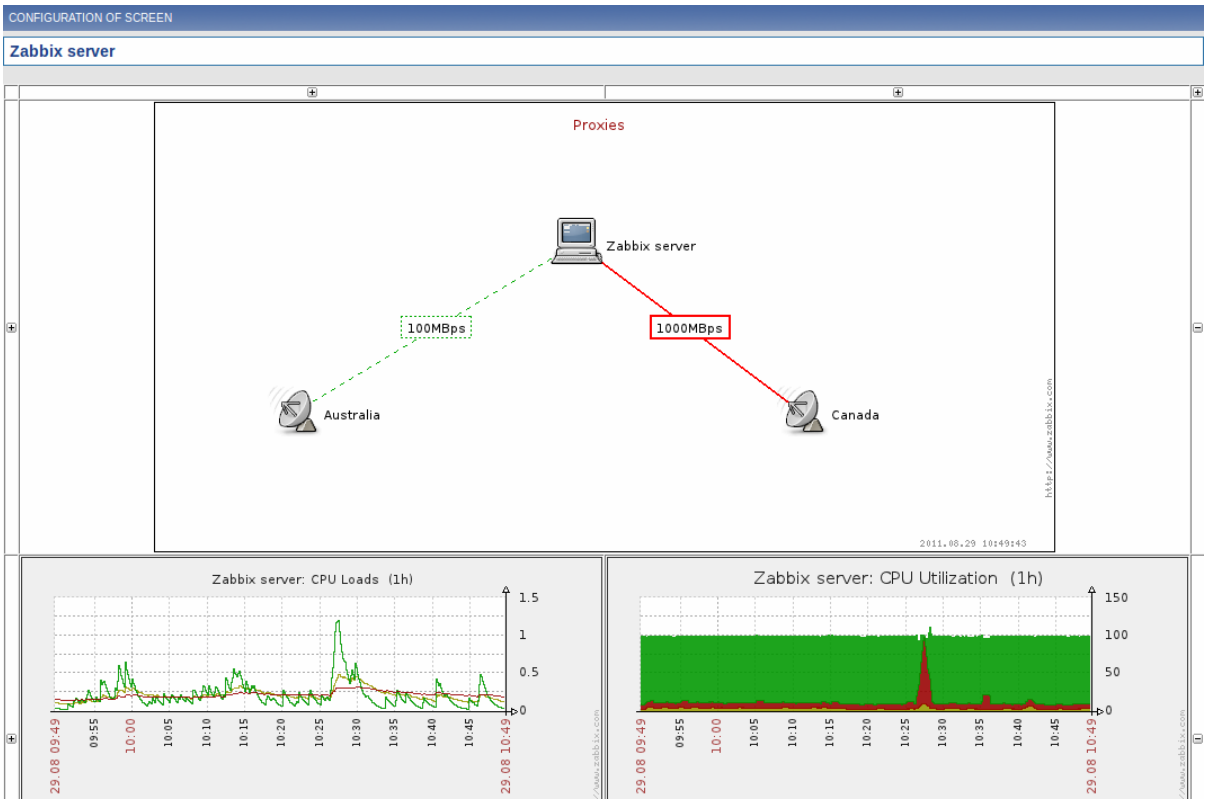



Screen high-level attributes:

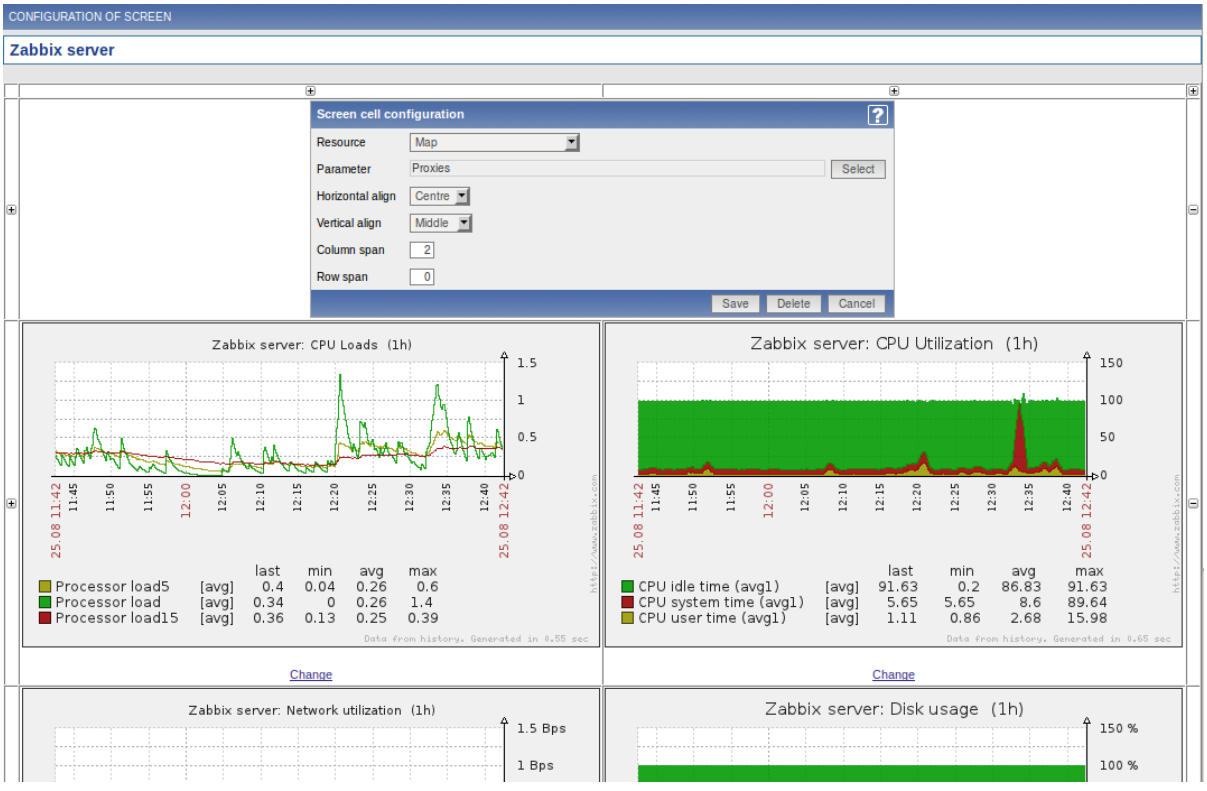
| Parameter | Description |
|---|---|
| **Name** | Unique screen name. |
| **Columns** | Number of columns in the screen. |
| **Rows** | Number of rows in the screen. |

Configuring screen elements

You can configure what will be displayed in each element (cell) of a screen.




Click on a screen element (cell) and a form will be displayed where you can edit the attributes of an element.




Screen element attributes:

| Parameter | Description |
|---|---|
| **Resource** | Information displayed in the cell:<br>**Clock** – digital or analog clock displaying current server or local time<br>**Data overview** – latest data for a group of hosts<br>**Graph** – single custom graph<br>**History of actions** – history of recent actions<br>**History of events** – latest events<br>**Hosts info** – high level host related information<br>**Map** – single map<br>**Plain text** – plain text data<br>**Screen** – screen (one screen may contain other screens inside)<br>**Server info** – server high-level information<br>**Simple graph** – single simple graph<br>**Triggers info** – high level trigger related information<br>**Triggers overview** - status of triggers for a host group<br>**URL** – include content from an external resource |
| **Horizontal align** | Possible values:<br>**Center**<br>**Left**<br>**Right** |
| **Vertical align** | Possible values:<br>**Middle**<br>**Top**<br>**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. |

**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)
- Simple graphs
- Plain text

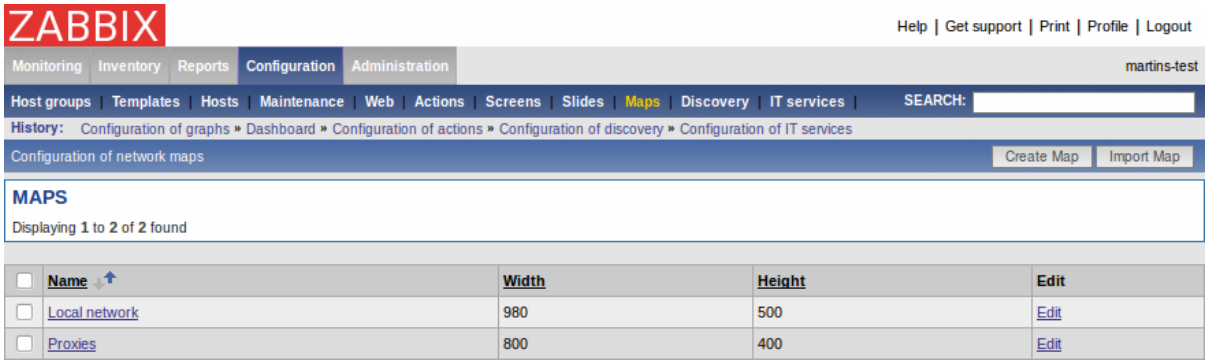
**Note:**
Clicking on a dynamic graph opens it in full view; although with custom graphs that is supported with the default host selected only.

2.7 Maps

*Configuration* → *Maps*

On this screen you can manage user-defined maps.

A list of existing maps is displayed.

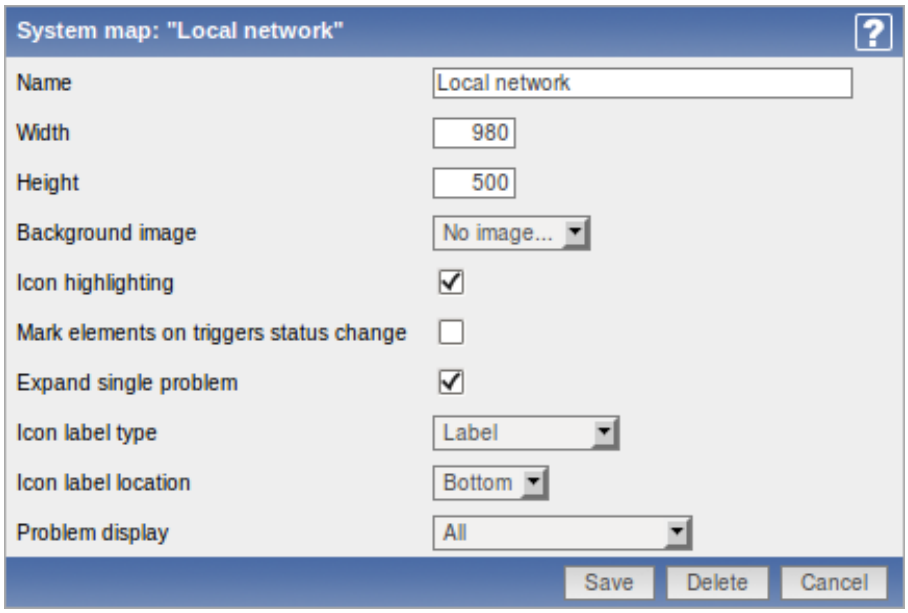



Displayed data:

| Parameter | Description |
|---|---|
| **Name** | Map name |
| **Width** | Map width in pixels. |
| **Height** | Map height in pixels. |

Click on *Create Map* in the upper right corner of the screen if you wish to add a map. If you wish to edit the elements of an existing map, click on its name in the list. If you wish to edit high-level information of an existing map, click on the Edit button. A form is displayed where you can configure a user-defined map.

Configuring a map (high-level)



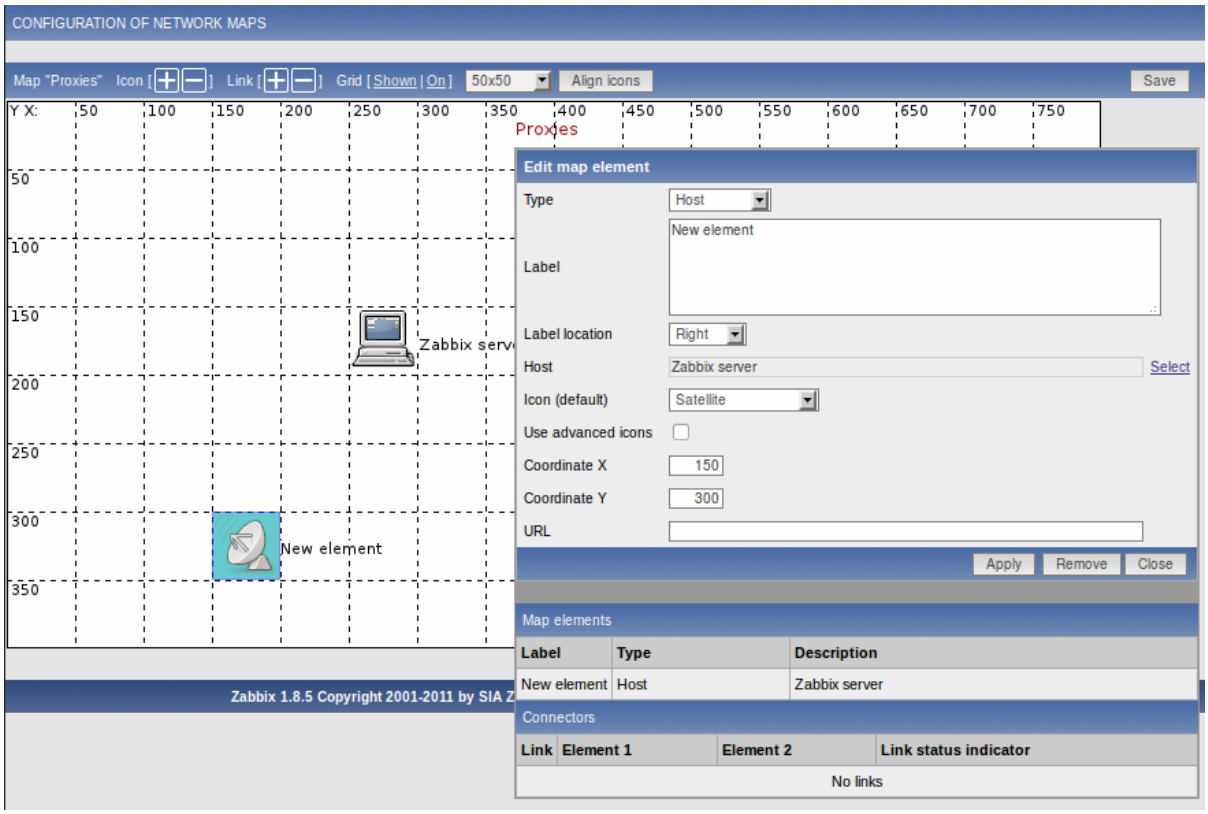

Map high-level attributes:

| Parameter | Description |
|---|---|
| **Name** | Unique map name. |
| **Width** | Map width in pixels. |
| **Height** | Map height in pixels. |
| **Background image** | Use background image:<br>**No image** - no background image (white background)<br>**Image** - selected image to be used as a background image. No scaling is performed. |
| **Icon highlighting** | Map elements will receive highlighting. If element has an active trigger, round background will be used, having same colour as the highest severity trigger. If element status is ”disabled” or ”in maintenance”, square background will be used. This option is available since Zabbix 1.8. |
| **Mark elements on trigger status change** | Elements that have a trigger status recently changed will be highlighted with markers. This option is available since Zabbix 1.8.3. |

| Parameter | Description |
|---|---|
| **Expand single problem** | If a map element (host, host group or another map) has a single problem, this option controls whether problem (trigger) name is printed, or problem count. If marked, problem name is used. This option is available since Zabbix 1.8.1. For upgrades from previous installations it is enabled by default on all maps. |
| **Icon label type** | Label type used for all map icons:<br>**Label** - icon label only<br>**IP address** - IP address only<br>**Element name** - element name (for example, host name)<br>**Status only** - status only (OK or PROBLEM)<br>**Nothing** - no icon labels are displayed |
| **Icon label location** | Display icon label on:<br>**Bottom** - bottom (under the icon)<br>**Left** - left side<br>**Right** - right side<br>**Top** - top of the icon |

Configuring a map element

To add an element to the map, click on the "+" next to Icon. The new element will appear at the top left corner of the map. Now you can drag and drop it on a desired place on the map. By clicking on the element, a form is displayed where you can edit the attributes of the element - its type, label, icon type etc.



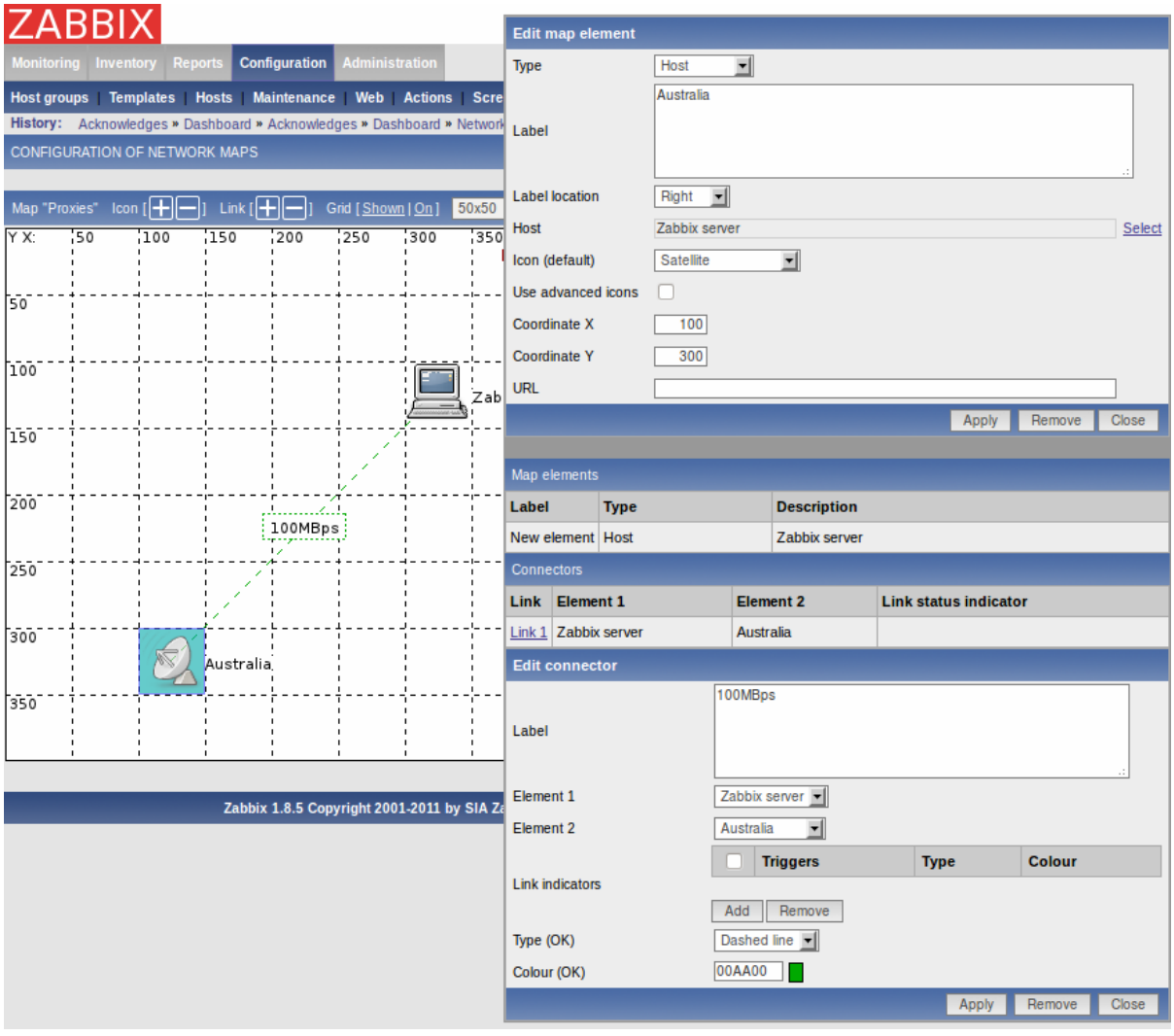

Map element attributes:

| Parameter | Description |
|---|---|
| **Type** | Type of the element:<br>**Host** - icon representing status of all triggers of the selected host<br>**Map** - icon representing status of all elements of a map<br>**Trigger** - icon representing status of a single trigger<br>**Host group** - icon representing status of all triggers of all hosts belonging to<br>**Image** - an icon, not linked to any resource |

| Parameter | Description |
|---|---|
| **Label** | Icon label, any string.<br>Macros and multi-line string can be used in labels starting from version 1.8 |
| **Label location** | Label location:<br>**Default** - Map's default label location<br>**Bottom** - bottom (under the icon)<br>**Left** - left side<br>**Right** - right side<br>**Top** - top of the icon |
| **Host** | Status of triggers for the selected host will be used. |
| **Map** | Status of all elements for the selected map will be used. |
| **Trigger** | Status of the selected trigger will be used. |
| **Host group** | Status of all triggers for the selected host group will be used. |
| **Icon (ok)** | Icon to be used when no problem exists. |
| **Icon (problem)** | Icon to be used in case of problems (one or more). |
| **Icon (unknown)** | Icon to be used if the selected host is in an unknown state. |
| **Icon (disabled)** | Icon to be used if the selected host is disabled. |
| **Coordinate X** | X coordinate for the map element. |
| **Coordinate Y** | Y coordinate for the map element. |
| **URL** | If set, the URL will be used when a user clicks on the screen element. |

#### Configuring a link

To link two elements in the map, select them both and click on the "+" next to Link. A form is displayed where you can click on the respective link and edit its attributes.




Map link attributes:

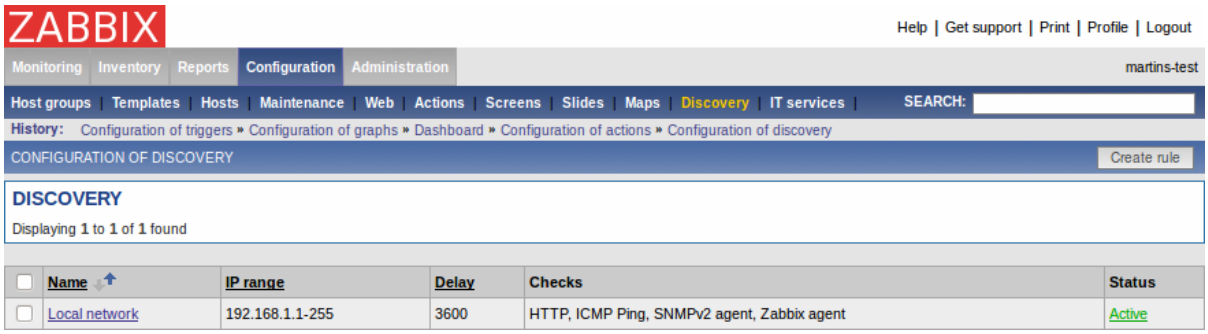
| Parameter | Description |
|---|---|
| **Label** | Label that will be rendered on top of the link. You can use macros here. |
| **Element 1** | First element that link connects. |
| **Element 2** | Second element that link connects. |
| **Link status indicators** | List of triggers linked to the link. In case if a trigger has status PROBLEM, its style is applied to the link. |
| **Type (OK)** | Default link style:<br>**Line** - single line<br>**Bold line** - bold line<br>**Dot** - dots<br>**Dashed line** - dashed line |
| **Colour (OK)** | Default link colour. |

2.8 Discovery

*Configuration → Discovery*

On this screen you can manage discovery rules.

A list of existing discovery rules is displayed.

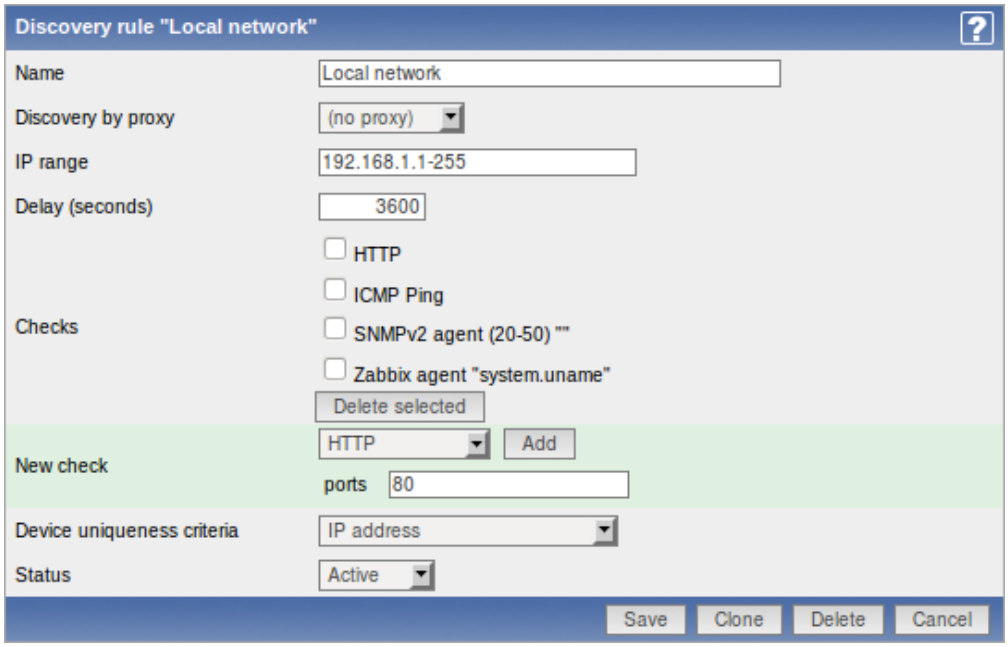



Displayed data:

| Parameter | Description |
|---|---|
| **Name** | Name of discovery rule. |
| **IP range** | Range of IP addresses affected by the discovery rule. |
| **Delay** | Frequency in seconds. |
| **Checks** | List of checks executed by the discovery rule. |
| **Status** | Status of the discovery rule:<br>**Active** - the rule is active<br>**Disabled** - the rule is disabled |

Click on *Create rule* in the upper right corner of the screen if you wish to add a discovery rule. If you wish to edit an existing discovery rule, click on its name in the list. A form is displayed where you can edit the parameters of a discovery rule.

Configuring a discovery rule




Discovery rule attributes:

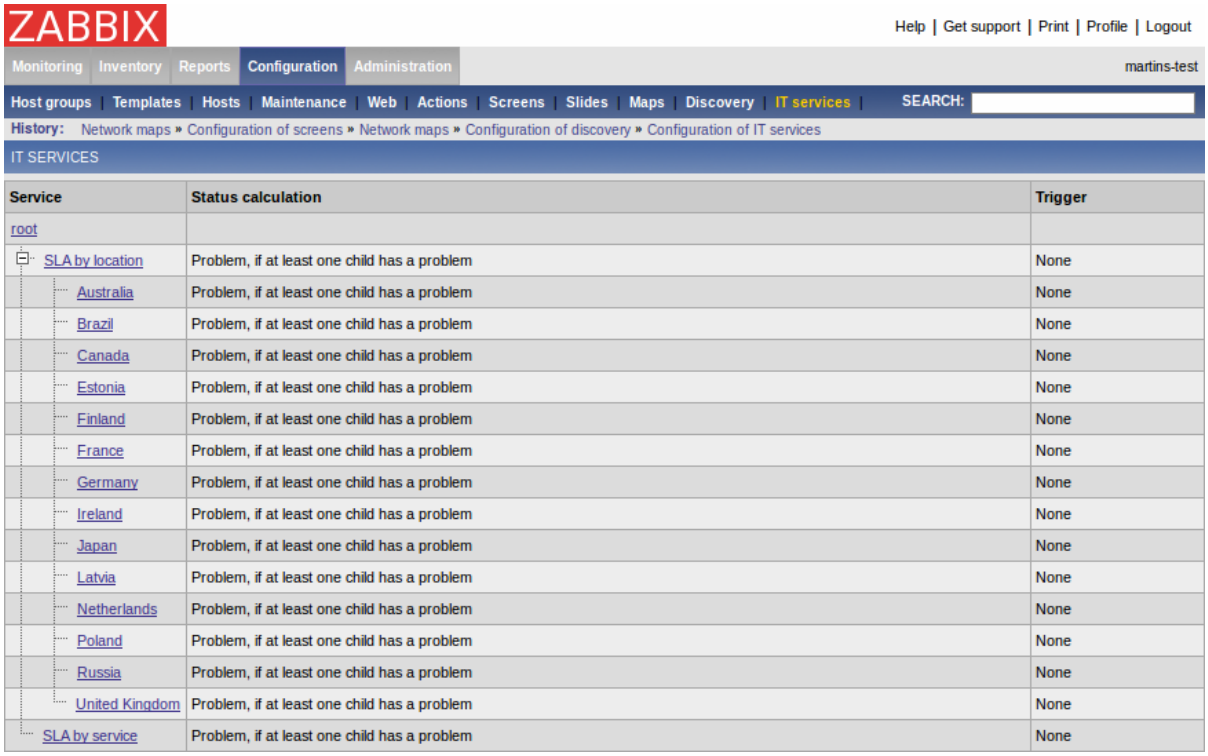
| Parameter | Description |
|---|---|
| **Name** | Unique name of the discovery rule. |
| **Discovery by proxy** | Who performs discovery:<br>**(no proxy)** - Zabbix server is doing discovery<br>**proxy name** - This proxy performs discovery |
| **IP range** | Range of IP addresses for discovery. Format:<br>Single IP: 192.168.1.33<br>Range of IP addresses: 192.168.1.1-255<br>List: 192.168.1.1-255,192.168.2.1-100,192.168.2.200<br>CIDR notation: 192.168.1.0/24 |
| **Delay (seconds)** | This parameter defines how often Zabbix should execute this rule in seconds. |
| **Checks** | List of supported checks:<br>SSH, LDAP, SMTP, FTP, HTTP, POP, NNTP, IMAP, TCP, Zabbix agent, SNMPv1 agent, SNMPv2 agent, SNMPv3 agent, ICMP ping<br><br>**Port range** may be: single port (*22*), a range of ports (*22-45*) or a list (*22-45,55,60-70*) |
| **Device uniqueness criteria** | If Zabbix will discover another device for which value, retrieved from the check that is specified as *Device uniqueness criteria*, it will be considered to be already discovered and new host will not be added |
| **Status** | Status of the discovery rule:<br>**Active** - the rule is active<br>**Disabled** - the rule is disabled |

2.9 IT Services

*Configuration → IT Services*

On this screen you can manage IT Services.

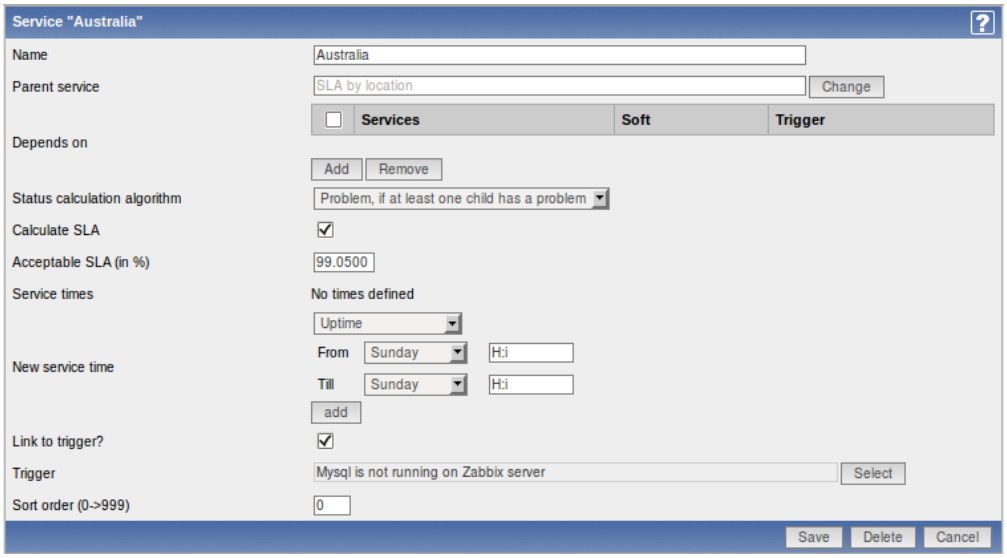
A list of existing IT Services is displayed.




Displayed data:

| Parameter | Description |
|---|---|
| **Service** | Service name. |
| **Status calculation** | How the service updates its status. |
| **Trigger** | Linked to a trigger:<br>**none** - no linkage<br>**trigger name** - linked to the trigger, thus depends on the trigger status |

Configuring an IT Service




IT Service attributes:

| Parameter | Description |
|---|---|
| **Name** | Service name. |
| **Parent service** | Parent service. For reference only, it cannot be changed. |
| **Depends on** | List of child services the service depends on. |

| Parameter | Description |
|---|---|
| **Status calculation algorithm** | How to calculate status of the service:<br>**Do not calculate** - do not calculate service status<br>**Problem, if it least one child has a problem** - considered to be a problem if at least one child service has a problem<br>**Problem, if all children have problems** - considered to be a problem if all child services have problems |
| **Calculate SLA** | Select to display SLA data. |
| **Acceptable SLA (in %)** | SLA percentage for this service. It is used for reporting. |
| **Service times** | By default, all services are expected to operate 24x7x365. Add new service times to make exceptions. |
| **New service time** | Service times:<br>**One-time downtime** - a single downtime. Service state within this period does not affect SLA.<br>**Uptime** - service uptime<br>**Downtime** - Service state within this period does not affect SLA. |
| **Link to trigger** | Services of the lowest level must be linked to triggers. |
| **Sort order** | Display sort order, lowest comes first. |

## 2.10 Export/Import

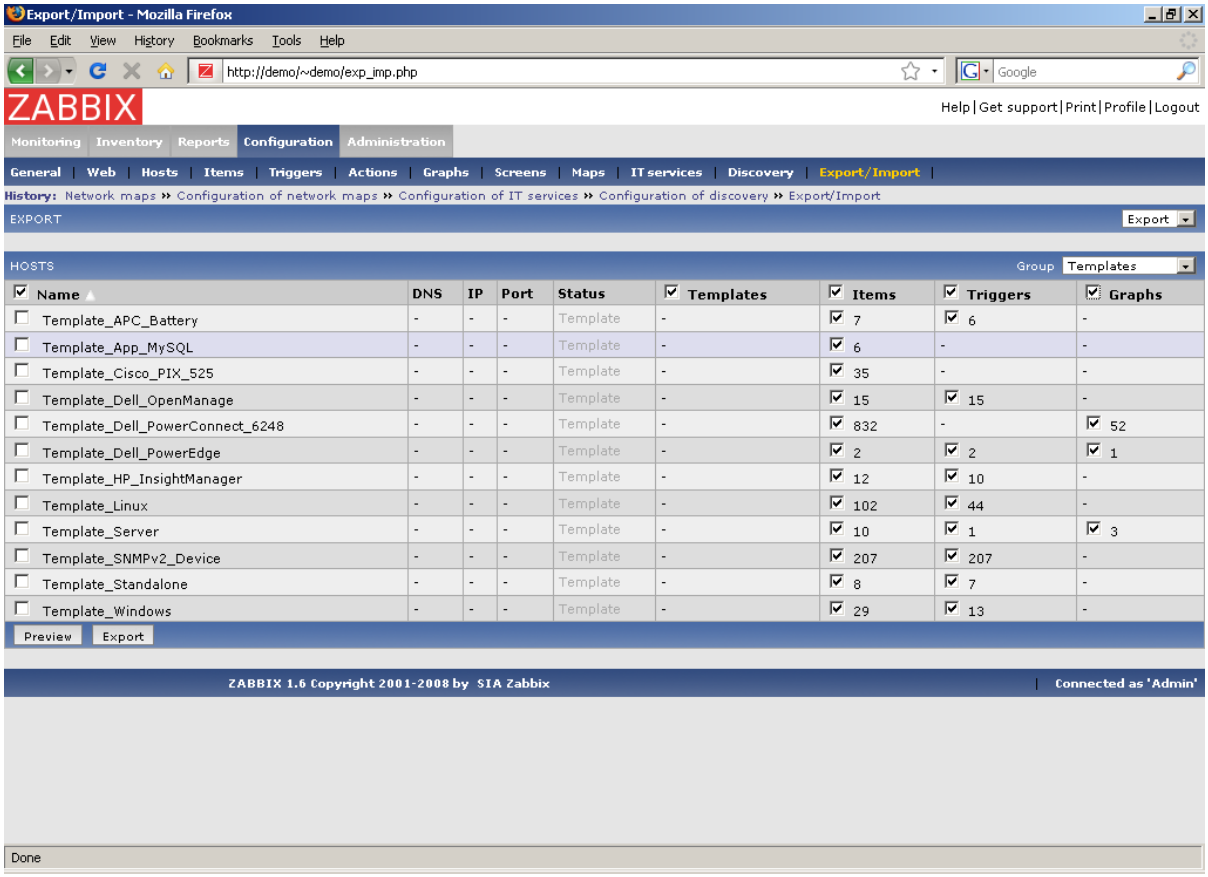
**Warning:**
Starting with Zabbix 1.8.3, this section has been removed. Import and export controls now are available in corresponding configuration pages (hosts, templates, maps or screens).

### 2.10.1 Export

The screen is used to export hosts, items, triggers and graphs.

**Export**

The screen provides list of hosts and their elements for export.

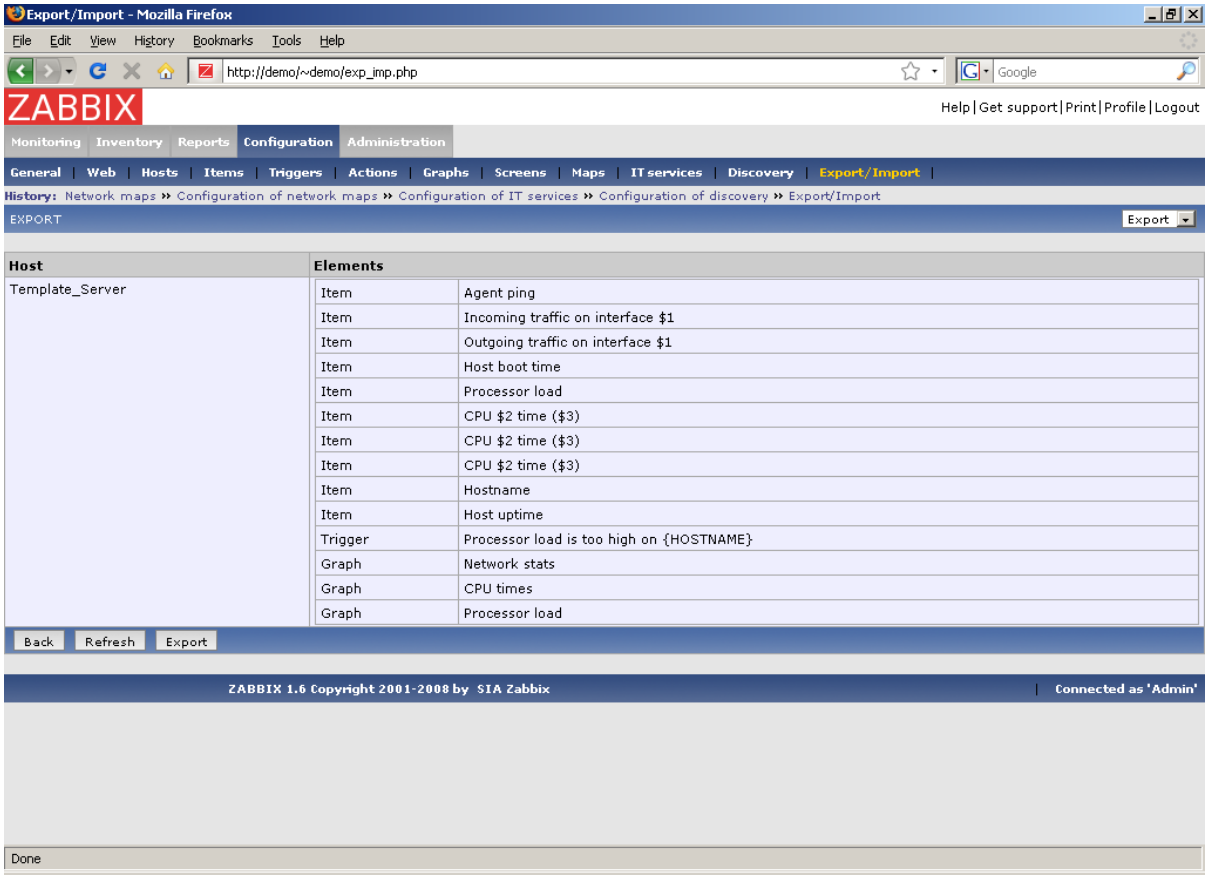



Select elements you would like to export, then press ”Preview” or ”Export”.

Displayed data:

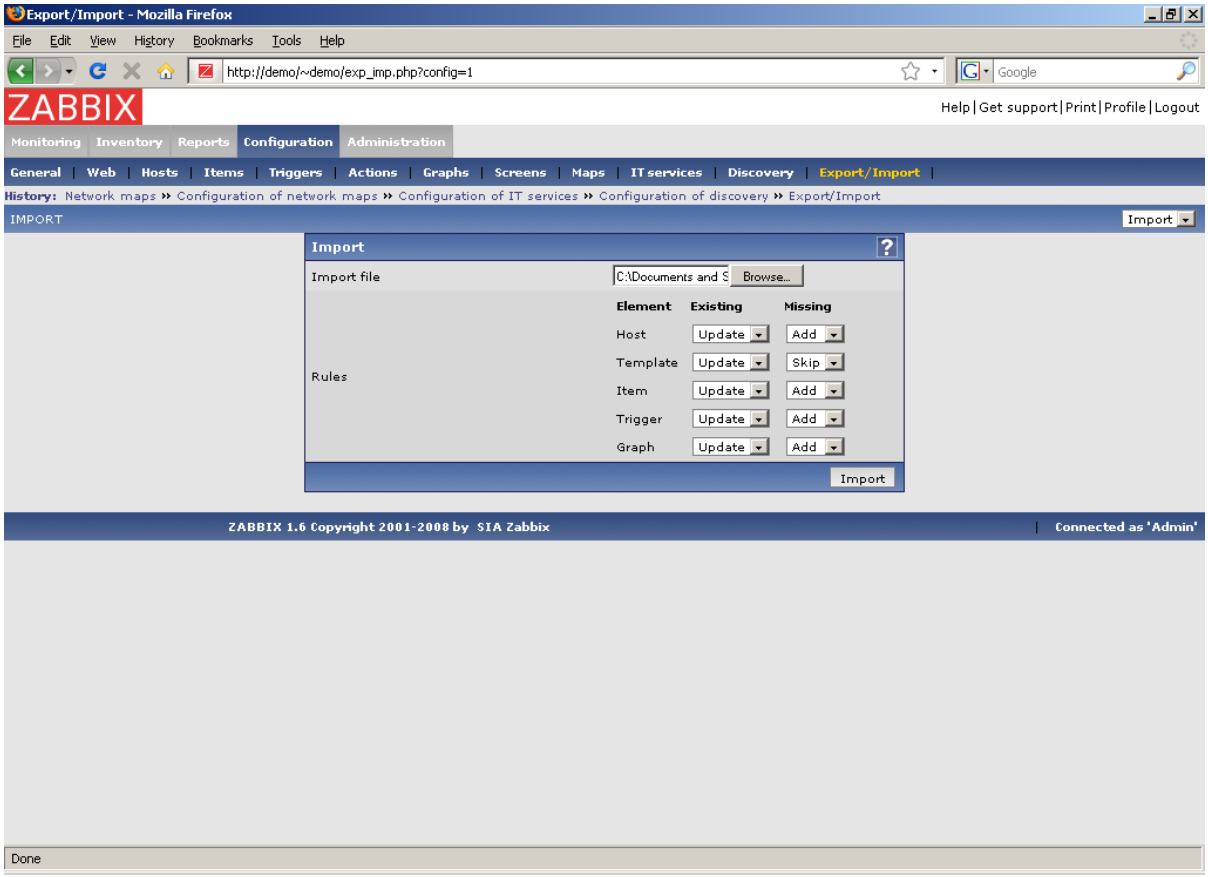
| Parameter | Description |
|---|---|
| **Name** | Host name. |
| **DNS** | Host DNS name. |
| **IP** | IP address of Zabbix agent. |
| **Port** | Zabbix agent port number. |
| **Status** | Host status. |
| **Templates** | Select to export template related information. |
| **Items** | Select to export host items. |
| **Triggers** | Select to export host triggers. |
| **Graphs** | Select to export host graphs. |

Preview page:




2.10.2 Import

The screen is used to perform XML import of host related data.




Import process options:

| Parameter | Description |
|---|---|
| Import file | XML file to import. |
| Rules | Set of rules for each type of element:<br>**Existing** - what to do if element already exists<br>**Missing** - what do to if element is missing<br>Possible actions:<br>**Update** - update existing element<br>**Add** - add element<br>**Skip** - do not process new data |

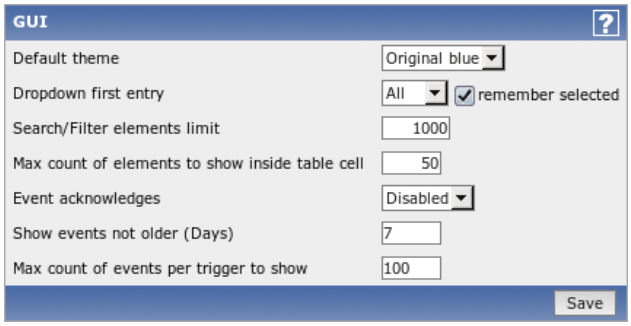
Press ”Import” to import selected file.

**3 Administration**

3.1 General

3.1.1 GUI

This section allows to set Zabbix frontend related defaults.

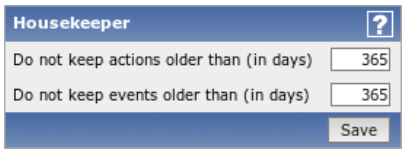



Configuration parameters:

| 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*. |
| **Search/Filter elements limit** | Maximum amount of elements that will be available as search or filter results. |
| **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. |
| **Event acknowledges** | This parameter defines if event acknowledges are activated in Zabbix interface. |
| **Show events not older (Days)** | This parameter defines for how many days event 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. |

### 3.1.2 Housekeeper

The Housekeeper is a periodical process which is executed by Zabbix Server. The process removes outdated information and information deleted by user.




Configuration parameters:

| Parameter | Description |
|---|---|
| **Do not keep actions older than (in days)** | This parameter defines how many days of executed actions (emails, jabber, SMS, etc) history Zabbix will keep in the database. Older actions will be removed. |
| **Do not keep events older than (in days)** | This parameter defines how many days of events history Zabbix will keep in the database. Older events will be removed. |

**Attention:**
To apply changes of these parameters Zabbix server has to be restarted.

### 3.1.3 Images

**List of images**

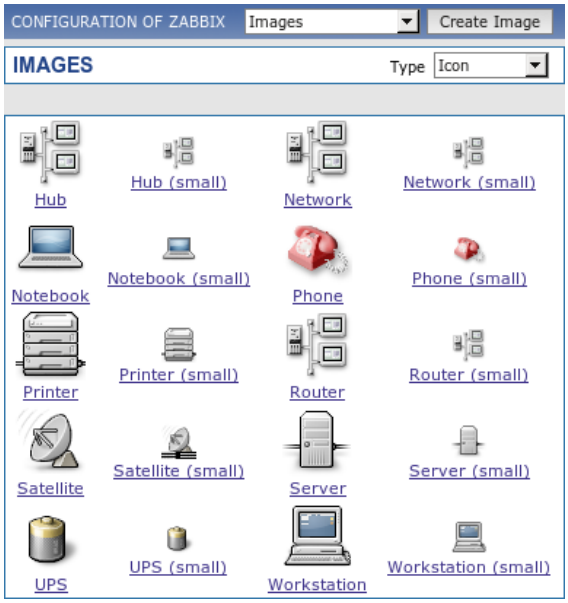
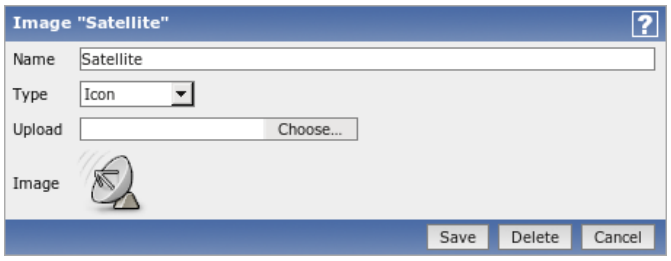


**Image definition**

Zabbix images are stored in the database. There are two types of images:

- Icon
- Background




Icons are used in for displaying System Map elements.

Backgrounds are used as background images of System Maps.

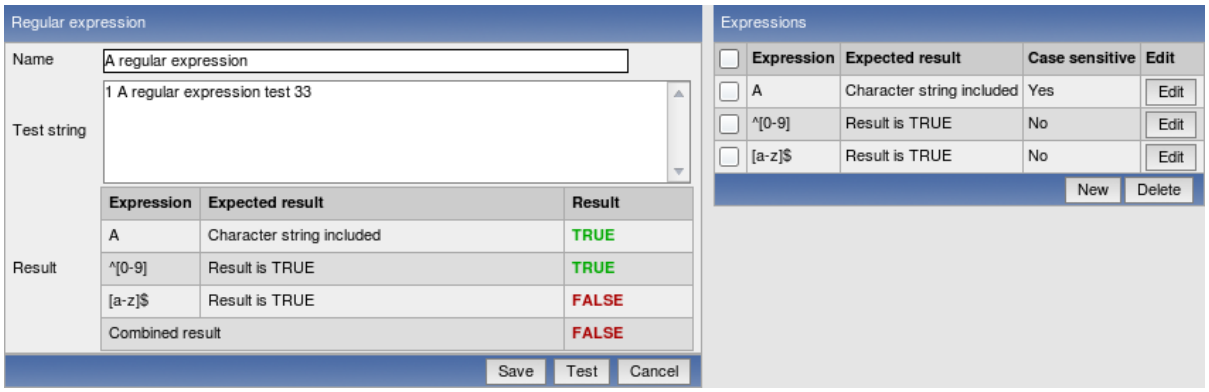
Image attributes:

| Parameter | Description |
|---|---|
| **Name** | Unique name of an image. |
| **Type** | Either **Icon** or **Background** |
| **Upload** | Name of local file (PNG, JPEG) to be uploaded to Zabbix |

**Note:**
Note that you may upload image of any size, however images bigger than 1.5MB may not be displayed in maps. Increase value of **max_memory_size** in php.ini if you have this problem.

3.1.4 Regular expressions

This section allows to create custom regular expressions for reusing elsewhere in Zabbix. A custom regular expression may consist of multiple subexpressions, and it can be tested in this section by providing a test string. Results show status of each subexpression and total custom expression status.




3.1.5 Value mapping

Value maps are used to create a mapping between numeric values and string representations.

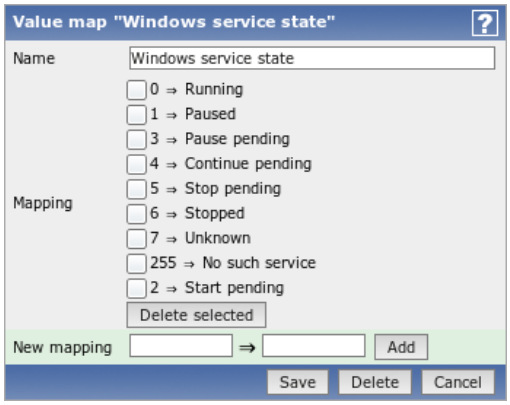
Value mappings are used for representation of data in both Zabbix front-end and information sent by email/jabber/SMS/whatever.

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'

**Note:**
Value mapping can be used only for items having type *Unsigned integer*.

**Value mapping definition**



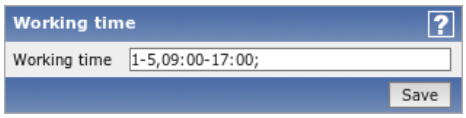

Parameters of a value mapping:

| Parameter | Description |
|---|---|
| **Name** | Unique name of set of value mappings. |
| **Mapping** | Set of mappings. |
| **New mapping** | Single mapping for addition. |

3.1.6 Working time

Working time is system-wide parameter which defines working time.

Currently this is used for graphs only. Working time is displayed as a white background, while non-working time is displayed as grey.




See Time period specification page for description of Working time format.

3.1.7 Other

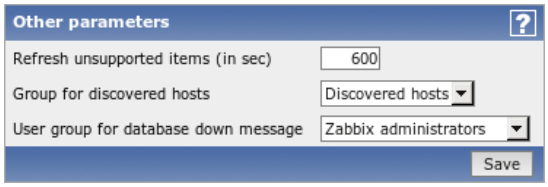
**Refresh unsupported items** Some items may become unsupported due to errors in User Parameters or because of an item being not supported by an agent.

Zabbix can be configured to periodically make unsupported items active.

**Database watchdog** Availability of Zabbix server depends on availability of back-end database. It cannot work without a database.

Database watchdog, a special Zabbix server process, is created in order to alarm Zabbix administrators in case of disaster.

The watchdog will send notifications to a user group in case if the database is down. Zabbix server will not stop; it will wait until the database is back again to continue processing.




| Parameter | Description |
|---|---|
| **Refresh unsupported items (in sec)** | Zabbix will activate unsupported item every N seconds. If set to 0, the automatic activation will be disabled.<br>Proxies check unsupported items every 10 minutes. This is not configurable for Proxies. |
| **Group for discovered hosts** | Hosts discovered by network discovery will be automatically placed in the hostgroup, selected here. |
| **User group for database down message** | User group for sending alarm message or 'None'. |

**Attention:**
Until Zabbix version 1.8.2 database watchdog is supported for MySQL only. Since 1.8.2, it is supported for all databases.

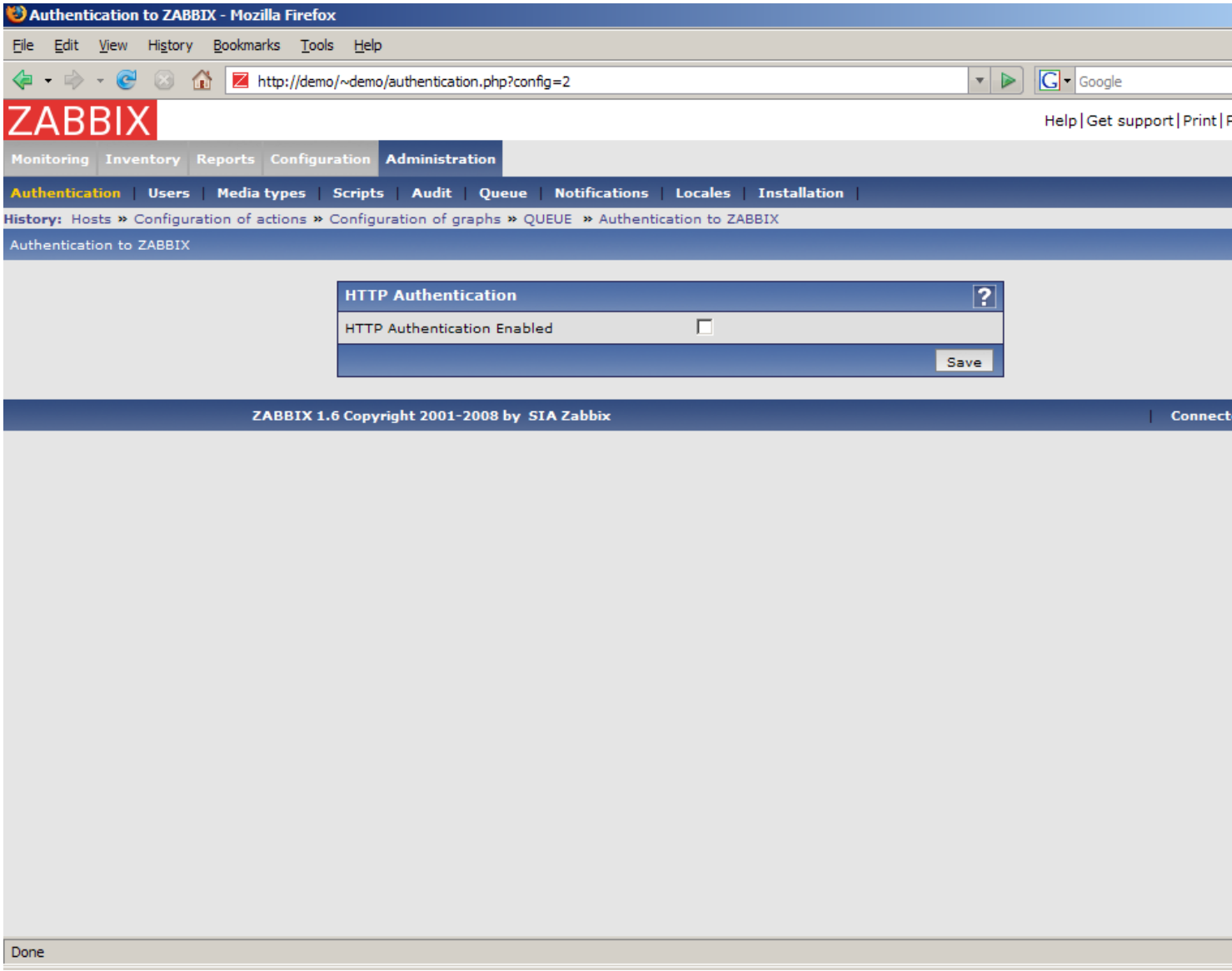
**Attention:**
To apply changes of these parameters Zabbix server has to be restarted.

The Administration Tab is available to users of type Super Administrators only.

3.2 Authentication

3.2.1 HTTP

The screen can be used to enable Apache based (HTTP) authentication. The authentication will be used to check user names and passwords. Note that an user must exist in Zabbix as well, however his Zabbix password will not be used.




Configuration parameters:

| Parameter | Description |
|---|---|
| **HTTP Authentication Enabled** | This parameter defines if Apache based authentication is enabled. |

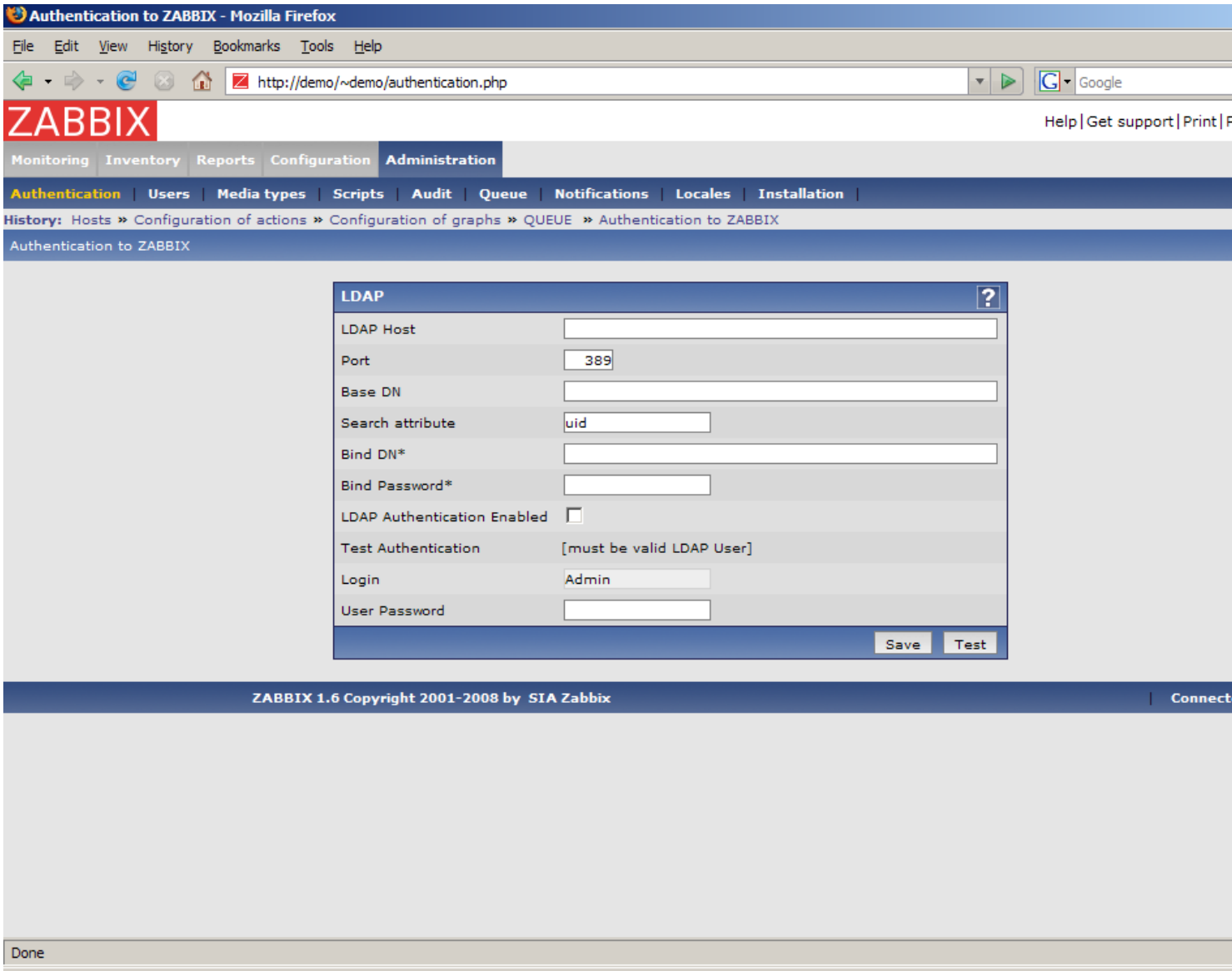
**Attention:**
Be careful! Make sure that Apache authentication is configured and works properly before switching it on.

**Note:**
In case of Apache authentication all users (even with GUI Access set to Internal) will be authorised by Apache, not by Zabbix!

3.2.2 LDAP

The screen can be used to enable external LDAP authentication. The authentication will be used to check user names and passwords. Note that an user must exist in Zabbix as well, however his 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<br>For secure LDAP server use *ldaps* protocol<br>ldaps://ldap.zabbix.com |
| **Port** | Port of LDAP server. Default is 389.<br>For secure LDAP connection port number is normally 636. |
| **Base DN** | ou=Users,ou=system |
| **Search Attribute** | uid |
| **Bind DN** | uid=Admin,ou=system |
| **Bind Password** | Password for binding to the LDAP server. |
| **LDAP Authentication Enabled** | Enable LDAP authentication. |
| **Test Authentication** | - |
| **Login** | Name of a test user. The user must exist in LDAP. |

| Parameter | Description |
|---|---|
| **User Password** | LDAP password of the test user. Zabbix will not activate LDAP authentication if it is unable to authenticate the test user. |

**Note:**
Some user groups can still be authorised by Zabbix. These groups must have GUI Access set to Internal.

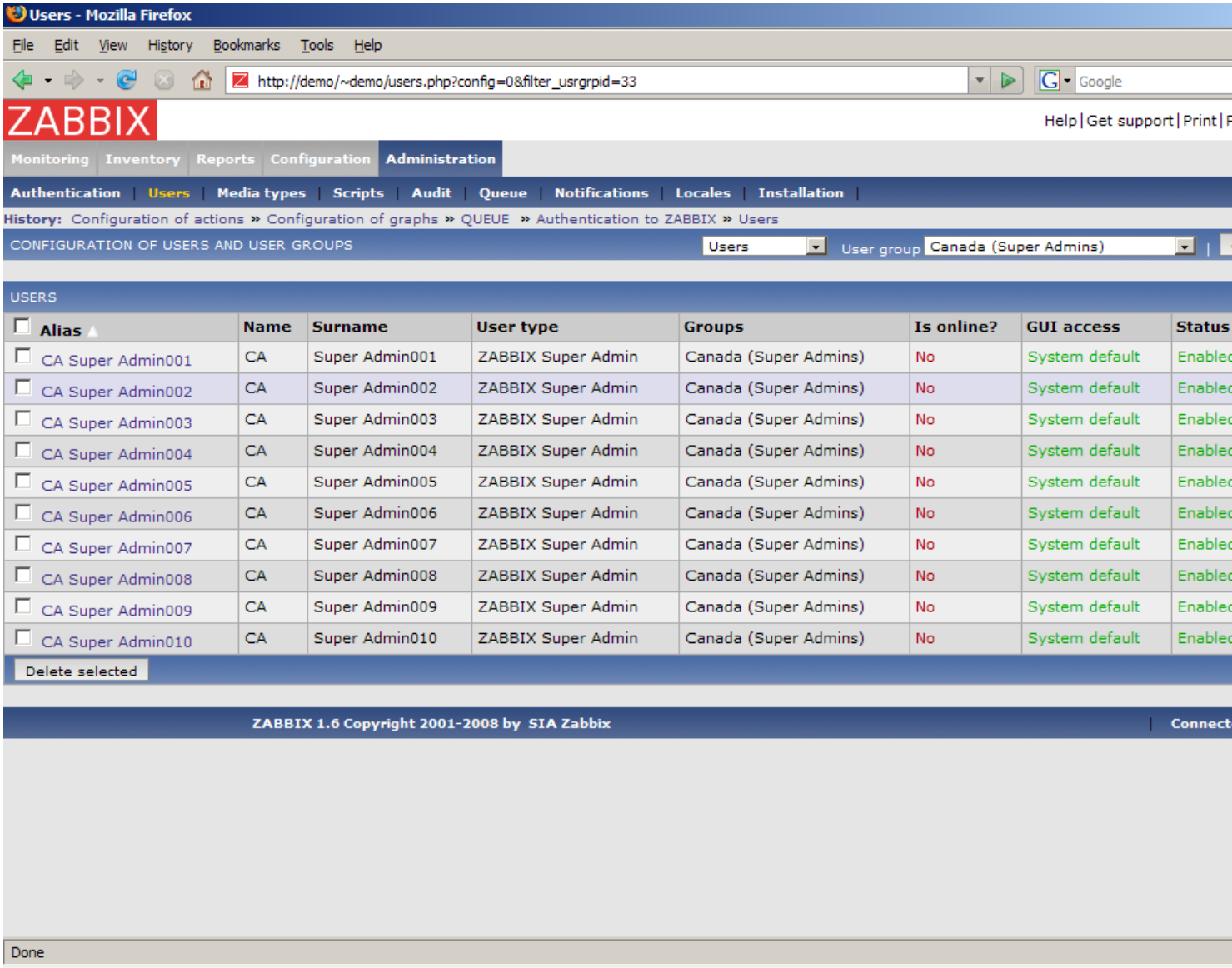
3.3 Users

3.3.1 Users

The screen can be used to manage Zabbix users.

**List of users**

It provides list of users.



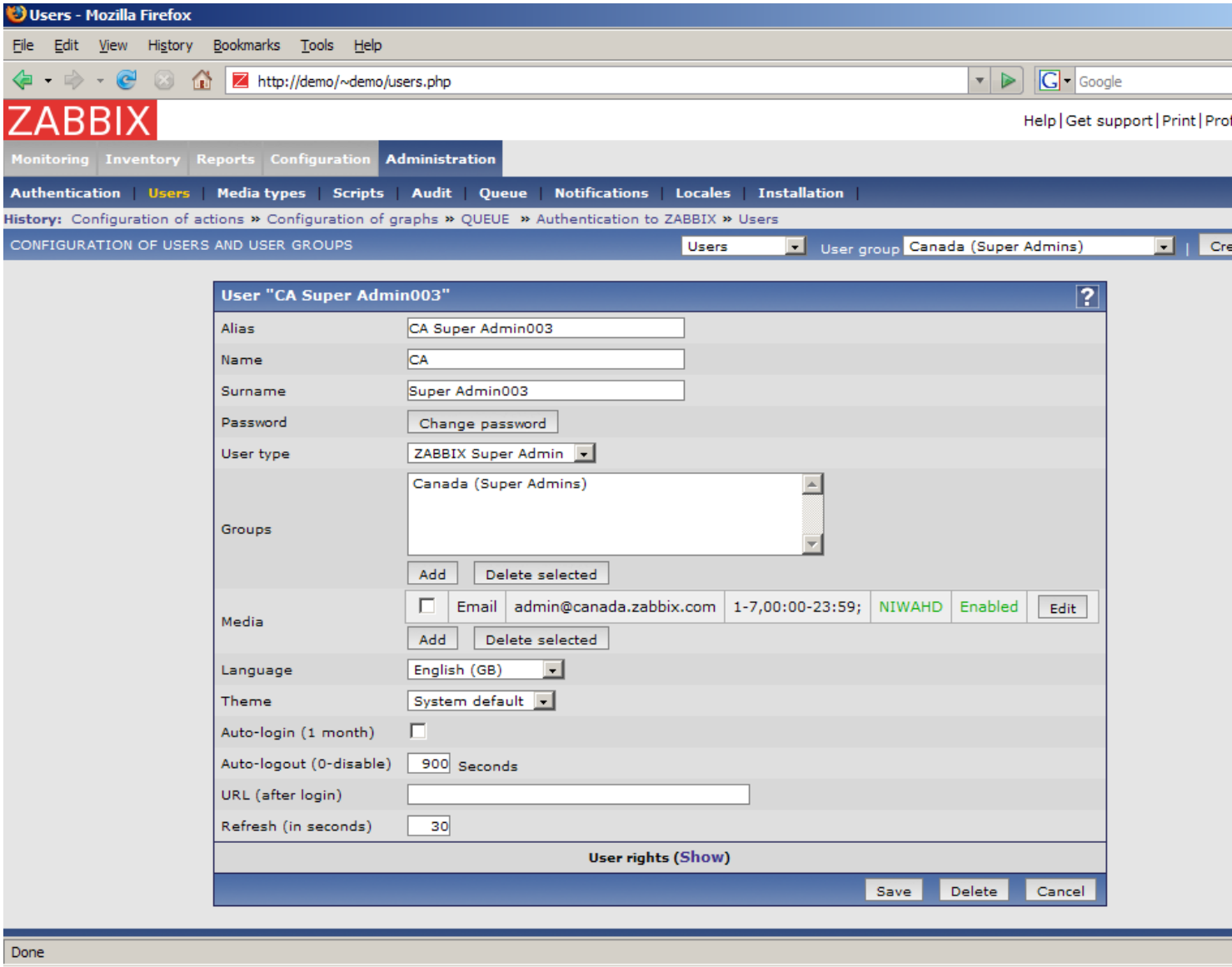

Displayed data:

| Parameter | Description |
|---|---|
| **Alias** | User short-name, i.e. login name. |
| **Name** | User name. |
| **Surname** | User surname. |

| Parameter | Description |
|---|---|
| **User type** | User type, one of following:<br>**Zabbix User**<br>**Zabbix Admin**<br>**Zabbix Super Admin** |
| **Groups** | List of all groups the user belongs to. |
| **Is online?** | Is user online. |
| **GUI Access** | Access to GUI, depends on settings of user groups:<br>**System default** – Zabbix, HTTP Authentication, LDAP Authentication<br>**Internal** – the user is authenticated by Zabbix regardless of system settings<br>**Disabled** – GUI access is restricted to this user |
| **Status** | User status, depends on settings of user groups:<br>**Enabled** – the user is active<br>**Disabled** – the user is disabled. The user is ignored by Zabbix. |
| **Actions** | |

#### User configuration

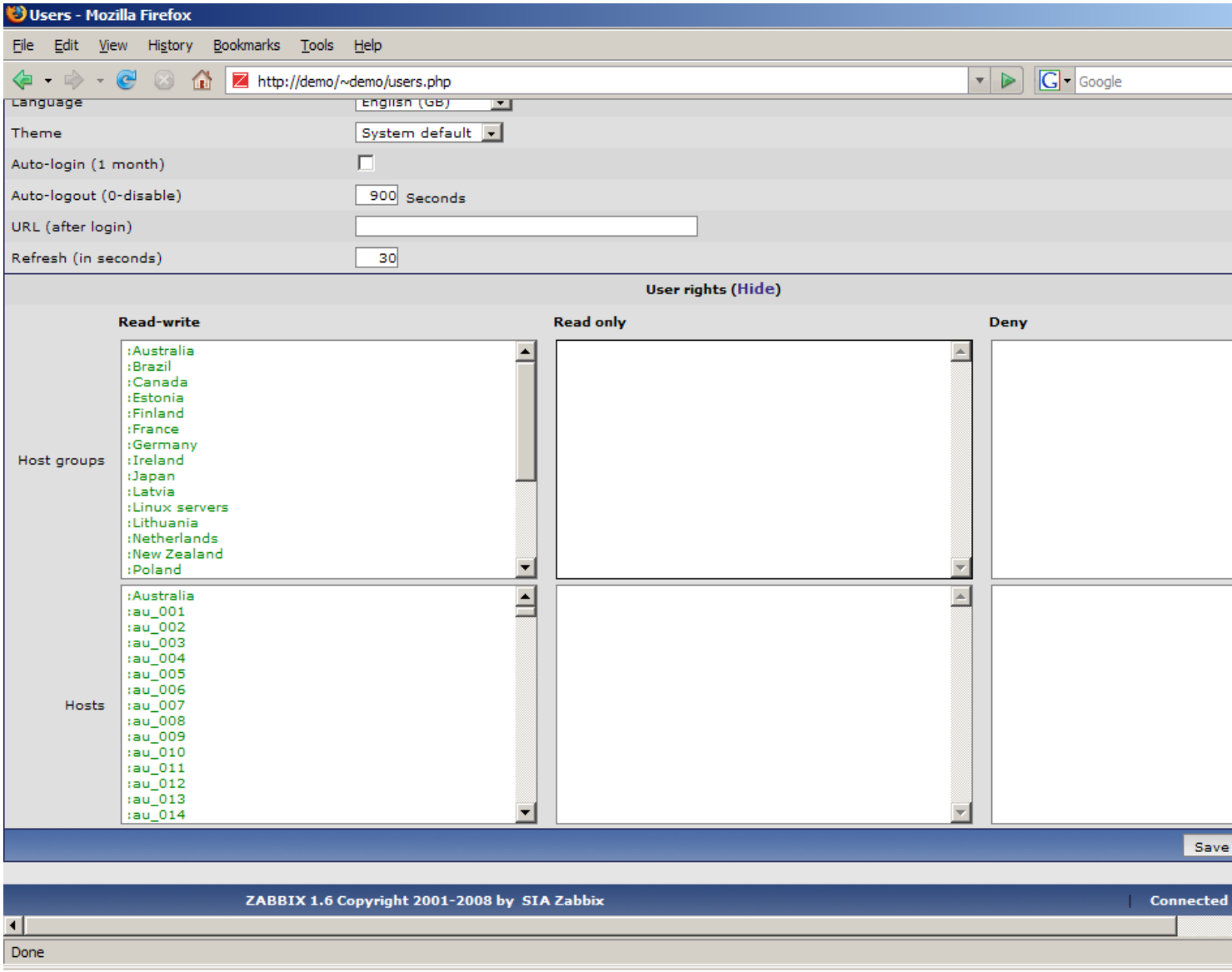
The screen provides user details and gives control to change user attributes.




Configuration parameters:

| Parameter | Description |
|---|---|
| **Alias** | User short-name, i.e. login name. Must be unique! |
| **Name** | User name. |
| **Surname** | User surname. |
| **User type** | User type, one of following:<br>**Zabbix User** – access to Monitoring tab only.<br>**Zabbix Admin** – access to Monitoring and Configuration tabs.<br>**Zabbix Super Admin** – access to everything, including Administration tabs. |
| **Groups** | List of all groups the user belongs to. |
| **Media** | List of all media for the user. The media are used by Zabbix for sending notifications.<br>You can specify the time period when the media is active. See Time period specification page for description of the format.<br>*Note:* Admin and Super Admin users can also edit their media details by accessing the Profile section in the upper right corner of the screen. |
| **Language** | Language of Zabbix GUI. |
| **Theme** | Defines how the GUI looks like:<br>**System Default** - use system settings<br>**Original Blue** – standard blue theme<br>**Black & Blue** – alternative theme |
| **Auto-login (1 month)** | Enable if you want Zabbix to remember you. Browser cookies are used for this. |
| **Auto-logout (0 - disable)** | User will be logged out after N seconds if inactivity. Set it to 0 to disable auto-logout. |
| **URL (after login)** | Make Zabbix to transfer you directly to a specific URL after successful login. |
| **Refresh (in seconds)** | Refresh rate used for graphs, screens, plain text data, etc. Can be set to 0 to disable. |

Click on User Rights **Show** to display user rights. It is impossible to change user rights here, the rights depend on user group membership! The information is available read-only.

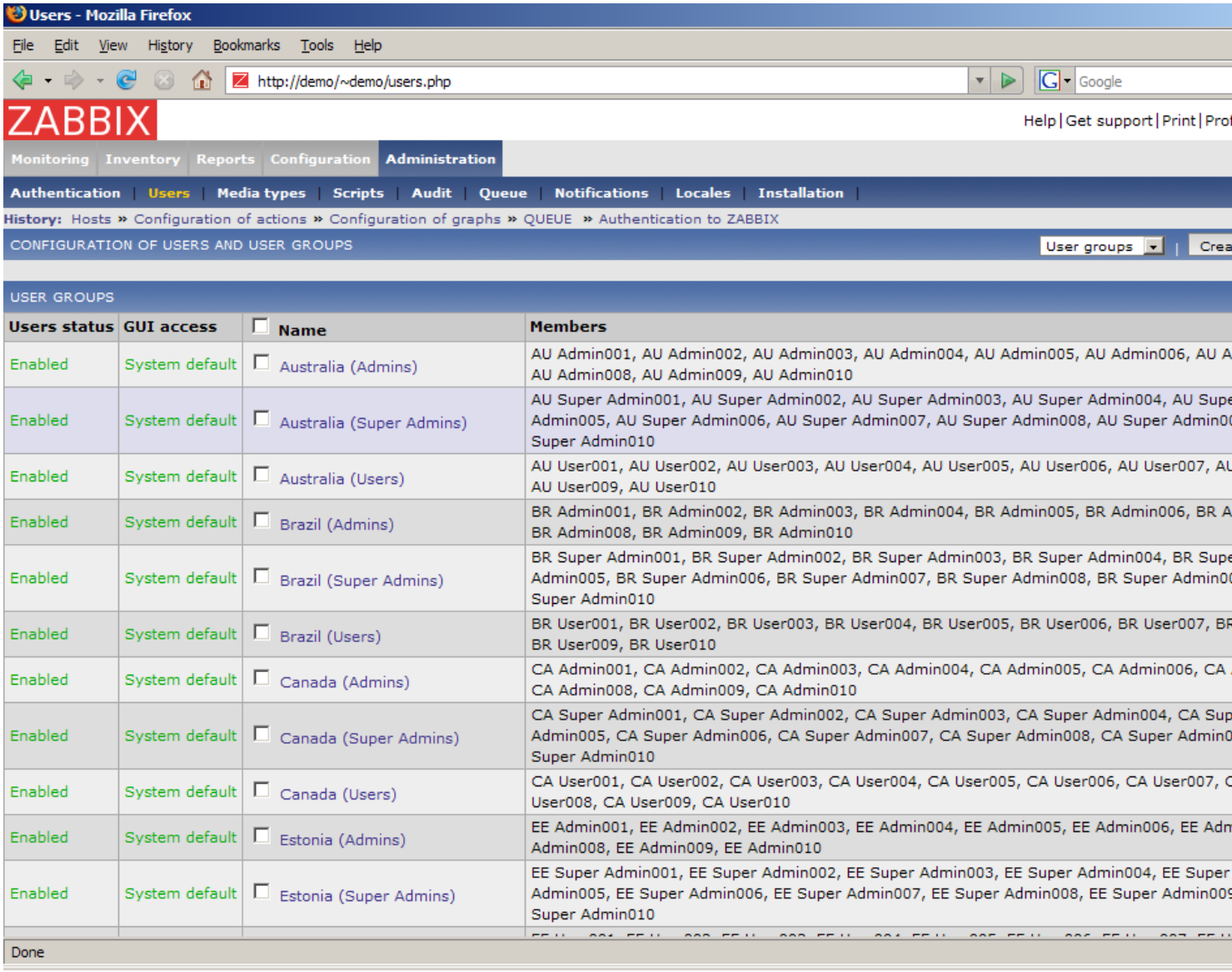



3.3.2 User Groups

The screen can be used to manage Zabbix user groups.

**List of user groups**

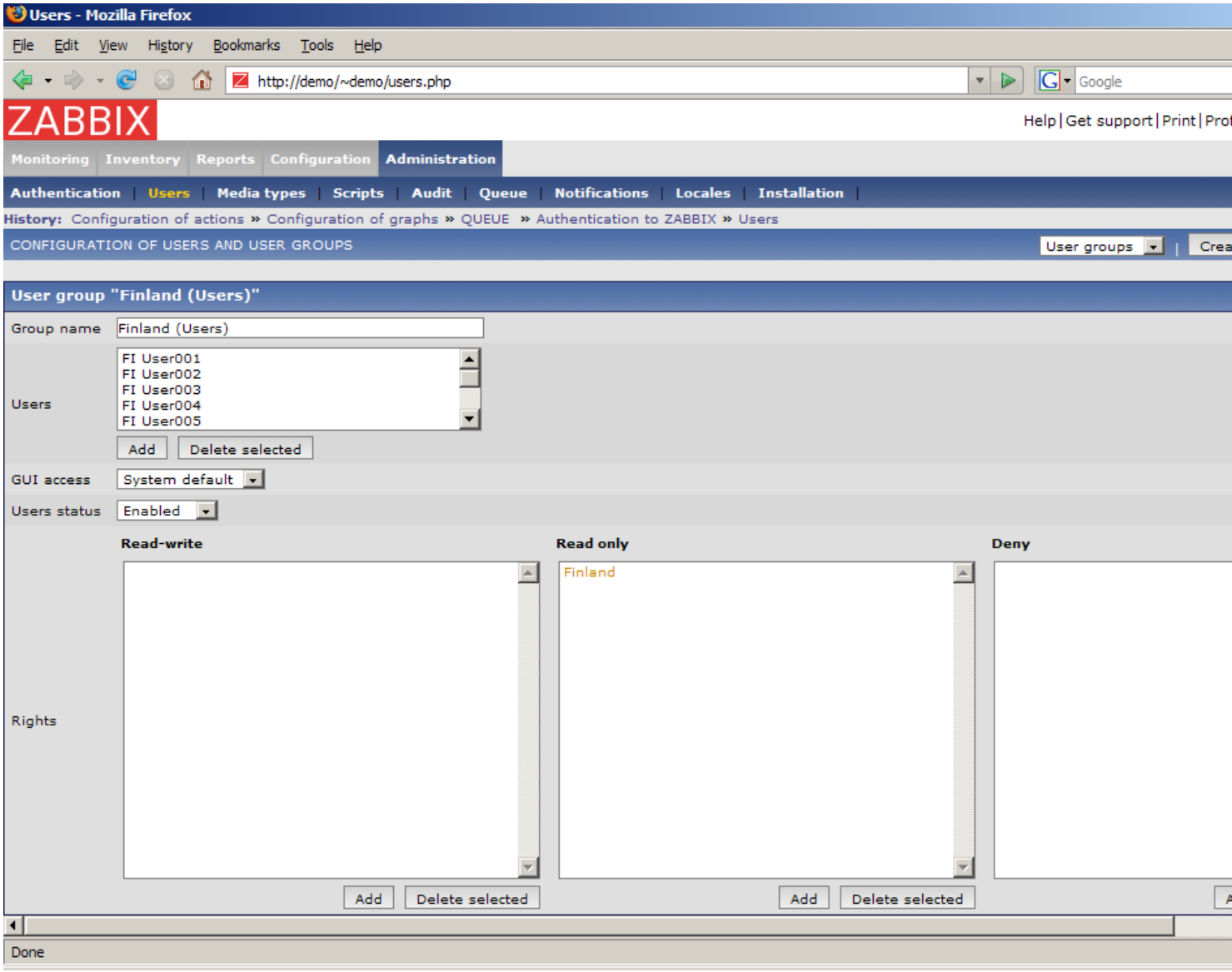
It provides list of user groups.




Displayed data:

| Parameter | Description |
|---|---|
| **Name** | Host group name. Must be unique. |
| **User status** | **Enabled** – users are active<br>**Disabled** – all users of the group are disabled |
| **GUI Access** | Displays how the users are authenticated.<br>**System default** – use default authentication<br>**Internal** – use Zabbix authentication<br>**Disabled** – access to Zabbix GUI is forbidden |
| **Members** | List of group members |

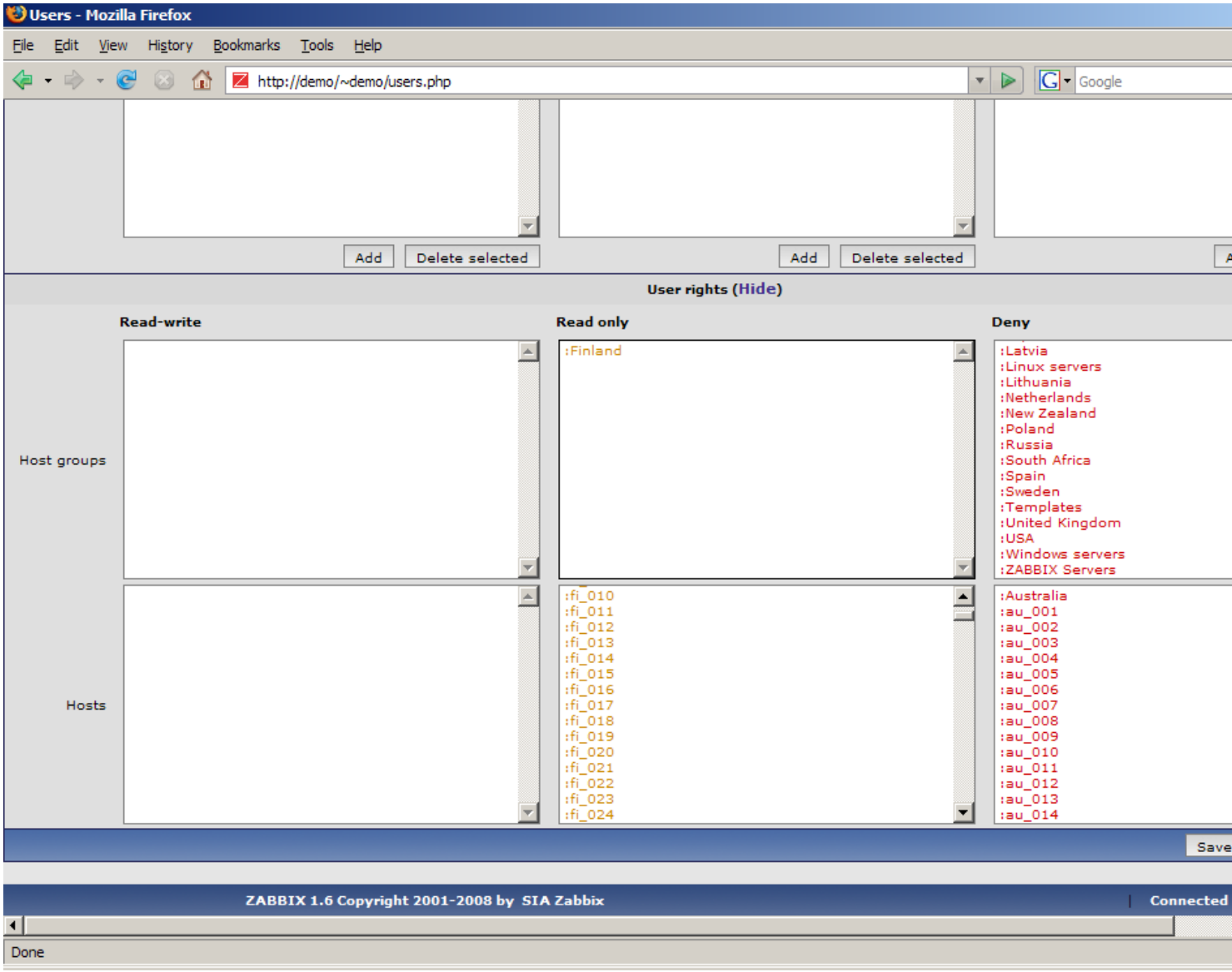
**User group configuration**




Configuration parameters:

| Parameter | Description |
|---|---|
| **Group name** | Unique group name. |
| **Users** | List of members of this group. |
| **GUI Access** | How the users of the group are authenticated.<br>**System default** – use default authentication<br>**Internal** – use Zabbix authentication<br>**Disabled** – access to Zabbix GUI is forbidden |
| **Users Status** | Status of group members:<br>**Enabled** – users are active<br>**Disabled** – users are disabled |
| **Rights** | Three lists for different host permissions:<br>**Read-write** – host groups with read-write access<br>**Read-only** – host groups with read-only access<br>**Deny** – host groups with deny access |

Click on User rights (**Show**) to see what permissions the user group has:




3.4 Media types

3.4.1 Media types

The screen can be used to manage Zabbix media types.

**List of media types**

Provides list of media types. Media type is a delivery method for user notifications.


Displayed data:

| Parameter | Description |
|---|---|
| Type | Media type:<br>**Email** – email notification<br>**SMS** – SMS notifications sent using serial GSM modem<br>**Jabber** – Jabber notification<br>**Script** – script based notification |
| Description | Name of the media. |
| Details | Configuration details, depends on media type. |

**Media configuration**

The screen provides user details and gives control to change media attributes.


Configuration parameters:

| Parameter | Description |
|---|---|
| **Description** | Unique media name. |
| **Type** | Media type:<br>**Email** – email notification<br>**SMS** – SMS notifications sent using serial GSM modem |
| | **SMTP Server** - server name<br>**SMTP Hello** – Hello string, normally domain name<br>**SMTP Email** – sender email address |
| | **GSM Modem** - serial device name of GSM modem |

| Parameter | Description |
|---|---|
| | **Jabber** – Jabber notification<br>**Jabber Identifier** - Jabber ID<br>**Password** – Password of the Jabber ID |
| | **Script** – script based notification<br>**Script name** - name of the custom script |

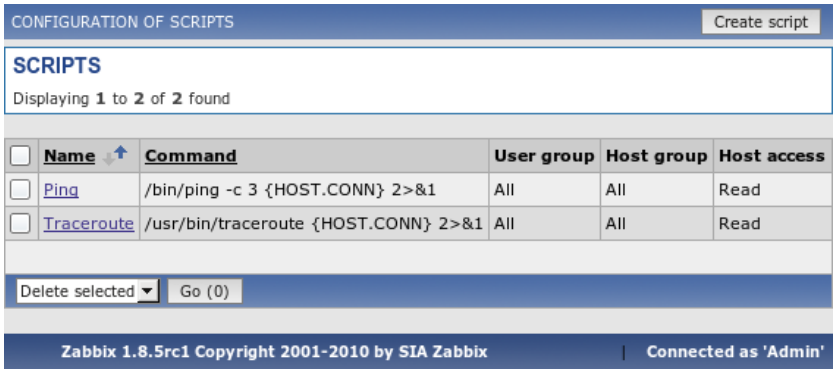
3.5 Scripts

The screen can be used to manage user-defined scripts. The scripts are executed on the Zabbix server even for hosts monitored by a proxy.

**List of scripts**

Provides a list of scripts known to Zabbix. Depending on permission, Zabbix user may execute a script from the front-end by clicking on host in these locations:

- Network maps
- Dashboard
- Status of triggers (*Monitoring* → *Triggers*)



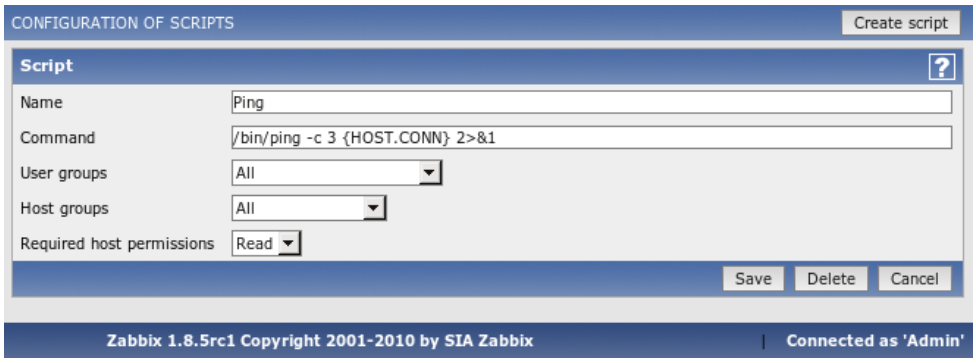

Displayed data:

| Parameter | Description |
|---|---|
| **Name** | Unique script name. |
| **Command** | Command to be executed. |
| **User group** | The script is available to members of the user group only. |
| **Host group** | The script is available for hosts of the host group only. |

| Parameter | Description |
|---|---|
| **Host access** | **Read** - user must have read permission for the host to execute the script<br>**Write** - user must have write permission for the host to execute the script. |

**Script configuration**

The screen provides script details and gives control to change script attributes.




Configuration parameters:

| Parameter | Description |
|---|---|
| **Name** | Unique script name. |
| **Command** | Full path to a command, which will be executed on user request.<br>The command will be run on the Zabbix server.<br>The following macros are supported here:<br>**{HOST.CONN}**<br>**{HOST.DNS}**<br>**{IPADDRESS}**<br>**{HOSTNAME}**<br>Example:<br>/bin/ping-c 3 {HOST.CONN}<br>A special syntax for IPMI commands must be used:<br>**IPMI <ipmi control> [value]**<br>Example:<br>IPMI power off |
| **User group** | The script is available to members of the user group only. |
| **Host group** | The script is available for hosts of the host group only. |
| **Host access** | **Read** - user must have read permission for the host to execute the script<br>**Write** - user must have write permission for the host to execute the script. |

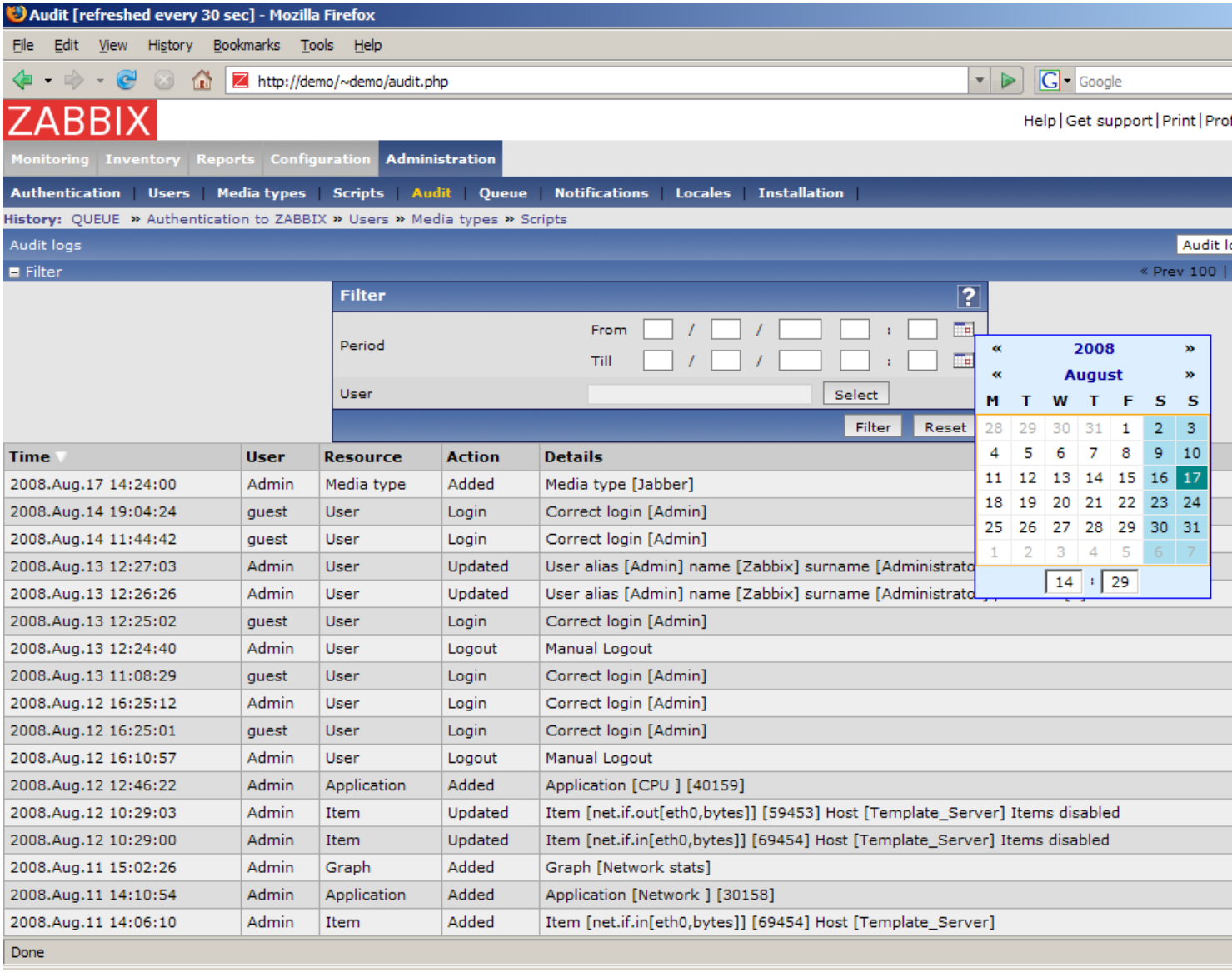
If macro may resolve to value with spaces (for example, host name), don’t forget to quote as needed.

Standard error is discarded, so make sure to redirect it to standard output manually.

3.6 Audit

The screen can be used to see front-end audit records and list of notifications sent to users.

**Audit logs**

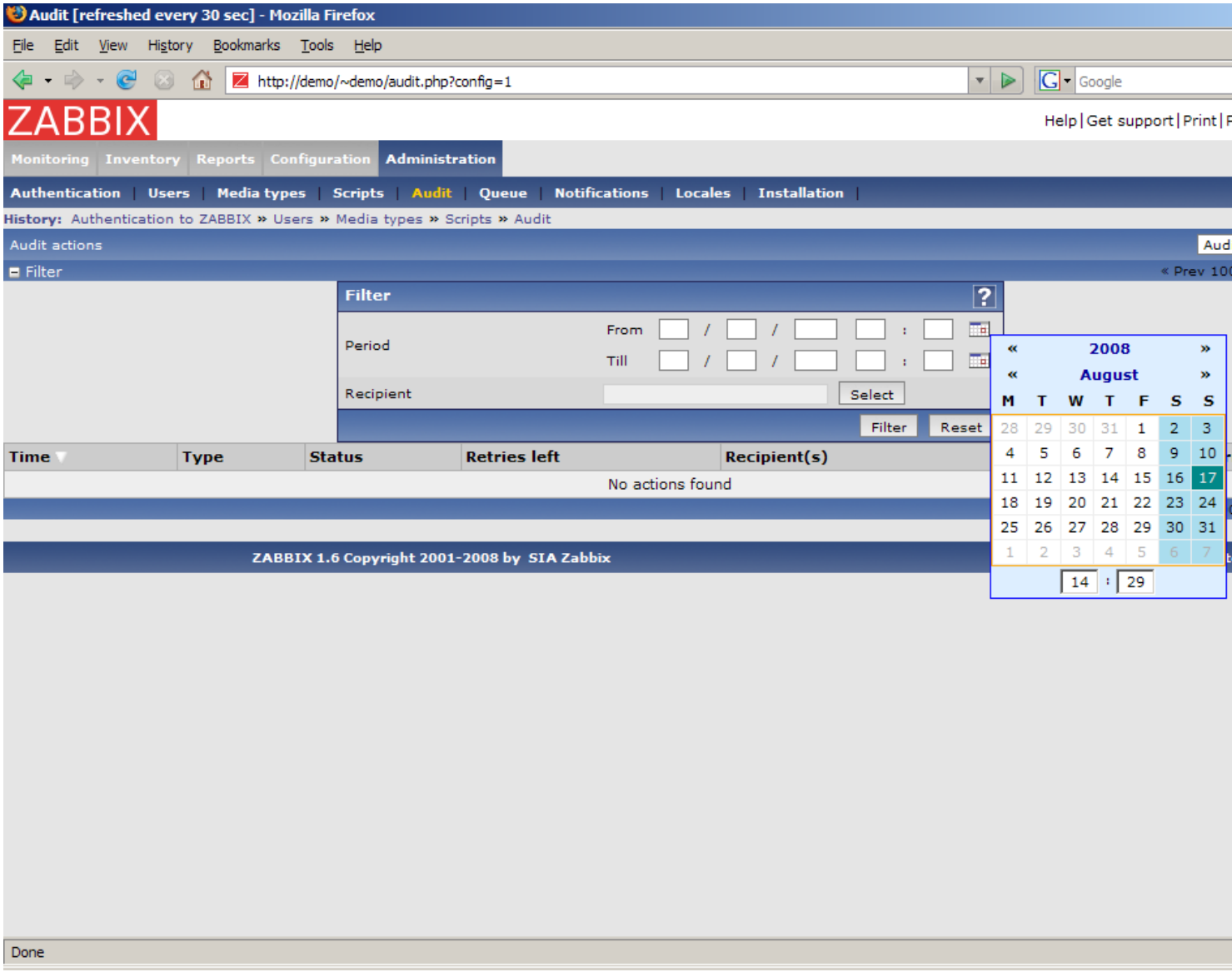



Displayed data:

| Parameter | Description |
|---|---|
| **Time** | Time stamp when an action took place. |
| **User** | User name. |
| **Resource** | Object, which was affected:<br>**Application**<br>**Graph**<br>**Host**<br>**Item**<br>**User** |
| **Action** | Performed action:<br>**Added**<br>**Login**<br>**Logout**<br>**Removed**<br>**Updated** |
| **Details** | More detailed information about action. |

**Audit actions**

The screen provides access to history of notifications and remote commands.



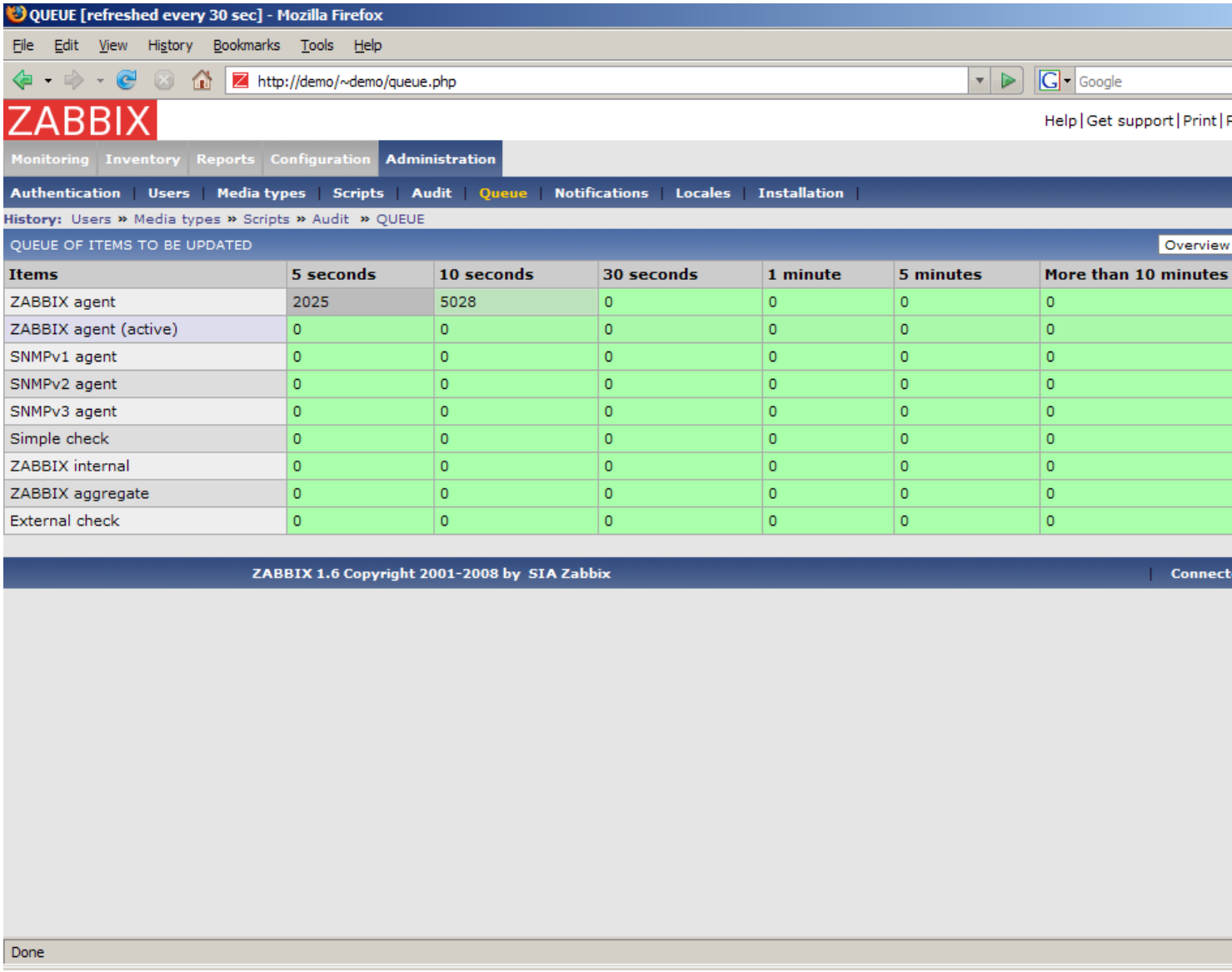

Displayed data:

| Parameter | Description |
|---|---|
| **Time** | Time stamp when an action took place. |
| **Type** | Type of executed operation:<br>**Notifications**<br>**Remote command** |
| **Status** | Status:<br>**Not sent**<br>**Sent** |
| **Retires left** | Number of retries left. |
| **Recipient(s)** | List of recipients. |
| **Message** | Message used in notification. |
| **Error** | Error if the notification was not sent. |

3.7 Queue

The Queue provides information about performance of Zabbix.

**Overview**



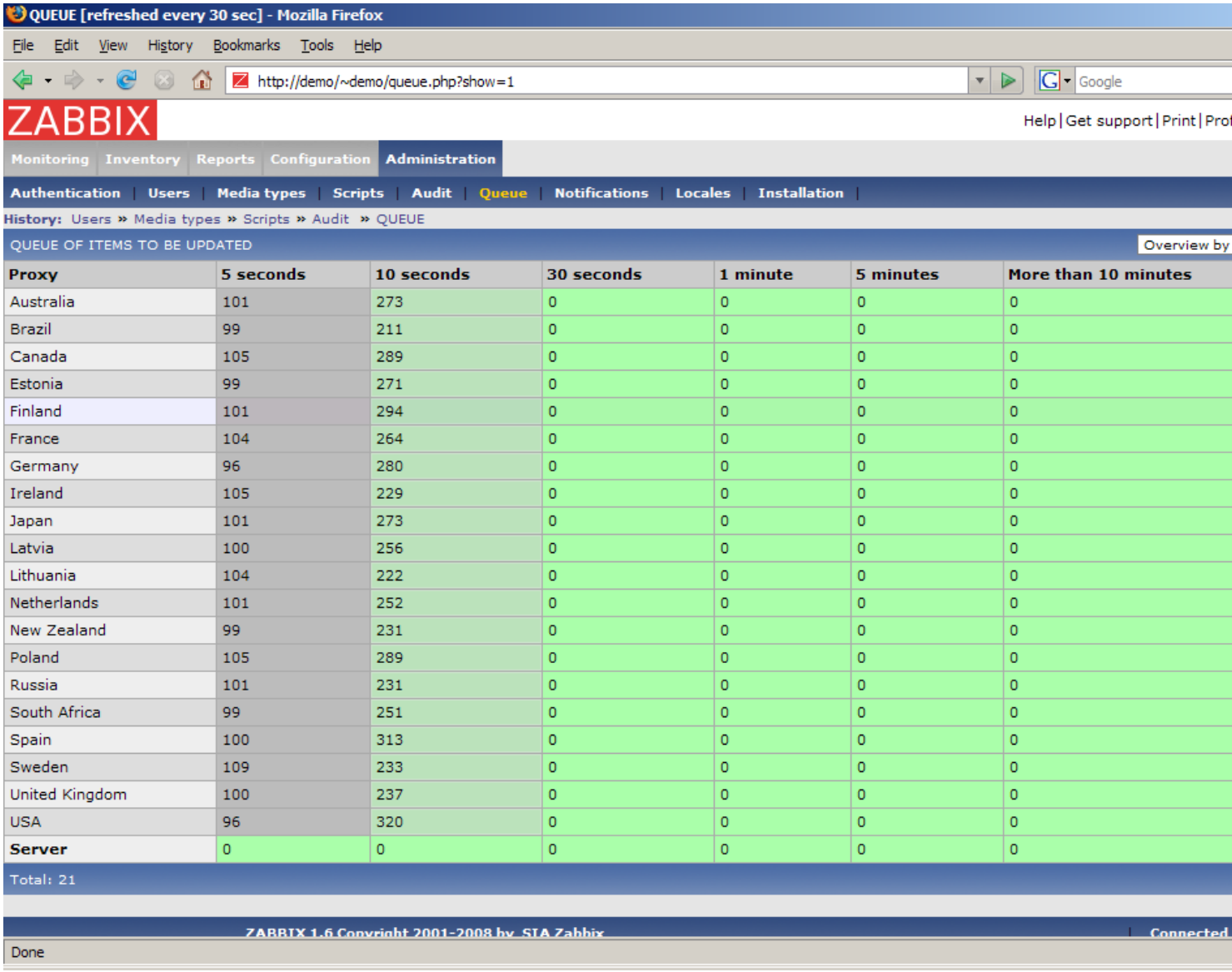

| Items | 5 seconds | 10 seconds | 30 seconds | 1 minute | 5 minutes | More than 10 minutes |
|---|---|---|---|---|---|---|
| ZABBIX agent | 2025 | 5028 | 0 | 0 | 0 | 0 |
| ZABBIX agent (active) | 0 | 0 | 0 | 0 | 0 | 0 |
| SNMPv1 agent | 0 | 0 | 0 | 0 | 0 | 0 |
| SNMPv2 agent | 0 | 0 | 0 | 0 | 0 | 0 |
| SNMPv3 agent | 0 | 0 | 0 | 0 | 0 | 0 |
| Simple check | 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 |

For each item type the following data is displayed:

| Parameter | Description |
|---|---|
| **Items** | Item type |
| **5 seconds** | Data is delayed for 5-10 seconds. |
| **10 seconds** | Data is delayed for 10-30 seconds. |
| **30 seconds** | Data is delayed for 30-60 seconds. |
| **1 minute** | Data is delayed for 1-5 minutes. |
| **5 minutes** | Data is delayed for 5-10 minutes. |
| **More than 10 minutes** | Data is delayed for more than 10 minutes. |

#### Overview by proxy

The view gives more detailed information about performance of Zabbix Server and Proxies.

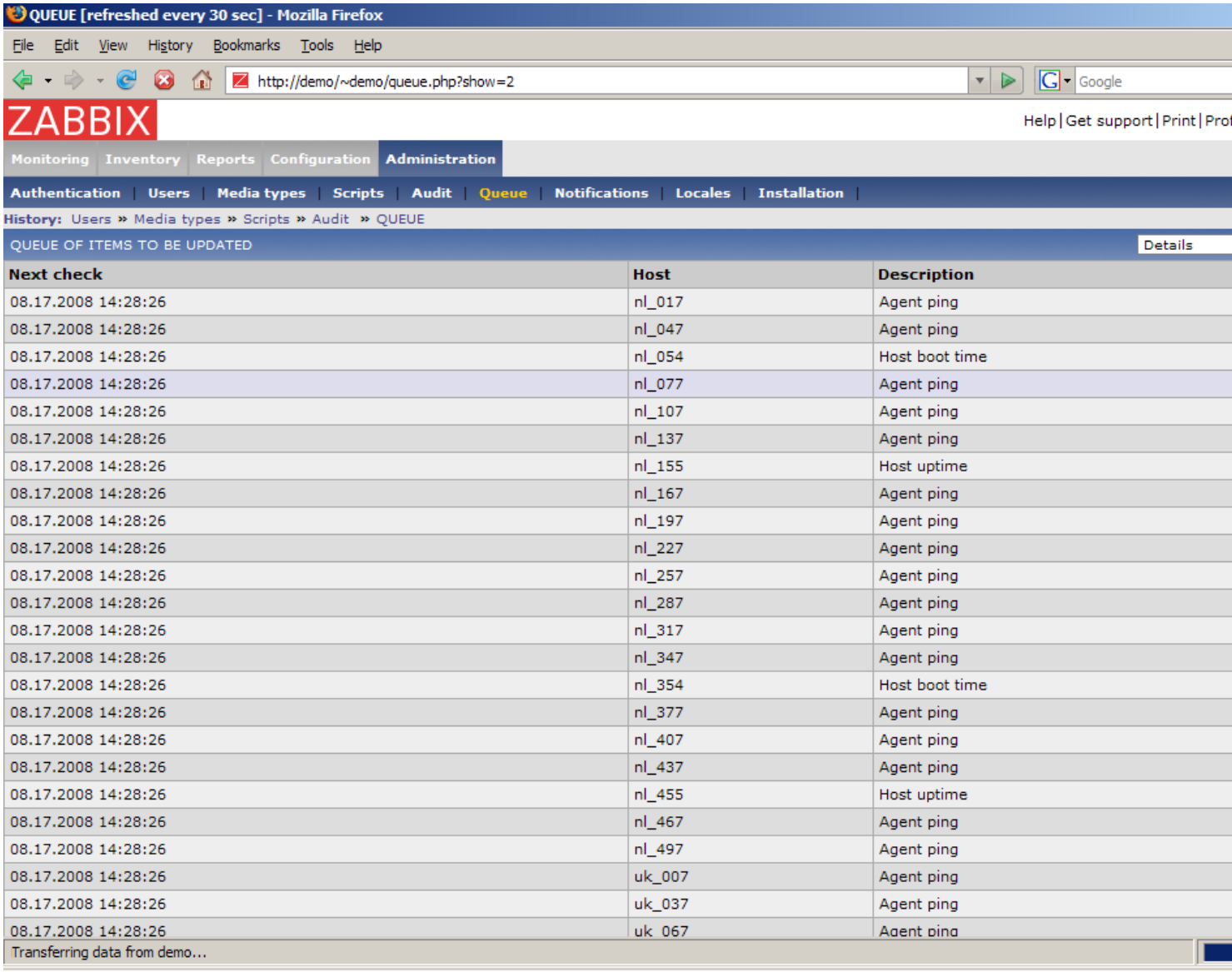



For each Proxy and local Zabbix Server the following data is displayed:

| Parameter | Description |
|---|---|
| **Proxy** | Proxy name or **Server**. **Server**, displayed last, shows statistics about local server. |

**Details**

The view gives very detailed information about delayed items.

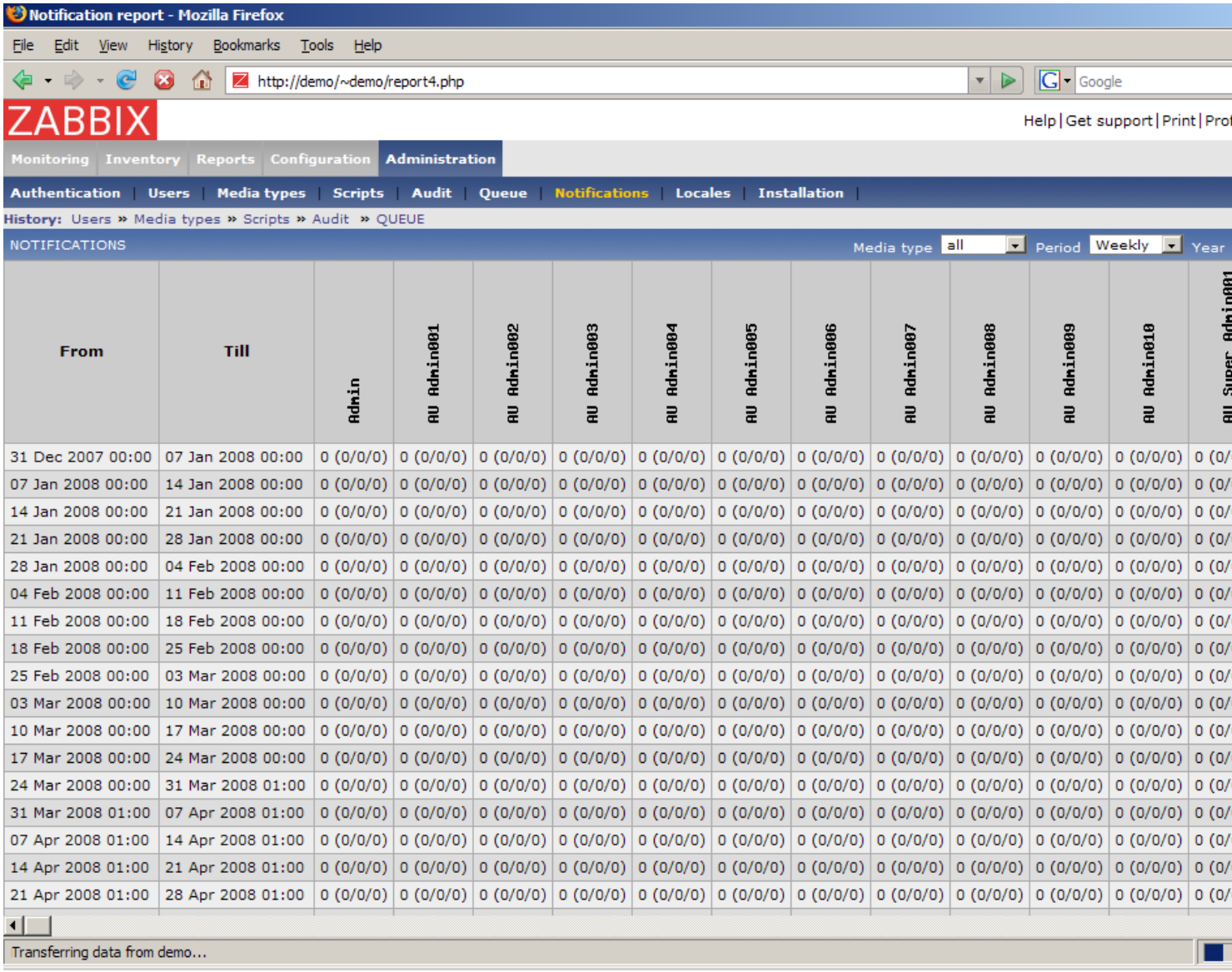



List of items is displayed with the following details:

| Parameter | Description |
|---|---|
| **Next check** | Expected time stamp of next data retrieval. The time stamps will always be in the past. |
| **Host** | Host name. |
| **Description** | Item name. |

### 3.8 Notifications

This is report on number of notifications sent to each user grouped by media types.



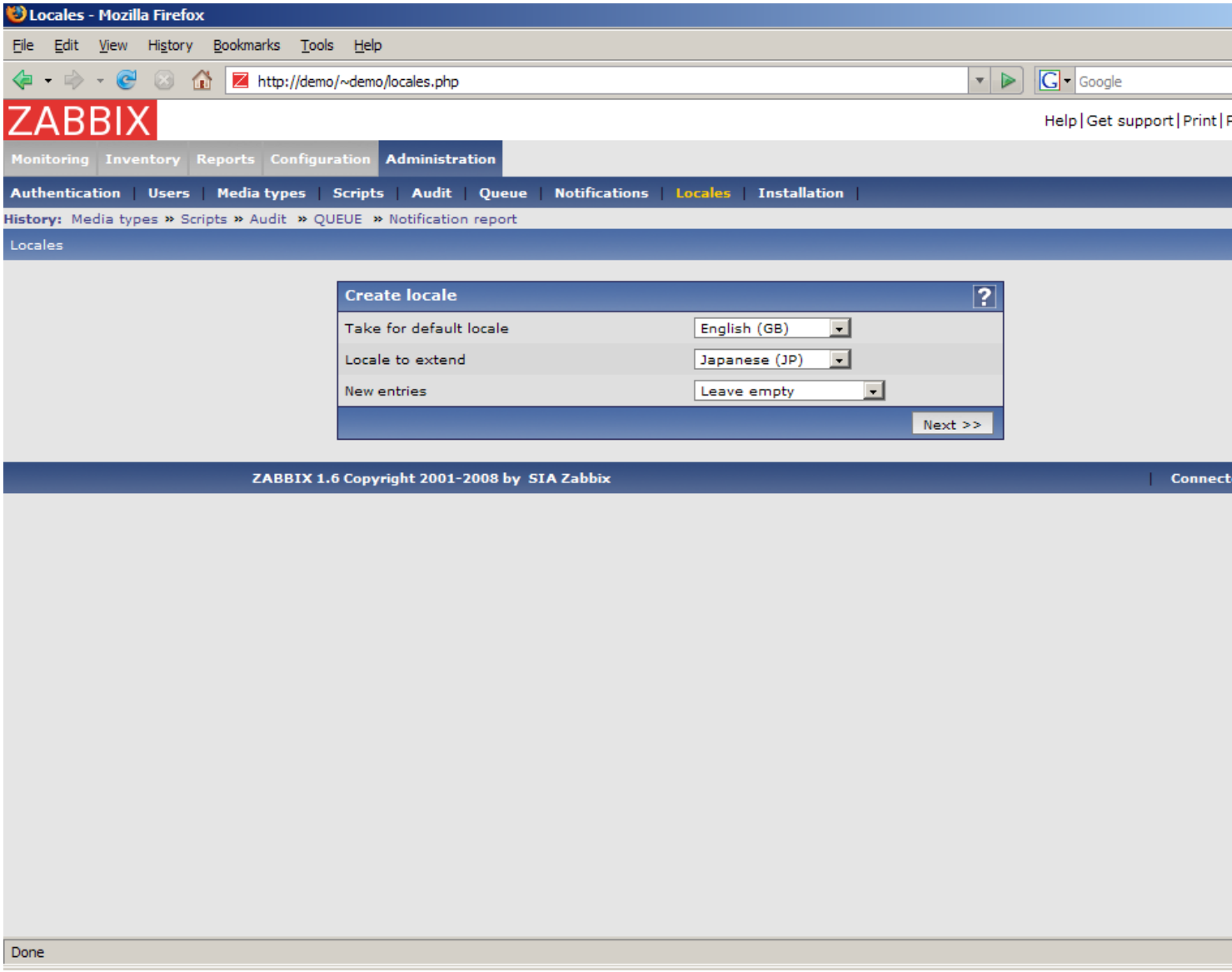

For each user number of notifications is displayed per each media type.

3.9 Locales

Locales provides functionality for easy editing of translations of Zabbix front-end.

**Locale selection**

Select locale you'd like to select for further processing.

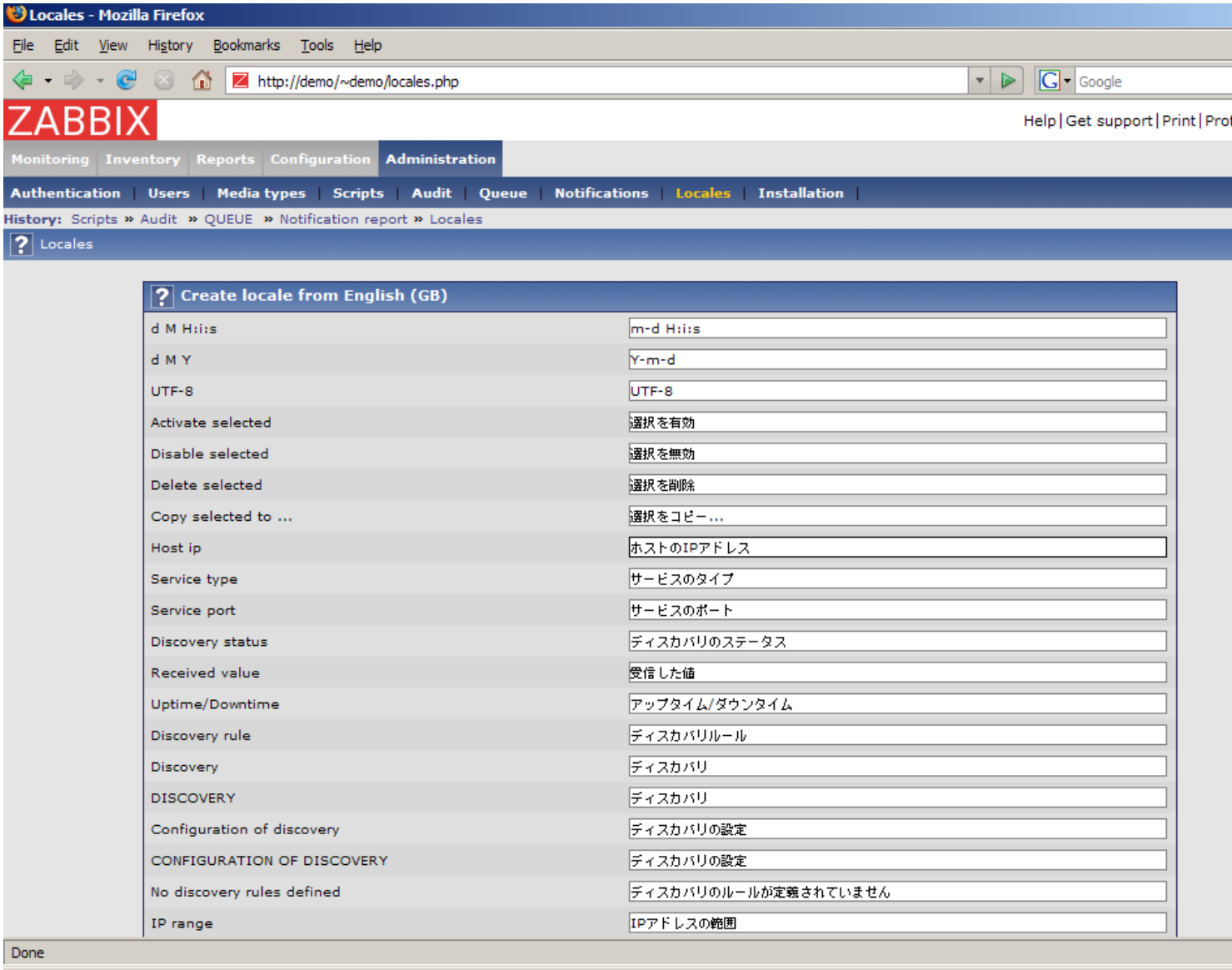



Parameters:

| Parameter | Description |
|---|---|
| **Take for default locale** | The locale will be used as a base one. |
| **Locale to extend** | Select language you'd like to improve. |
| **New entries** | **Do not add** – if something is not translated, ignore it<br>**Leave empty** – if something is not translated, leave translation empty<br>**Fill with default value** – if something is not translated, fill translation with default value |

**Translation form**

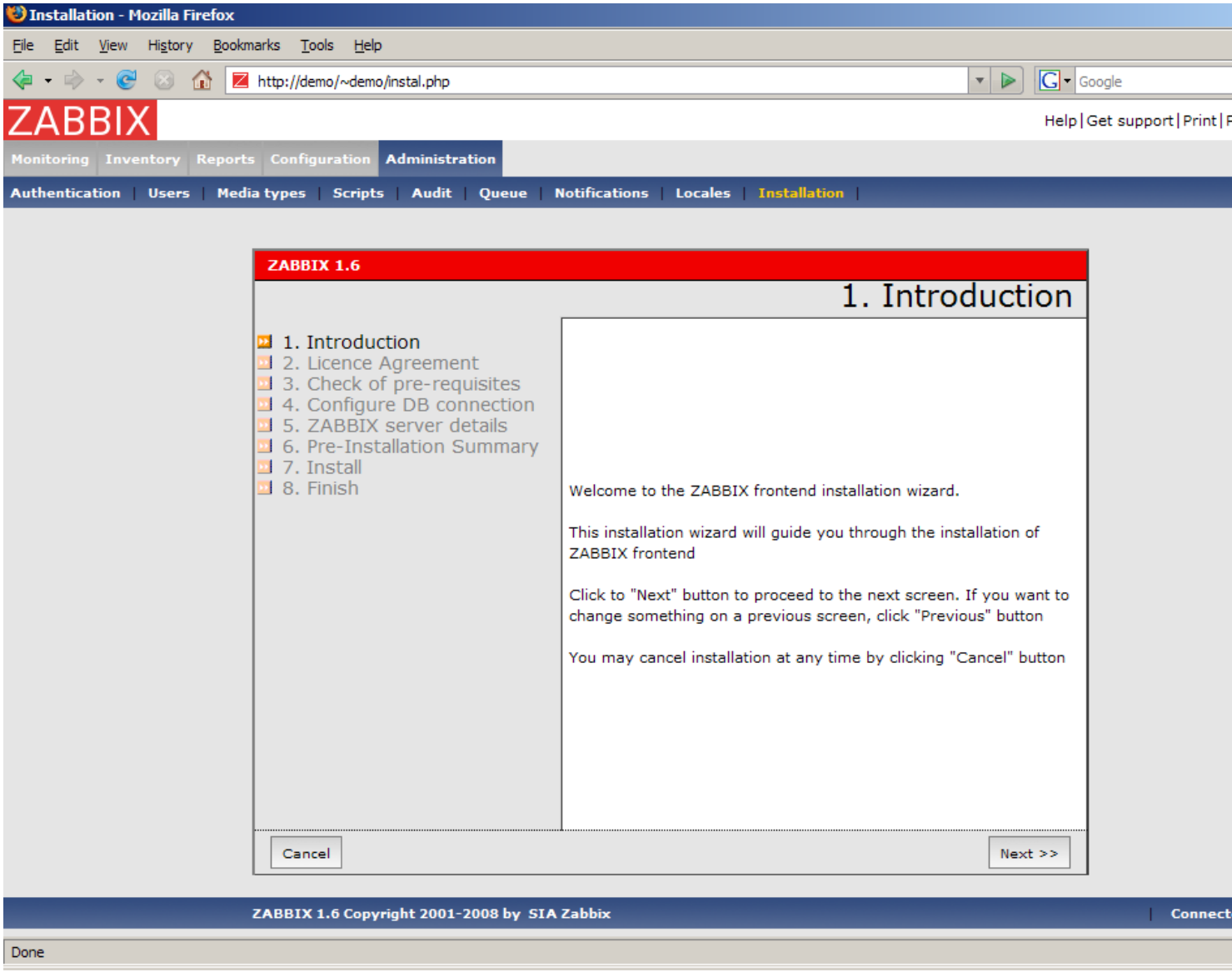
This form is used to translate phrases used in Zabbix front-end. Left side is filled with default language, right side consists of translated phrases.




Once translation is ready, press button “Download” to have translation file, which can be used to replace files under **include/locales**.

3.10 Installation

The screen makes possible creation of Zabbix front-end configuration file.




**4 Page parameters**

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 (&).

4.1 Status of triggers

Accessed as *Monitoring* → *Triggers*, page name `tr_status.php`.

4.1.1 Generic parameters

- groupid
- hostid
- fullscreen

4.1.2 Page specific parameters

- show_triggers - filter option **Triggers status**, 1 - Problem, 2 - Any
- 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**, -1 - All, 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

## 18 Performance Tuning

real_world_configuration performance_tuning_detail

### 1 Real world configuration

Server with Zabbix 1.0 installed (RedHat Linux 8.0, kernel 2.4.18-14, MySQL/MyISAM 3.23.54a-4, Pentium IV 1.5Ghz, 256Mb, IDE) is able to collect more than 200 parameters per second from servers being monitored (assuming no network delays).

How many servers can be monitored by Zabbix on the hardware, one may ask? It depends on number of monitored parameters and how often Zabbix should acquire these parameters. Suppose, each server you monitor has ten parameters to watch for. You want to update these parameters once in 30 seconds. Doing simple calculation, we see that Zabbix is able to handle 600 servers (or 6000 checks). In case if these parameters need to be updated once in a minute, the hardware configuration will be able to handle 600x2=1200 servers. These calculations made in assumption that all monitored values are retrieved as soon as required (latency is 0). If this is not a requirement, then number of monitored servers can be increased even up to 5x-10x times.

### 2 Performance tuning

It is very important to have Zabbix system properly tuned for maximum performance.

2.1 Hardware

General advices 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
- User fast RAID storage
- Use fast Ethernet adapter
- Having more memory is always better

2.2 Operating System

- Use latest (stable!) version of OS
- Exclude unnecessary functionality from kernel
- Tune kernel parameters

2.3 Zabbix configuration parameters

Many parameters may be tuned to get optimal performance.

2.3.1 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].

**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.

2.4 Database Engine

This is probably most important part of Zabbix tuning. Zabbix heavily depends on availability and performance of database engine.

- use fastest database engine, i.e. MySQL
- 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 of MySQL temporary files in tmpfs is recommended

2.5 General advices

- monitor required parameters only
- tune 'Update interval' for all items. Keeping 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 same information.

Example: why use system[procload],system[procload5] andsystem[procload15] if system[procload] contains all.

- avoid use of triggers with long period given as function argument. For example, max(3600) will be calculated significantly slower than max(60).

## 19 Cookbook

general specific integration

### 1 General Recipes

1.1 Monitoring of server's availability

At least three methods (or combination of all methods) may be used in order to monitor availability of a server.

- ICMP ping (Key "icmpping")
- Key "status"
- Trigger function nodata() for monitoring availability of hosts using only active checks

1.2 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.

### 2 Monitoring of Specific Applications

2.1 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.

2.2 MySQL

Configuration file misc/conf/zabbix_agentd.conf contains list of parameters that can be used for monitoring of MySQL.

```
### Set of parameter 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 -f1 -d"T"
#UserParameter=mysql[threads],mysqladmin -uroot status|cut -f3 -d":"|cut -f1 -d"Q"
#UserParameter=mysql[questions],mysqladmin -uroot status|cut -f4 -d":"|cut -f1 -d"S"
#UserParameter=mysql[slowqueries],mysqladmin -uroot status|cut -f5 -d":"|cut -f1 -d"O"
#UserParameter=mysql[qps],mysqladmin -uroot status|cut -f9 -d":"
#UserParameter=version[mysql],mysql -V
```

2.2.1 mysql[ping]

Check whether MySQL is alive

```
    Result: 0 - not started 1 - alive
```

2.2.2 mysql[uptime]

Number of seconds MySQL is running

2.2.3 mysql[threads]

Number of MySQL threads

2.2.4 mysql[questions]

Number of processed queries

2.2.5 mysql[slowqueries]

Number of slow queries

2.2.6 mysql[qps]

Queries per second

2.2.7 mysql[version]

Version of MySQL Example: mysql Ver 11.16 Distrib 3.23.49, for pc-linux-gnu (i686)

2.3 Mikrotik routers

Use SNMP agent provided by Mikrotik. See http://www.mikrotik.com for more information.

2.4 WIN32

Use Zabbix W32 agent included (pre-compiled) into Zabbix distribution.

2.5 Novell

Use MRTG Extension Program for NetWare Server (MRTGEXT.NLM) agent for Novell. The agent is compatible with protocol used by Zabbix. It is available from http://forge.novell.com/modules/xfmod/project/?mrtgext.

Items have to be configured of type Zabbix Agent and must have keys according to the MRTGEXT documentation.

For example:

** UTIL1 **

1 minute average CPU utilization

** CONNMAX **

Max licensed connections used

** VFKSys **

bytes free on volume Sys:

Full list of parameters supported by the agent can be found in readme.txt, which is part of the software.

2.6 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.

2.7 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.

2.8 JMX

First of all, you need to configure your jvm to allow jmx monitoring. How do you know if you can do this? You can use the sun jconsole utility that comes with the jdk and point it at your machine running the jvm. If you can connect, you are good.

In my tomcat environment, I enable it by setting the following options for the jvm:

```
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=xxxxx \
-Dcom.sun.management.jmxremote.ssl=false \
-Dcom.sun.management.jmxremote.authenticate=true \
-Dcom.sun.management.jmxremote.password.file=/path/java/jre/lib/management/jmxremote. password"
```

This tells the jmx server to run on port XXXXX, to use password authentication, and to refer to the passwords stored in the jmxremote.password file. See the sun docs on jconsole for details. (You might consider enabling ssl to make the connection more secure.)

Once that is done, I can then run jconsole and see everything that is currently exposed (and to verify that I can connect properly). jconsole will also provide you the information you need to query specific jmx attributes from the information tab.

Now, since I use Tomcat, there are two ways that I can grab the jmx attribute values (or effect a jmx operation). The first way is I can use the servlet provided by Tomcat. (Don't know what jboss has). The second way is I can send well formatted requests via a jmx command line tool.

Let's say I am interested in peak threads used by the system. I browse down through the jmx objects via jconsole, find it under java.lang, Threading. After selecting Threading, I click on the info tab, and I can see the name of the mbean is "java.lang:type=Threading"

With tomcat, I can do the following:

```
curl -s -u<jmxusername>:<jmxpassword> 'http://<tomcat_hostname>/manager/jmxproxy/?qry=java.lang:type=Threa
```

where the jmx username and password are the ones defined in the file defined in the jvm options above, the qry string is the one obtained from jconsole.

The output from this will be all the metrics from this jmx key. Parse the output and grab the number of your choice.

If you don't have a servlet that will allow you to make a http request to the jmx interface, you can use the command line tool like this

```
/<pathTo>/java -jar /<pathTo>/cmdline-jmxclient.jar <jmxusername>:<jmxpassword> <jvmhostname>:<jmxport> ja
```

The difference with the command line client is you need to specify the attribute you are interested in specifically. Leaving it out will give you a list of all the attributes available under Threading.

Again, parse the output for the data of your choice.

Once you can reliably grab the data you are interested in, you can then turn that command into a zabbix userparm. e.g.

```
UserParameter=jvm.maxthreads, /usr/bin/curl -s -u<jmxusername>:<jmxpassword> 'http://<tomcat_hostname>/man
```

or

```
UserParameter=jvm.maxthreads, /<pathTo>/java -jar /<pathTo>/cmdline-jmxclient.jar <jmxusername>:<jmxhostna
```

That's it.

I prefer getting my stats from the servlet via http rather than using the java command line client as it is much "lighter" to start up and grab the information.

Need a command line jmx client? I use the one from here: http://crawler.archive.org/cmdline-jmxclient/

Information on setting up jmx monitoring for your jvms http://java.sun.com/j2se/1.5.0/docs...ment/agent.html

General Information on JMX http://java.sun.com/j2se/1.5.0/docs...verviewTOC.html

> **Note:**
> Apparently the 1.5 jvm also supports SNMP which provides another option.

## 3 Integration

### 3.1 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.

## 20 Troubleshooting

sound_in_browsers error_and_warning_messages

#### 1 Error and warning messages

Zabbix daemons generate error and warning messages in case of any problems. The messages are written to log files depending on configuration parameters.

Some of the messages are numbered.

The table contains complete list of numbered messages with additional details.

| Error | Message | Details |
|---|---|---|
| **Z3001** | Connection to database '%s' failed: [%d] %s | Zabbix daemon is unable to establish connection to the database. Additional information:<br>database name<br>database error code<br>database error string |
| **Z3002** | Cannot create database '%s': [%d] %s | Zabbix daemon is unable to create database. Additional information:<br>database name<br>database error code<br>database error string |
| **Z3003** | No connection to the database. | This should never happen. Report to Zabbix Team. |
| **Z3004** | Cannot close database: [%d] %s | Zabbix daemon is unable to close connection to the database. Additional information:<br>database error code<br>database error string |
| **Z3005** | Query failed: [%d] %s [%s] | SQL query execution failed. Additional information:<br>database error code<br>database error string<br>SQL query string |

| Error | Message | Details |
|---|---|---|
| **Z3006** | Fetch failed: [%d] %s | Record fetch failed. Additional information:<br>database error code<br>database error string |

**Note:**
The numbered error messages are supported starting from Zabbix 1.8.

**2 Sound in browsers**

Sounds in web browsers for Zabbix frontend have been tested in the following 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

For playing sounds in Zabbix in the user's profile "GUI Messaging" should be enabled for all trigger severities and in the GUI global notification pop-up window sounds also should be enabled.

2.1 Safari 5.0

Quick Time Player is required.

2.2 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 Windows Media Player in *Tools→Options→File Types* enable *Windows audio file (wav)*

In Windows Media Player in Tools→Options tab "File Types" is available only if user is a member of groups "Power Users" or "Administrators", i.e. regular User do not have access to this tab and do not see it.

Additional thing - if IE do not have some *.wav file in the local cache directory (%userprofile%\Local Settings\Temporary Internet Files) then sound will not play the first time.

2.3 Firefox v 3.5.16

For playing wav files in the Firefox browser can use one of the following applications: Windows Media Player or Quick Time plug-in. These configuration settings should be performed in the Tools→Options→Applications menu, there are settings for the "Wave sound (audio/wav)" -you should set Windows Media Player for playing these files.

2.4 Known not to work

Browsers where the sound did not work:

- Opera 10.11 on Linux.

## 21 Escalations and repeated notifications

overview simple_messages remote_commands repeated_notifications delayed_notifications escalate_to_boss complex_scenario

## 1 Overview

Zabbix provides effective and extremely flexible functionality for escalations and repeated notifications. Depending on configuration, Zabbix will automatically escalate (increase escalation step) unresolved problems and execute actions assigned to each escalation step.

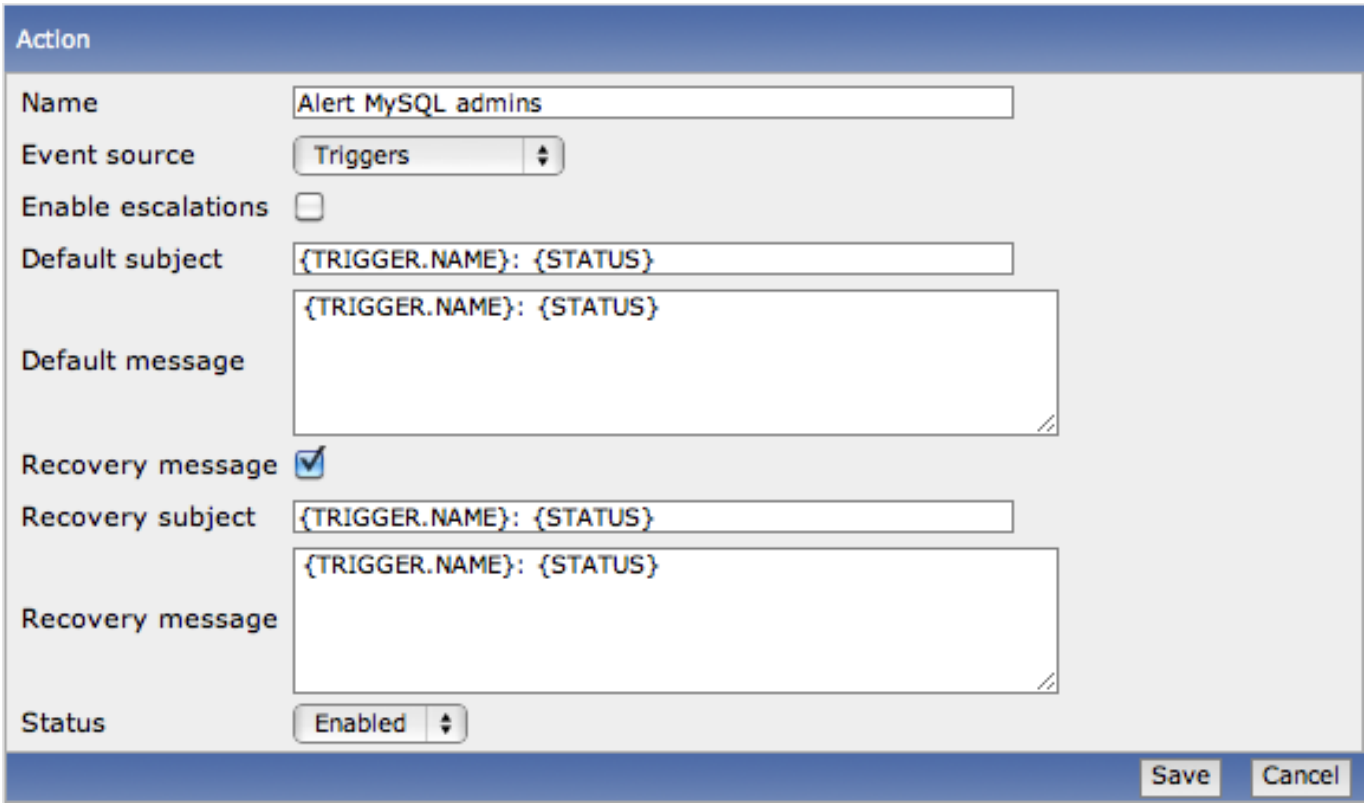
Zabbix supports the following scenarios for escalations, notifications and remote commands:

- Immediately inform users about new problems
- Pro-active monitoring, Zabbix executes arbitrary scripts (remote commands)
- Repeated notifications until problem is resolved
- Delayed notifications and remote commands
- Escalate problems to other user groups
- Different escalation path for acknowledged and unacknowledged problems
- Execute actions (both notifications and remote commands) if a problem exists for more than N hours (seconds, minutes, etc).
- Recovery message to all interested parties
- Zabbix supports unlimited number of escalation steps

## 2 Simple messages

**Warning:**
Warning: before enabling recovery messages or escalations, make sure to add "Trigger value = PROBLEM" condition to the action, otherwise remedy events can become escalated as well.

In order to alert MySQL Administrators about any issues with MySQL applications the following configuration can be used:

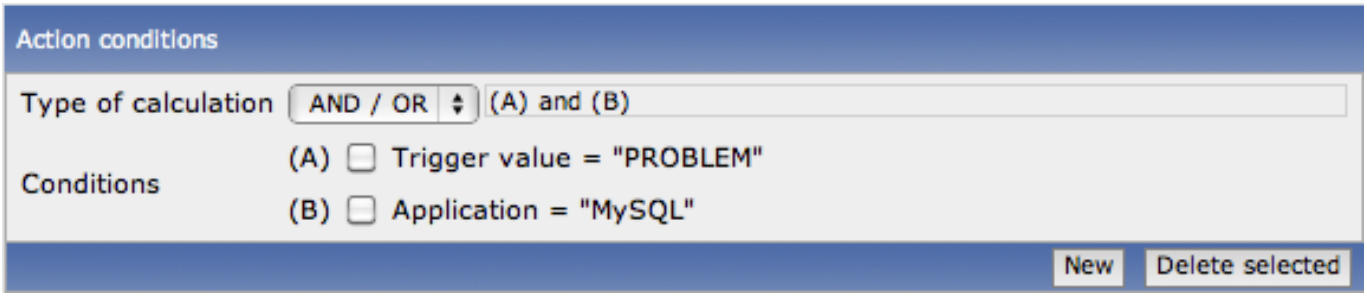



Since we are not interested in sending multiple messages or escalating MySQL problems to other user groups, escalations are not enabled.

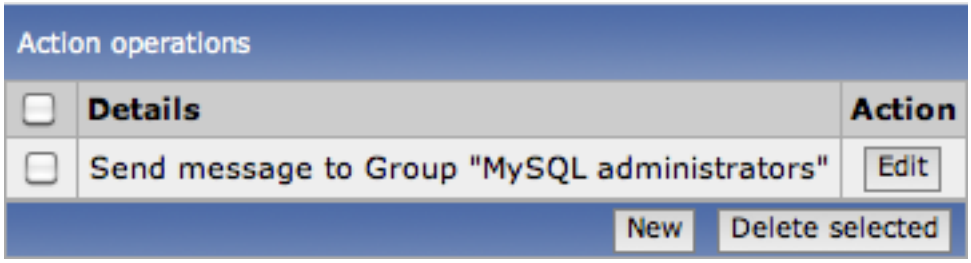
Zabbix will send a single message to MySQL Administrators and a recovery message when problem is resolved. If sending of recovery messages is not enabled, Zabbix will send only one message with information about new problem, no messages will be sent on recovery, i.e. when the problem is resolved.

Action conditions is defined so that it will be activated in case of any problem with any of MySQL applications.

Note also use of macros in the messages. Zabbix supports wide range of macros. Complete list of macros is available here: macros




Actions are defined as:




A message will be sent to all members of the group MySQL Administrators.

**3 Remote commands**

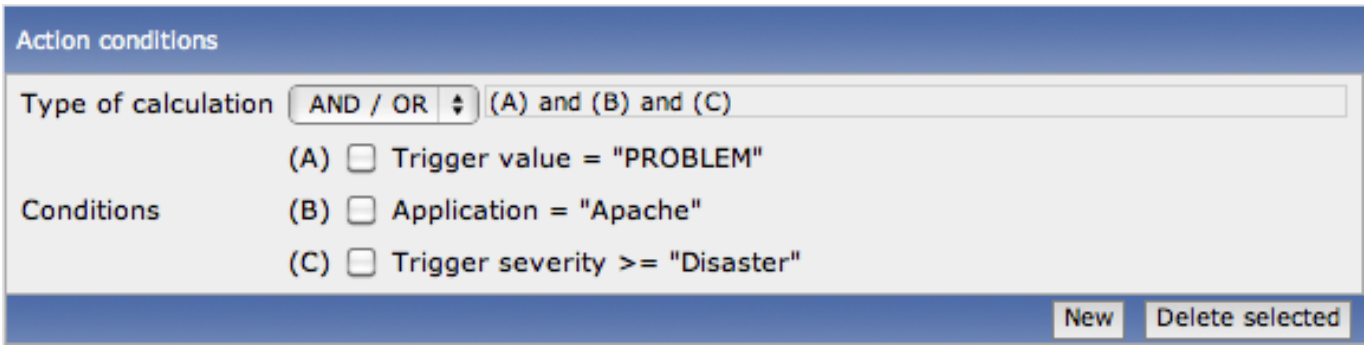
Remote commands is a powerful mechanism for smart pro-active monitoring. Zabbix can execute a command on a monitored host in case of any pre-defined conditions.

Here is the list of some of the most obvious uses of the feature:

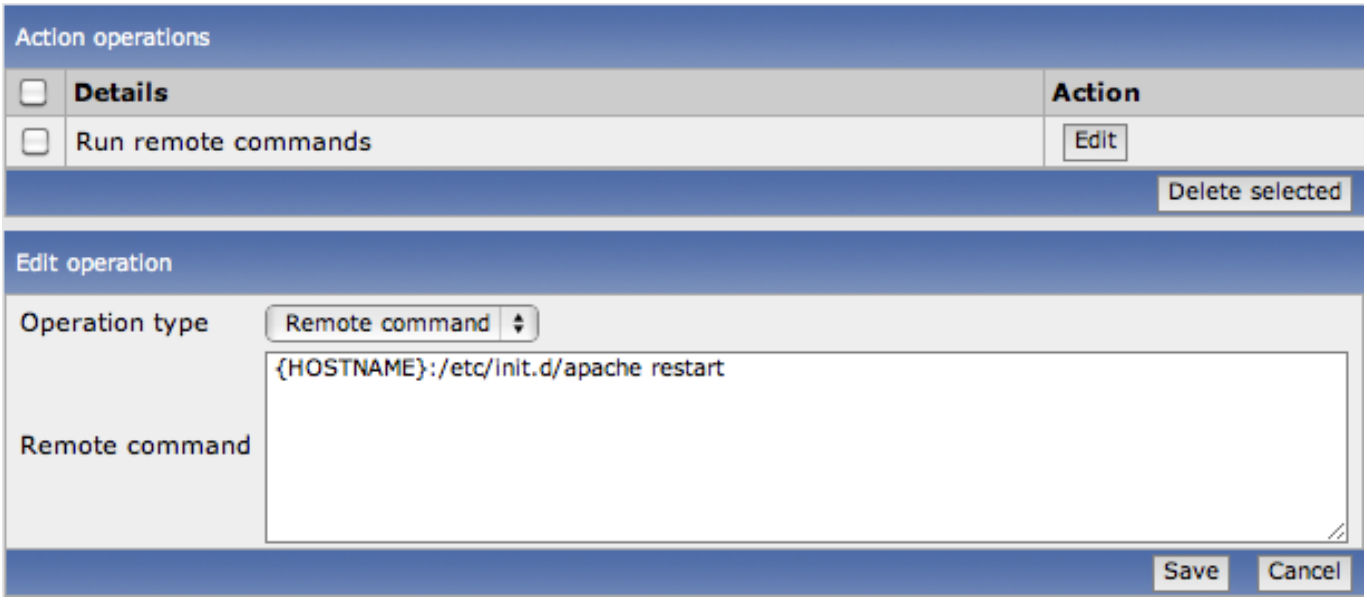
- Automatically restart application (WEB server, middleware, CRM) if it does not respond
- Using IPMI 'reboot' command reboot remote server if it does not answer requests
- Try to automatically free disk space (remove older files, clean /tmp) if we are running out of disk space
- Migrate one VM from one physical box to another depending on CPU load
- Add new nodes to the cloud environment if we have insufficient CPU (disk, memory, whatever) resources

Configuration of action for remote commands is similar to messaging, the only difference is that Zabbix will execute a command instead of sending a message.

The action condition is defined so that it will be activated in case of any disaster problems with one of Apache applications.




As a reaction to the disaster problem Zabbix will try to restart Apache process:




Note use of the macro {HOSTNAME} here.

**Note:**
User 'zabbix' must have enough permissions to execute this script. Also Zabbix agent should run on a remote host and accept incoming connections. Remote commands are disabled by default and can be enabled in Zabbix agent daemon configuration file on Unix-like or Windows systems.

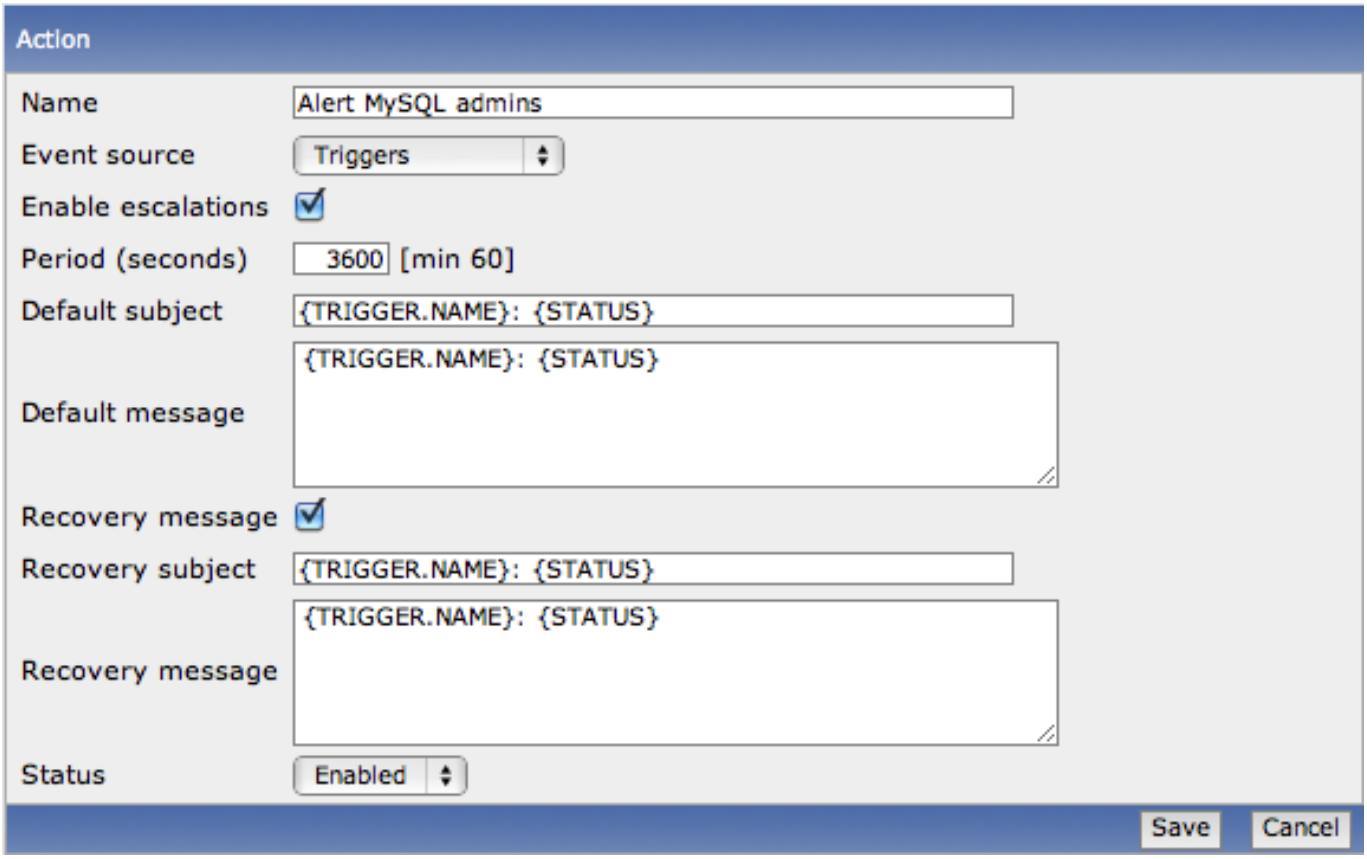
**Attention:**
Remote commands do not work with active Zabbix agents.

See remote command tutorial for more information.

#### 4 Repeated notifications

Repeated notifications is probably one of the most common use of Zabbix escalations.

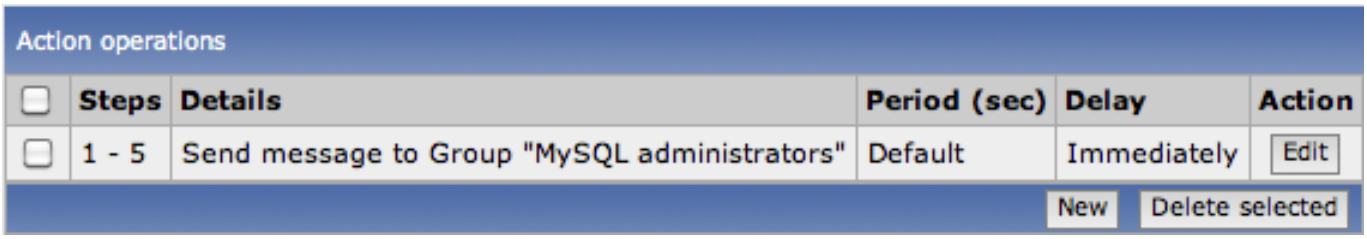
Make sure that escalations are enabled in the action details:




The period defines how frequently Zabbix should increase escalation step. By default, it goes to the next step every hour, i.e. 3600 seconds.

As soon as we enabled escalations, actions operations get additional options: Step(s), Period and Conditions.

Suppose we would like to send 5 messages every hour, so we defined that the operation will be active from escalation step 1 till 5. The escalation period will be taken from action definition unless we overwrote it for an individual operation.




As soon as we have a problem, Zabbix is at step 1, so all operations assigned to the step will be executed. After one hour, escalation period will be increased automatically (if the problem still exists obviously), so all operations of step 2 will be execute. And so on.

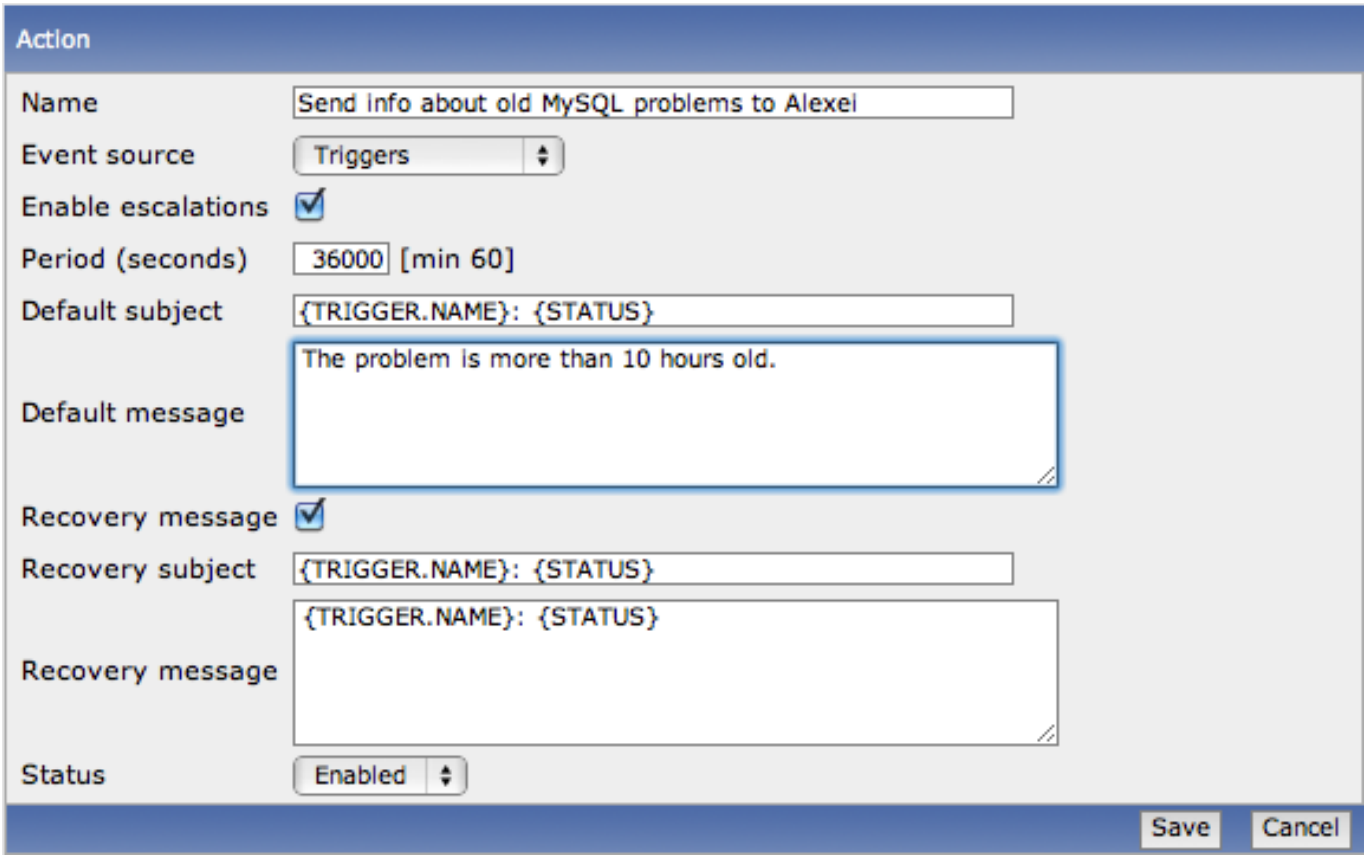
A recovery message will be sent only to those people who received at least one message before in scope of the escalation.

**Note:**
If the trigger that generated an active escalation is disabled, Zabbix sends a message informing about this fact to persons that have already received notifications.

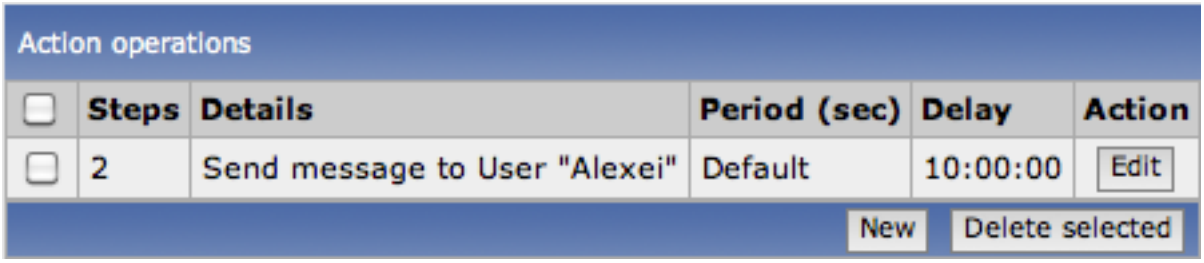
#### 5 Delayed notifications

Zabbix escalations supports sending of delayed notifications.

Suppose we would like to be notified about long-standing MySQL problems only. Note that the escalation period was changed to 10 hours and we use a custom default message:




The operation is assigned only to step 2. It means it will be executed once after one escalation period, i.e. 10 hours:

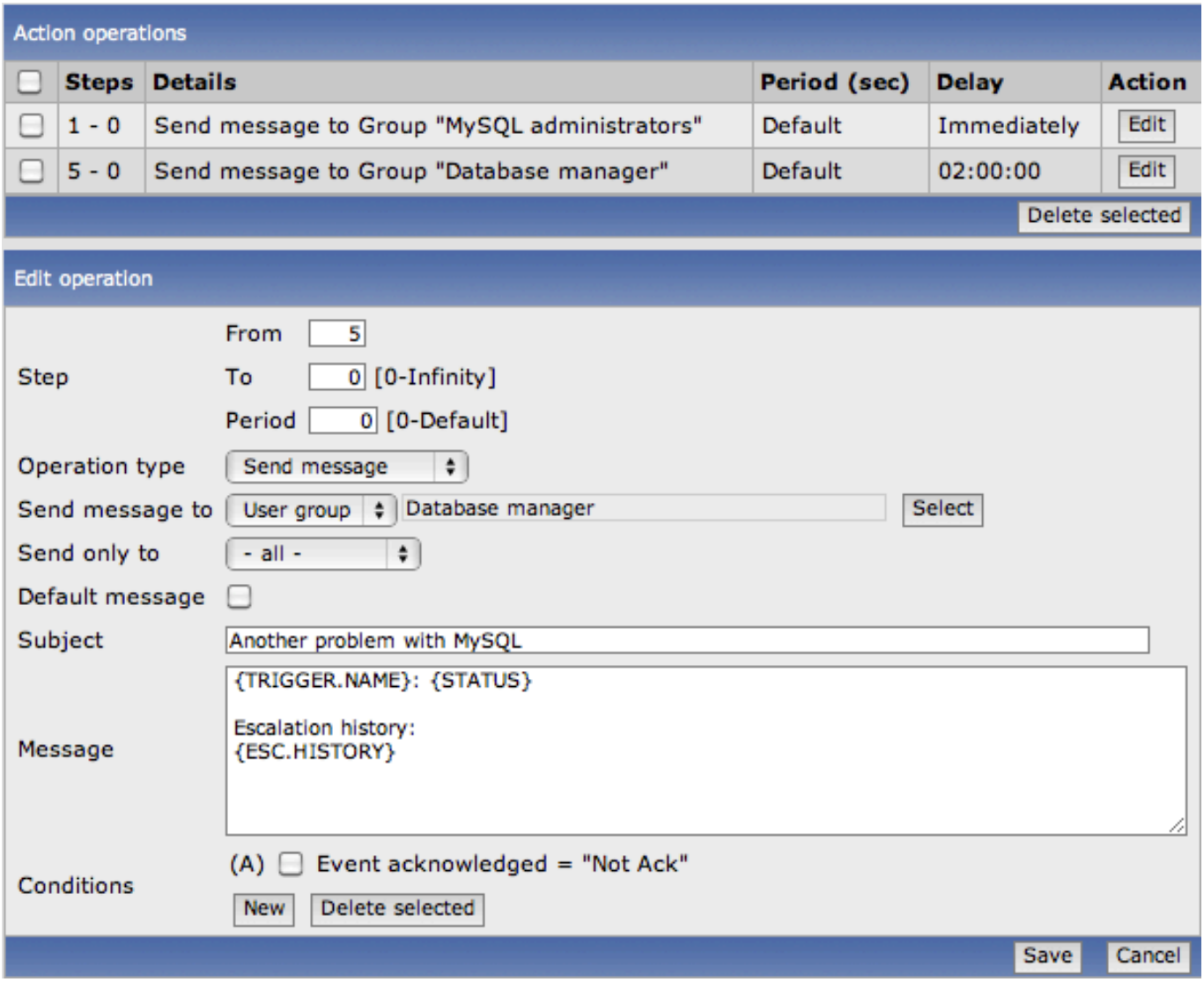



Therefore user 'Alexei' will get a message only in case if a problem exists for more than 10 hours. The notification delay is controlled by the escalation period.

**6 Escalate to Boss**

Zabbix escalations can be used to escalate problem to other users and user groups. Problem is not being fixed by MySQL admins? Escalate to their BOSS!

Now we configured periodical sending of messages to MySQL administrators. 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 if the problem is not acknowledged yet, supposedly no one is working on it.



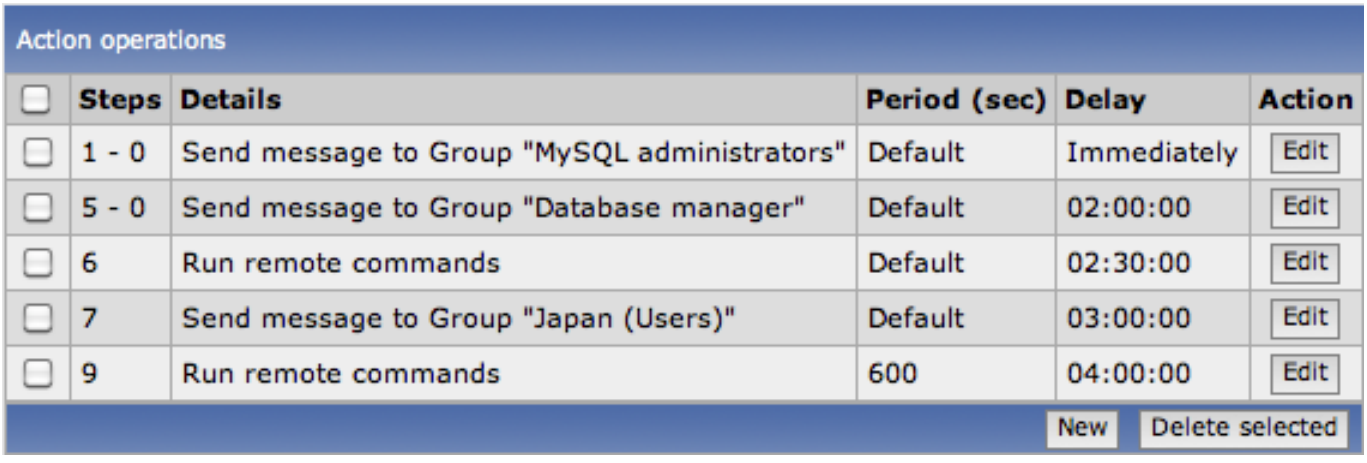

Note use of the {ESC.HISTORY} macros in the message. The macro will contain information about all previously executed steps. The manager will get information about all email and all action executed before. MySQL administrators, beware!

**7 Complex scenario**

Look at this set of actions. After multiple messages to MySQL administrators and escalation to the manager, Zabbix will try to restart the MySQL database. It will happen if problem exists for 2:30 hours and it hasn't been acknowledged.

If the problems still exists, after another 30 minutes Zabbix will send a message to all users in Japan.

If this does help, after another hour Zabbix will reboot server with the MySQL database (second remote command) using IPMI commands.



# Zabbix API

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

Currently Zabbix API specification is in draft state. All objects marked as 'draft' are experimental and should be used with a great care. We do not guarantee compatibility with future releases.

Object specifications without the 'draft' mark are stable and can be used for production purposes.

See Zabbix wiki for community provided solutions around the API.

## Action

**Methods** Class containing methods for operations with Actions.

| Methods | Description |
|---|---|
| get() | Get action details |
| exists() | Check if action exists |
| create() | Create actions |
| update() | Update action details |
| delete() | Delete actions |

**Object details** The table contains complete list of Action attributes.

**Action**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **actionid** | *integer* | Action ID | |
| **name** | *string* | Name | |
| **eventsource** | *integer* | Event source | Triggers / Discovery / Auto registration |
| **evaltype** | *integer* | Height | |
| **status** | *integer* | Status | Enabled/Disabled |
| **esc_period** | *integer* | Default escalation period | |
| **def_shortdata** | *string* | Default message subject | |
| **def_longdata** | *string* | Default message | |
| **recovery_msg** | *integer* | Send recovery message | On/Off |
| **r_shortdata** | *string* | Default recovery message subject | |
| **r_longdata** | *string* | Default message subject | |

**Conditions**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **conditionid** | *integer* | Condition ID | |
| **actionid** | *integer* | Action ID | |
| **conditiontype** | *integer* | Condition type | |
| **operator** | *integer* | Comparision type | |
| **value** | *string* | Condition value | |

**Operations**

| | | |
|---|---|---|
| **operationid** | *integer* | Condition ID |
| **actionid** | *integer* | Action ID |
| **operationtype** | *integer* | Condition type |
| **object** | *integer* | Comparision type |
| **objectid** | *integer* | Condition value |
| **shortdata** | *string* | Custom message subject |
| **longdata** | *string* | Custom message |

| | | | |
|---|---|---|---|
| **esc_period** | *integer* | Custom escalation period | |
| **esc_step_from** | *integer* | Escalation step start from | |
| **esc_step_to** | *integer* | Escalation step end on | |
| **default_msg** | *integer* | Use default mesages | On/Off |
| **evaltype** | *string* | Default recovery message subject | |

**Operation media types**

| | | |
|---|---|---|
| **opmediatypeid** | *integer* | Operation media type ID |
| **operationid** | *integer* | Operation ID |
| **mediatypeid** | *integer* | Media type ID |

**Operation conditions**

| | | |
|---|---|---|
| **opconditionid** | *integer* | Operation Condition ID |
| **operationid** | *integer* | Operation ID |
| **conditiontype** | *integer* | Condition Type |
| **operator** | *integer* | Operator |
| **vaelue** | *integer* | Value |

**Common tasks** The table contains list of common action-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add an action | Use method **action.create** |
| Add a bunch of new actions | Use method **action.create** with array of Action objects |
| Remove action by Action IDs | Use method **action.delete** array of Action **IDs** |
| Retrieve action details by Action IDs | Use method **action.get** with parameter **actionids** |
| Retrieve action details by Action name | Use method **action.get** with parameter **filter**, specify **”name”:”<your action>”** |

**create()**

This function allows you to create a action as defined by the **action data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **action data** | *array or object* | | Array of Action objects with additional paramters **operations** and **conditions** | **actionid** shouldn’t be specified |
| **action conditions data** | *array or object* | | array of action conditions objects | |
| **action operations data** | *array or object* | | array of action operations objects | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Action IDs. **actionid** are assigned to each Action object |
| **error** | In case of any errors |

**Example** Create new action

```
{
"jsonrpc":"2.0",
"method":"action.create",
"params":[{
    "name":"ZABBIX ACTION",
    "eventsource":"0",
    "evaltype":"0",
    "status":"1",
    "esc_period":"3600",
    "def_shortdata":"{TRIGGER.NAME}: {STATUS}",
    "def_longdata":"{TRIGGER.NAME}: {STATUS}",
    "recovery_msg":"0",
    "r_shortdata":"{TRIGGER.NAME}: {STATUS}",
    "r_longdata":"{TRIGGER.NAME}: {STATUS}",
    "conditions":[{
        "conditiontype":"3",
        "operator":"2",
        "value":"TEST"
    },{
        "conditiontype":"1",
        "operator":"1",
        "value":"100100000010096"
    }],
    "operations":[{
        "operationtype":"0",
        "object":"0",
        "objectid":"100100000000017",
        "shortdata":"{TRIGGER.NAME}: {STATUS}",
        "longdata":"{TRIGGER.NAME}: {STATUS}",
        "esc_period":"0",
        "esc_step_from":"1",
        "esc_step_to":"1",
        "default_msg":"1",
        "evaltype":"0",
        "opconditions":[],
        "opmediatypes":[]
    },{
        "operationtype":"0",
        "object":"0",
        "objectid":"100100000000001",
        "shortdata":"{TRIGGER.NAME}: {STATUS}",
        "longdata":"{TRIGGER.NAME}: {STATUS}",
        "esc_period":"0",
        "esc_step_from":"2",
        "esc_step_to":"3",
        "default_msg":"1",
        "evaltype":"0",
        "opconditions":[],
        "opmediatypes":[{
            "mediatypeid":"100100000000001"
        }]
    },{
        "operationtype":"0",
        "object":"0",
        "objectid":"100100000000003",
        "shortdata":"{TRIGGER.NAME}: {STATUS}",
        "longdata":"{TRIGGER.NAME}: {STATUS}",
        "esc_period":"0",
        "esc_step_from":"3",
        "esc_step_to":"4",
        "default_msg":"1",
```

```
        "evaltype":"0",
        "opconditions":[{
            "conditiontype":"14",
            "operator":"0",
            "value":"0"
        },{
            "conditiontype":"14",
            "operator":"0",
            "value":"1"
        }],
        "opmediatypes":[{
            "mediatypeid":"100100000000001"
        }]
    }]
}],
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":2
}
```

Action created successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "actionids":["100100000012213"]
},
"id":2
}
```

Action already exists:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CAction::create ] Action [ ZABBIX Server ] already exists"
},
"id":2
}
```

**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several actions. Action items will be removed.

**Parameters** Array of Action IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Action IDs. |
| **error** | In case of any errors |

**Example** Delete actions by action **ID**

```
{
"jsonrpc":"2.0",
"method":"action.delete",
"params":["107824",  "107825"],
"auth":"3a57200802b24cda67c4e4010b50c065",
```

```
"id":2
}
```

Actions deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "actionids": ["107824", "107825"]
},
"id":2
}
```

Actions does not exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CAction::delete ] Action does not exist"
},
"id":2
}
```

**exists()**

Available since version: **1.8.3**
This function allows you to check whether action with given action name or action ID exists.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **nodeids** | *array* | yes | List of node IDs where to search for given action ID or action name | |
| **actionid** | *string* | yes | Action ID | |
| **name** | *string* | yes | Action name | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain boolean variable. |
| **error** | In case of any errors |

**Example**

```
{
"jsonrpc":"2.0",
"method":"action.exists",
"params":{
    "nodeids": ["1"],
    "name": "Zabbix Server Action"
},
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Action exists:

```
{
"jsonrpc":"2.0",
"result": true,
"id":2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve Action details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **groupids** | *array* | Group IDs | |
| **hostids** | *array* | Host IDs | |
| **triggerids** | *array* | Trigger IDs | |
| **actionids** | *array* | Action IDs | |
| **mediatypeids** | *array* | Media type IDs | |
| **userids** | *array* | User IDs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by action fields | |
| **search** | *array* | Return actions by any given action object field pattern | |
| **startSearch** | *integer* | Search actions field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, actions by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_conditions** | *string* | Select action conditions | Values: shorten, refer, extend |
| **select_operations** | *string* | Select action operations | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count actions, return the number of actions found | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by Action field | Values: actionid, name |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *integer* | max number of Action objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Action objects. |
| **error** | In case of any errors |

**Example** Get actions details by Action name ”zabbix”, with action elements and links:

```
{
"jsonrpc":"2.0",
"method":"action.get",
"params":{
```

```
    "filter": {"name": "ZABBIX Action"},
    "select_operations": "extend",
    "select_conditions": "extend",
    "output": "extend"
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved Action details:

```
{
"jsonrpc":"2.0",
"result":[{
    "actionid":"100100000000013",
    "name":"ZABBIX Action",
    "eventsource":"0",
    "evaltype":"0",
    "status":"1",
    "esc_period":"3600",
    "def_shortdata":"{TRIGGER.NAME}: {STATUS}",
    "def_longdata":"{TRIGGER.NAME}: {STATUS}",
    "recovery_msg":"0",
    "r_shortdata":"{TRIGGER.NAME}: {STATUS}",
    "r_longdata":"{TRIGGER.NAME}: {STATUS}",
    "conditions":[{
        "conditionid":"100100000000097",
        "actionid":"100100000000013",
        "conditiontype":"3",
        "operator":"2",
        "value":"Server"
    },{
        "conditionid":"100100000000098",
        "actionid":"100100000000013",
        "conditiontype":"1",
        "operator":"1",
        "value":"100100000010096"
    }],
    "operations":[{
        "operationid":"100100000000082",
        "actionid":"100100000000013",
        "operationtype":"0",
        "object":"0",
        "objectid":"100100000000001",
        "shortdata":"{TRIGGER.NAME}: {STATUS}",
        "longdata":"{TRIGGER.NAME}: {STATUS}",
        "esc_period":"0",
        "esc_step_from":"1",
        "esc_step_to":"3",
        "default_msg":"1",
        "evaltype":"0",
        "opconditions":[],
        "opmediatypes":[{
            "opmediatypeid":"100100000000004",
            "operationid":"100100000000082",
            "mediatypeid":"100100000000001"
        }]
        },{
        "operationid":"100100000000083",
        "actionid":"100100000000013",
        "operationtype":"0",
        "object":"0",
        "objectid":"100100000000003",
        "shortdata":"{TRIGGER.NAME}: {STATUS}",
```

```
        "longdata":"{TRIGGER.NAME}: {STATUS}",
        "esc_period":"0",
        "esc_step_from":"3",
        "esc_step_to":"4",
        "default_msg":"1",
        "evaltype":"0",
        "opconditions":[{
            "opconditionid":"100100000000001",
            "operationid":"100100000000083",
            "conditiontype":"14",
            "operator":"0",
            "value":"0"
        },{
            "opconditionid":"100100000000002",
            "operationid":"100100000000083",
            "conditiontype":"14",
            "operator":"0",
            "value":"1"
        }],
        "opmediatypes":[{
            "opmediatypeid":"100100000000005",
            "operationid":"100100000000083",
            "mediatypeid":"100100000000001"
        }]
    }]
}]
 ,
"id":2
}
```


**update()**

Available since version: **1.8**
The method is used to control all action attributes including action conditions and operations.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **actionid** | *string* | | Action ID. | |
| **action attribute** | *any* | Yes | New value for an action attribute. | |
| **action conditions data** | *array or object* | | array of action conditions objects | |
| **action operations data** | *array or object* | | array of action operations objects | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated action IDs. |
| **error** | In case of any errors |

**Example** Set action name to ”New Name”:

```
{
 "jsonrpc": "2.0",
 "method": "action.update",
 "params": {
```

```
        "actionid": "100100000010092",
        "name": "New Name"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 2
}
```

Retrieved updated action IDs:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": ["100100000010092"]
    },
    "id": 2
}
```

**Example #2** Disable action:

```
{
 "actionid": "1",
 "status": "0",
}
```

Parameter ”status” serves for setting status of your action.

## Alert

**Methods** Class containing methods for operations with Alerts.

| Methods | Description |
|---|---|
| get() | Get alert details |

**Object details** The table contains complete list of Alert attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| **alertid** | *integer* | Alert ID | |
| **actionid** | *integer* | Action ID | |
| **eventid** | *integer* | Event ID | |
| **userid** | *integer* | User ID | |
| **clock** | *integer* | Date | Unix timestamp |
| **mediatypeid** | *integer* | Madia type ID | |
| **sendto** | *string* | Address | |
| **subject** | *string* | Alert subject | |
| **message** | *string* | Alert message | |
| **status** | *integer* | Alert status | |
| **retries** | *integer* | Retries made to send | |
| **error** | *string* | Error details in case if sending failed | |

**Common tasks** The table contains list of common alert-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Remove a bunch of alerts | Use method **alert.delete** with array of Alert objects |
| Retrieve alert details by Alert IDs | Use method **alert.get** with parameter **alertids** |
| Retrieve alerts details by some period | Use method **alert.get** with parameters **time_from** and **time_till** |

**get()**

Available since version: **1.8**
This function allows you to retrieve alert details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **groupids** | *array* | HostGroup IDs | |
| **hostids** | *array* | Host IDs | |
| **alertids** | *array* | Alert IDs | |
| **triggerids** | *array* | Trigger IDs | |
| **eventids** | *array* | Event IDs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **time_from** | *integer* | Search alerts from given date | Unix timestamp |
| **time_till** | *integer* | Search alerts till given date | Unix timestamp |
| **filter** | *array* | Optional filter by alert fields | |
| **search** | *array* | Return alerts by any given alert object field pattern | |
| **startSearch** | *integer* | Search alerts field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, alerts by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_hosts** | *string* | Select hosts | Values: shorten, refer, extend |
| **select_mediatypes** | *string* | Select mediatypes | Values: shorten, refer, extend |
| **select_users** | *string* | Select users | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count alerts, return number of alerts found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by alert field | Values: alertid,clock,eventid,status |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of alert objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Alert objects. |
| **error** | In case of any errors |

**Example** Get alerts details by trigger **IDs** and limit output to 10 alerts, return only **alert** IDS:

```
{
"jsonrpc":"2.0",
"method":"alert.get",
"params":{
```

```
    "output":"shorten",
    "triggerids": ["100100000010137", "100100000010138"],
    "time_from": 1285077093,
    "time_till": 1285107165,
    "limit": 10
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved alerts details:

```
{
"jsonrpc":"2.0",
"result":[
    {"alertid":"100100000010048"},
    {"alertid":"100100000010137"},
    {"alertid":"100100000017431"},
    {"alertid":"100100000017533"},
    {"alertid":"100100000017635"},
    {"alertid":"100100000017737"},
    {"alertid":"100100000017839"},
    {"alertid":"100100000017941"},
    {"alertid":"100100000018043"},
    {"alertid":"100100000018145"}
],
"id":2
}
```

## APIInfo

Class contains methods for getting information about Zabbix and Zabbix API.

**Methods**

| Methods | Description |
|---|---|
| version() | Get Zabbix API version |

**Common tasks** The table contains list of common apiinfo-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Get API version | Use method **apiinfo.version** |

**version()**

Available since version: **1.8.1**
Get Zabbix API version.

**Parameters** This method does not accept any parameters.

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain API version string. |
| **error** | In case of any errors |

**Example** Get Zabbix API version:

```
{
"jsonrpc":"2.0",
"method":"apiinfo.version",
"params":[],
"auth":"a6e895b98fde40f4f7badf112fd983bf",
"id":2
}
```

Retrieved API version:

```
{
"jsonrpc":"2.0",
"result":"1.3",
"id":2
}
```

## Application

**Methods** Class containing methods for operations with Applications.

| Methods | Description |
|---|---|
| get() | Get application details |
| exists() | Check if application exists |
| create() | Create applications |
| update() | Update application details |
| delete() | Delete applications |

**Object details** The table contains complete list of Application attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| applicationid | *int* | Application ID | |
| hostid | *int* | Host ID | |
| name | *string* | Application description | |
| templateid | *int* | Parent application ID | |

**Common tasks** The table contains list of common application-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add an application | Use method **application.create** |
| Add a bunch of new applications | Use method **application.create** with array of Application objects |
| Rename an application | Use method **application.update**, set **”name”:1”<new name>”** |
| Delete an application | Use method **application.delete** |
| Retrieve application details by Application IDs | Use method **application.get** with parameter **applicationids** |
| Retrieve applications details by Application name | Use method **application.get** with parameter **filter**, specify **”name”: [”<your application1>”, ”<your application2>”]** |

### create()

This function allows you to create a application as defined by the **application data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **application data** | *array or object* | | Array of Application objects or a single object | **applicationid** shouldn't be specified |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Application IDs. **applicationid** are assigned to each Application object |
| **error** | In case of any errors |

**Example** Create new application for host with Host **ID** "100100000010048"

```
{
"jsonrpc":"2.0",
"method":"application.create",
"params":[{
      "name": "SNMP Items",
      "hostid": "100100000010048",
}],
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":2
}
```

Application created successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "applicationids": ["100100000214797"]
},
"id":2
}
```

Application already exists:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CApplication::create ] Cannot create Application"
},
"id":2
}
```

**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several applications.

**Parameters** Array of Application IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Application IDs. |
| **error** | In case of any errors |

**Example** Delete applications by application **ID**

```
{
"jsonrpc":"2.0",
"method":"application.delete",
"params":["107824",  "107825"],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Applications deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "applicationids": ["107824", "107825"]
},
"id":2
}
```

Applications does not exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CApplication::delete ] Application does not exist"
},
"id":2
}
```

**exists()**

Available since version: **1.8.3**
This function allows you to check whether application with given application data exists.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **nodeids** | *array* | yes | List of node IDs where to search for given application | |
| **name** | *string* | No | Application name | |
| **hostid** | *string* | yes | Host ID | |
| **host** | *string* | yes | Host name | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain boolean variable. |
| **error** | In case of any errors |

**Example** Check if application with name ”OS” exists for host ”Windows-Server”

```
{
"jsonrpc":"2.0",
"method":"application.exists",
"params":{
    "host": "Windows-Server",
    "name": "OS"
},
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Application exists:

```
{
"jsonrpc":"2.0",
"result": true,
"id":2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve application details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **groupids** | *array* | HostGroup IDs | |
| **hostids** | *array* | Host IDs | |
| **templateids** | *array* | Template IDs | |
| **itemids** | *array* | Item IDs | |
| **applicationids** | *array* | Application IDs | |
| **inherited** | *integer* | Inherited from templates | ”0” - not inherited, ”1” - inherited |
| **templated** | *integer* | Templated applications | ”0” - belongs to hosts, ”1” - belongs to templates |
| **monitored** | *integer* | Monitored applications | Checks application and host status |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by application fields | |
| **search** | *array* | Return applications by given application fields pattern | |
| **startSearch** | *integer* | Search given pattern only in start of the fields | |
| **excludeSearch** | *integer* | Exclude from result applications by given pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **expandData** | *string* | Output additional fields | Adds host name to output objects |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_hosts** | *string* | Select hosts | Values: shorten, refer, extend |
| **select_items** | *string* | Select items | Values: shorten, refer, extend |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **countOutput** | *integer* | Count applications, return the number of applications found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by application field | Values: applicationid,description,key_,delay,history,tren |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of application objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Application objects. |
| **error** | In case of any errors |

**Example** Get applications details by application name pattern ”file” and limit output to 2 applications, expand data:

```
{
"jsonrpc":"2.0",
"method":"application.get",
"params":{
    "search": {"name": "file"},
    "output": "extend",
    "expandData": 1,
    "limit": 2
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved applications details:

```
{
"jsonrpc":"2.0",
"result":[
    {
        "hosts":[{
            "hostid":"100100000010097"
        }],
        "applicationid":"100100000000572",
        "name":"Filesystem",
        "templateid":"100100000000005",
        "host":"192.168.3.1"
    },
    {
        "hosts":[{
            "hostid":"100100000010097"
        }],
        "applicationid":"100100000000575",
        "name":"Log files",
        "templateid":"100100000000011",
        "host":"192.168.3.1"
    }
],
"id":2
}
```

**massAdd()**

Available since version: **1.8**

**Parameters** Multidimensional array with Item applications data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **applications** | *array* | | Item application to update. | |
| **items** | *array* | Yes | Item objects that should be added to item applications. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Item application IDs. |
| **error** | In case of any errors |

**Example** Add two items with **ID** ”100100000010092”, ”100100000010086” to two applications with **ID** ”100100000000042”, ”100100000000013”

```
{
"jsonrpc":"2.0",
"method":"application.massAdd",
"params":{
    "applications": [
      {"applicationid": "100100000000042"},
      {"applicationid": "100100000000013"}
    ],
    "items": [
        {"itemid": "100100000010092"},
        {"itemid": "100100000010086"}
    ]
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

Item applications updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "applicationids":["100100000000042","100100000000013"]
},
"id":2
}
```

**update()**

Available since version: **1.8**
The method is used to control all application attributes including application applications linkage.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **applicationid** | *string* | | Application ID. | |
| **application attribute** | *any* | Yes | New value for a application attribute. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Application IDs. |
| **error** | In case of any errors |

**Example** Rename application:

```
{
 "jsonrpc":"2.0",
 "method":"application.update",
 "params":{
    "applicationid": "100100000010092",
    "name": "IPMI Items"
 },
 "auth":"700ca65537074ec963db7efabda78259",
 "id":2
}
```

Retrieved updated application IDs:

```
{
 "jsonrpc":"2.0",
 "result": {
     "applicationids":["100100000010092"]
 },
 "id":2
}
```

## DCheck

**Methods** Class containing methods for operations with Discovered Checks.

| Methods | Description |
|---|---|
| get() | Get discovery check details |

**Object details** The table contains complete list of Discovery Check attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| **dcheckid** | *integer* | Discovery Check ID | |
| **druleid** | *integer* | Discovery Rule ID | |
| **type** | *integer* | Check type | |
| **key_** | *string* | ZABBIX Agent key | |
| **ports** | *string* | Port | Separated by comma |
| **snmp_community** | *string* | SNMP community | |
| **snmpv3_securityname** | *string* | SNMP | |
| **snmpv3_securitylevel** | *integer* | SNMP security level | |
| **snmpv3_authpassphrase** | *string* | SNMP authentication phrase | |
| **snmpv3_privpassphrase** | *string* | SNMP Private phrase | |

**Common tasks** The table contains list of common discovery check-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Retrieve discovery check details by Discovery Rule IDs | Use method **dcheck.get** with parameter **dcheckids** |

**get()**

Available since version: **1.8**
This function allows you to retrieve discovery check details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **druleids** | *array* | Discovery rule IDs | |
| **dhostids** | *array* | Discovery host IDs | |
| **dcheckids** | *array* | Discovery check IDs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by discovery check fields | |
| **search** | *array* | Return discovery checks by any given discovery check object field pattern | |
| **startSearch** | *integer* | Search discovery checks field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, discovery checks by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **selectDRules** | *string* | Select discovery rules | Values: shorten, refer, extend |
| **selectDHosts** | *string* | Select discovery hosts | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count discovery checks, return the number of discovery checks found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by discovery check field | Values: dcheckid |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *integer* | max number of discovery check objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Discovery check objects. |
| **error** | In case of any errors |

**Example** Get discovery checks details by discovery rule ID:

```
{
"jsonrpc":"2.0",
```

```
"method":"dcheck.get",
"params":{
    "druleids": ["100100000000003"],
    "output": "extend",
    "limit": 3
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved discovery check details:

```
{
"jsonrpc":"2.0",
"result":[{
    "drules":[{"druleid":"100100000000003"}],
    "dcheckid":"100100000000037",
    "druleid":"100100000000003",
    "type":"4",
    "key_":"",
    "snmp_community":"",
    "ports":"80",
    "snmpv3_securityname":"",
    "snmpv3_securitylevel":"0",
    "snmpv3_authpassphrase":"",
    "snmpv3_privpassphrase":""
}],
"id":2
}
```

## DHost

**Methods** Class containing methods for operations with Discovered Hosts.

| Methods | Description |
|---|---|
| get() | Get discovery host details |
| delete() | Delete discovery hosts |

**Object details** The table contains complete list of Discovery Host attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| **dhostid** | *integer* | Discovery Host ID | |
| **druleid** | *integer* | Discovery Rule ID | |
| **status** | *integer* | Host Status | |
| **lastup** | *integer* | Last UP state date | Unix timestamp |
| **lastdown** | *integer* | Last DOWN state date | Unix timestamp |

**Common tasks** The table contains list of common discovery host-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Retrieve discovery host details by Discovery Rule IDs | Use method **dhost.get** with parameter **dhostids** |

**delete()**

**get()**

Available since version: **1.8**
This function allows you to retrieve discovery host details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **druleids** | *array* | Discovery rule IDs | |
| **dhostids** | *array* | Discovery host IDs | |
| **dserviceids** | *array* | Discovery service IDs | |
| **groupids** | *array* | Host group IDs | |
| **hostids** | *array* | Host IDs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by discovery host fields | |
| **search** | *array* | Return discovery hosts by any given discovery host object field pattern | |
| **startSearch** | *integer* | Search discovery hosts field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, discovery hosts by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **selectDRules** | *string* | Select discovery ruless | Values: shorten, refer, extend |
| **selectDChecks** | *string* | Select discovery checks | Values: shorten, refer, extend |
| **selectDServices** | *string* | Select discovery services | Values: shorten, refer, extend |
| **selectGroups** | *string* | Select groups | Values: shorten, refer, extend |
| **selectHosts** | *string* | Select hosts | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count discovery hosts, return the number of discovery hosts found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by discovery host field | Values: dhostid |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of discovery host objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Discovery host objects. |
| **error** | In case of any errors |

**Example** Get discovery hosts details by discovery rule ID:

```
{
"jsonrpc":"2.0",
"method":"dhost.get",
"params":{
    "druleids": ["100100000000003"],
    "output": "extend",
    "limit": 3
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved discovery host details:

```
{
"jsonrpc":"2.0",
"result":[{
    "drules":[{
        "druleid":"100100000000003"
    }],
    "dhostid":"100100000000002",
    "druleid":"100100000000003",
    "status":"0",
    "lastup":"1245250108",
    "lastdown":"0"
},{
    "drules":[{
        "druleid":"100100000000003"
    }],
    "dhostid":"100100000000003",
    "druleid":"100100000000003",
    "status":"0",
    "lastup":"1245250130",
    "lastdown":"0"
},{
    "drules":[{
        "druleid":"100100000000003"
    }],
    "dhostid":"100100000000004",
    "druleid":"100100000000003",
    "status":"0",
    "lastup":"1245250131",
    "lastdown":"0"
}],
"id":2
}
```

## DRule

**Methods** Class containing methods for operations with Discovery Rules.

| Methods | Description |
|---|---|
| get() | Get discovery rule details |
| exists() | Check if discovery rule exists |
| create() | Create discovery rules |
| update() | Update discovery rule details |
| delete() | Delete discovery rules |

**Object details** The table contains complete list of Discovery Rule attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| druleid | *integer* | Discovery Rule ID | |
| proxy_hostid | *integer* | Proxy Host ID. | |
| name | *string* | Discovery Rule name. | |
| iprange | *string* | Ip range. | |
| delay | *integer* | Delay between checks | |
| nextcheck | *integer* | Next check date | Zabbix internal field |
| status | *integer* | Status | |
| unique_dcheckid | *string* | Unique discovery check ID | |

**Common tasks** The table contains list of common discovery rule-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add a bunch of new discovery rules | Use method **discovery rule.create** with array of Discovery Rule objects |
| Enable a discovery rule | Use method **drule.update**, set **”status”:0** |
| Disable a discovery rule | Use method **drule.update**, set **”status”:1** |
| Retrieve discovery rule details by Discovery Rule IDs | Use method **drule.get** with parameter **druleids** |
| Retrieve discovery rule details by Discovery Rule name | Use method **drule.get** with parameter **filter**, specify **”name”:”<your discovery rule>”** |

**create()**

**delete()**

**exists()**

**get()**

Available since version: **1.8**
This function allows you to retrieve discovery rule details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **druleids** | *array* | Discovery rule IDs | |
| **dhostids** | *array* | Discovery host IDs | |
| **dserviceids** | *array* | Discovery service IDs | |
| **dcheckids** | *array* | Discovery check IDs | |
| **editable** | *integer* | only with read-write permission.<br>Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by discovery rule fields | |
| **search** | *array* | Return discovery rules by any given discovery rule object field pattern | |
| **startSearch** | *integer* | Search discovery rules field pattern only in start of the field | |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **excludeSearch** | *integer* | Exclude from result, discovery rules by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **selectDHosts** | *string* | Select discovery hosts | Values: shorten, refer, extend |
| **selectDChecks** | *string* | Select discovery checks | Values: shorten, refer, extend |
| **selectDServices** | *string* | Select discovery services | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count discovery rules, return the number of discovery rules found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by discovery rule field | Values: druleid, name |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of discovery rule objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Discovery rule objects. |
| **error** | In case of any errors |

**Example** Get discovery rules details by discovery rule name pattern ”local”:

```
{
"jsonrpc":"2.0",
"method":"drule.get",
"params":{
    "search": {"name": "local"},
    "output": "extend"
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved discovery rule details:

```
{
"jsonrpc":"2.0",
"result":[{
    "druleid":"100100000000003",
    "proxy_hostid":"0",
    "name":"Local network",
    "iprange":"192.168.0.1-255",
    "delay":"3600",
    "nextcheck":"1280502905",
    "status":"0",
    "unique_dcheckid":"0"
}],
"id":2
}
```

**update()**

## DService

**Methods** Class containing methods for operations with Discovered Services.

| Methods | Description |
|---|---|
| get() | Get discovery service details |

**Object details** The table contains complete list of Discovery Service attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| **dserviceid** | *integer* | Discovery Service ID | |
| **dhostid** | *integer* | Discovery Host ID | |
| **dcheckid** | *integer* | Discovery Check ID | |
| **type** | *integer* | Service type | |
| **key_** | *string* | ZABBIX Agent key | |
| **port** | *integer* | Port | |
| **value** | *integer* | Value | |
| **status** | *integer* | Status | |
| **lastup** | *integer* | Last date of UP state | |
| **lastdown** | *integer* | Last date of DOWN state | |
| **ip** | *string* | Discovered host IP | |

**Common tasks** The table contains list of common discovery service-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Retrieve discovery service details by Discovery Rule IDs | Use method **dservice.get** with parameter **dserviceids** |

**create()**

**delete()**

**exists()**

**get()**

Available since version: **1.8**
This function allows you to retrieve discovery service details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **druleids** | *array* | Discovery rule IDs | |
| **dhostids** | *array* | Discovery host IDs | |
| **dcheckids** | *array* | Discovery check IDs | |
| **dserviceids** | *array* | Discovery service IDs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by discovery service fields | |
| **search** | *array* | Return discovery services by any given discovery service object field pattern | |
| **startSearch** | *integer* | Search discovery services field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, discovery services by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **selectDRules** | *string* | Select discovery rules | Values: shorten, refer, extend |
| **selectDChecks** | *string* | Select discovery checks | Values: shorten, refer, extend |
| **selectDHosts** | *string* | Select discovered hosts | Values: shorten, refer, extend |
| **selectHosts** | *string* | Select hosts | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count discovery hosts, return the number of discovery services found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by discovery service field | Values: dserviceid, dhostid |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of discovery service objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Discovery service objects. |
| **error** | In case of any errors |

**Example** Get discovery services details by discovery rule ID:

```
{
"jsonrpc":"2.0",
"method":"dservice.get",
"params":{
    "druleids": ["100100000000003"],
    "output": "extend",
    "limit": 2
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved discovery services details:


```
{
"jsonrpc":"2.0",
"result":[{
    "drules":[{
        "druleid":"100100000000003"
    }],
    "dhosts":[{
        "dhostid":"100100000000036"
    }],
    "dchecks":[{
        "dcheckid":"100100000000037"
    }],
    "dserviceid":"100100000000042",
    "dhostid":"100100000000036",
    "type":"4",
    "key_":"",
    "value":"",
    "port":"80",
    "status":"0",
    "lastup":"1252320879",
    "lastdown":"0",
    "dcheckid":"100100000000037",
    "ip":"192.168.3.1",
    "druleid":"100100000000003"
},{
    "drules":[{
        "druleid":"100100000000003"
    }],
    "dhosts":[{
        "dhostid":"100100000000037"
    }],
    "dchecks":[{
        "dcheckid":"100100000000037"
    }],
    "dserviceid":"100100000000043",
    "dhostid":"100100000000037",
    "type":"4",
    "key_":"",
    "value":"",
    "port":"80",
    "status":"1",
    "lastup":"0",
    "lastdown":"1271859008",
    "dcheckid":"100100000000037",
    "ip":"192.168.3.2",
    "druleid":"100100000000003"
}],
"id":2
}
```


**update()**

## Event

**Methods** Class containing methods for operations with Events.

| Methods | Description |
|---|---|
| get() | Get event details |
| acknowledge() | Acknowledge events |
| delete() | Delete events |

**Object details** The table contains complete list of Event attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| **eventid** | *integer* | Event ID | |
| **source** | *integer* | Event generation source | |
| **object** | *integer* | Event relation object | |
| **objectid** | *integer* | Related object ID | |
| **clock** | *integer* | Time of generated event | |
| **value** | *integer* | Status | |
| **acknowledged** | *integer* | Flag indicating event ack | |

**Field values**

**Source**

| Value | Type |
|---|---|
| 0 | Triggers |
| 1 | Network discovery |

**Value** For triggers

| Value | Type |
|---|---|
| 0 | OK |
| 1 | PROBLEM |
| 2 | UNKNOWN |

For network discovery

| Value | Type |
|---|---|
| 0 | UP |
| 1 | DOWN |
| 2 | Discovered |
| 3 | Lost |

**Common tasks** The table contains list of common event-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Acknowledge an event | Use method **event.acknowledge** with array of Event objects |
| Remove events | Use method **event.delete** with array of Event **IDs** |
| Retrieve event details by Event IDs | Use method **event.get** with parameter **eventids** |
| Retrieve events details by Trigger IDs | Use method **event.get** with parameter **triggerids** |

**acknowledge()**

Available since version: **1.8**
The method is used to control all event attributes including event applications linkage.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **eventids** | *string* | No | Array of event IDs. | |
| **message** | *any* | Yes | Acknowledge message. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of acknowledged Event IDs. |
| **error** | In case of any errors |

**Example** Acknowledge two events and leave a message:

```
{
 "jsonrpc":"2.0",
 "method":"event.acknowledge",
 "params":{
    "eventids": ["100100000010092", "100100000010094"],
    "message": "Problem resolved"
 },
 "auth":"700ca65537074ec963db7efabda78259",
 "id":2
}
```

Retrieved acknowledged event IDs:

```
{
 "jsonrpc":"2.0",
 "result": {
     "eventids":["100100000010092", "100100000010094"]
 },
 "id":2
}
```

**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several events. All event-related information will be removed including sent alerts.

**Parameters** Array of Event IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Event IDs. |
| **error** | In case of any errors |

**Example** Delete events by event **ID**

```
{
"jsonrpc":"2.0",
"method":"event.delete",
"params":["107824",  "107825"],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Events deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "eventids": ["107824", "107825"]
},
"id":2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve event details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **groupids** | *array* | Host group IDs | |
| **hostids** | *array* | Host IDs | |
| **triggerids** | *array* | Trigger IDs | |
| **eventids** | *array* | Event IDs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **object** | *integer* | Event object | |
| **value** | *integer* | Event status | |
| **source** | *integer* | Event source | |
| **acknowledged** | *integer* | Acknowledged events | |
| **hide_unknown** | *integer* | Hide unknown events | |
| **time_from** | *integer* | Events since specified date | Unix timestamp |
| **time_till** | *integer* | Events till specified date | Unix timestamp |
| **eventid_from** | *integer* | Events with IDs greater or equal than specified | |
| **eventid_till** | *integer* | Events with IDs less or equal than specified | |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_hosts** | *string* | Select hosts | Values: shorten, refer, extend |
| **select_items** | *string* | Select items | Values: shorten, refer, extend |
| **select_triggers** | *string* | Select event triggers | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count events, return the number of events found | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by event field | Values: eventid,clock |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of event objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Event objects. |
| **error** | In case of any errors |

**Example** Get last 10 events for specified period:

```
{
"jsonrpc": "2.0",
"method": "event.get",
"params": {
    "time_from": "1284910040",
    "time_till": "1284991200",
    "output": "extend",
    "sortfield": "clock",
    "sortorder": "desc",
    "limit": 10
},
"auth": "6f38cddc44cfbb6c1bd186f9a220b5a0",
"id": 2
}
```

Retrieved events details:

```
{
"jsonrpc": "2.0",
"result": [{
    "eventid": "100100000884382",
    "source": "0",
    "object": "0",
    "objectid": "100100000064507",
    "clock": "1284910089",
    "value": "1",
    "acknowledged": "0"
},{
    "eventid": "100100000884383",
    "source": "0",
    "object": "0",
    "objectid": "100100000064661",
    "clock": "1284910089",
    "value": "1",
    "acknowledged": "0"
},{
    "eventid": "100100000884385",
    "source": "0",
    "object": "0",
    "objectid": "100100000064661",
    "clock": "1284910118",
    "value": "0",
    "acknowledged": "0"
},{
    "eventid": "100100000884384",
    "source": "0",
    "object": "0",
    "objectid": "100100000064507",
    "clock": "1284910119",
    "value": "0",
    "acknowledged": "0"
},{
    "eventid": "100100000884386",
    "source": "0",
    "object": "0",
    "objectid": "100100000064661",
    "clock": "1284910176",
    "value": "1",
    "acknowledged": "0"
},{
    "eventid": "100100000884387",
    "source": "0",
```

```
    "object": "0",
    "objectid": "100100000064661",
    "clock": "1284910206",
    "value": "0",
    "acknowledged": "0"
},{
    "eventid": "100100000884388",
    "source": "0",
    "object": "0",
    "objectid": "100100000064661",
    "clock": "1284910326",
    "value": "1",
    "acknowledged": "0"
},{
    "eventid": "100100000884389",
    "source": "0",
    "object": "0",
    "objectid": "100100000064509",
    "clock": "1284910351",
    "value": "1",
    "acknowledged": "0"
},{
    "eventid": "100100000884390",
    "source": "0",
    "object": "0",
    "objectid": "100100000064546",
    "clock": "1284910351",
    "value": "1",
    "acknowledged": "0"
},{
    "eventid": "100100000884391",
    "source": "0",
    "object": "0",
    "objectid": "100100000012788",
    "clock": "1284910353",
    "value": "1",
    "acknowledged": "0"
}],
"id": 2
}
```


## Graph

**Methods** Class containing methods for operations with Graphs.

| Methods | Description |
|---|---|
| get() | Get graph details |
| exists() | Check if graph exists |
| create() | Create graphs |
| update() | Update graph details |
| delete() | Delete graphs |

**Object details** The table contains complete list of Graph attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| graphid | *integer* | Graph ID | |
| name | *string* | Graph name. | |

| Parameter | Type | Description | Details |
|---|---|---|---|
| width | *integer* | Width. | |
| height | *integer* | Height. | |
| yaxismin | *integer* | Y axis min value. | |
| yaxismax | *integer* | Y axis max value. | |
| templateid | *integer* | Parent graph **ID**. | |
| show_work_period | *integer* | Show work period. | |
| show_triggers | *integer* | Show items triggers if possible | |
| graphtype | *integer* | Chart or Pie. | |
| show_legend | *integer* | Show legend for pie graphs. | |
| show_3d | *integer* | Show pie graph in 3D view. | |
| percent_left | *float* | Show percentile line (left). | |
| percent_right | *float* | Show percentile line (right). | |
| ymin_type | *integer* | Y axis min limitation type. | Calculated, user defined, by item value. |
| ymax_type | *integer* | Y axis max limitation type. | Calculated, user defined, by item value. |
| ymin_itemid | *integer* | Y axis min limitation by Item **ID**. | |
| ymax_itemid | *integer* | Y axis max limitation by Item **ID**. | |

**Field values**

**Y axis min/max type**

| Value | Type |
|---|---|
| 0 | Calculated |
| 1 | Fixed |
| 2 | By item value |

**Common tasks** The table contains list of common graph-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add a graph | Use method **graph.create** |
| Add a bunch of new graphs | Use method **graph.create** with array of Graph objects |
| Remove graph by Graph IDs | Use method **graph.delete** array of Graph **IDs** |
| Retrieve graph details by Graph IDs | Use method **graph.get** with parameter **graphids** |
| Retrieve graph details by Graph name | Use method **graph.get** with parameter **filter**, specify **"name":"<your graph>"** |

**create()**

This function allows you to create a graph as defined by the **graph data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **graph data** | *array or object* | | Array of Graph objects or a single object with additional parameter **gitems**, array of graph item objects | **graphid** shouldn't be specified |

**Note:**
See also graph item details.

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Graph IDs. **graphid** are assigned to each Graph object |
| **error** | In case of any errors |

**Example** Create new graph

```
{
"jsonrpc":"2.0",
"method":"graph.create",
"params":[{
    "gitems":[{
        "itemid":"100100000018469",
        "drawtype":"0",
        "sortorder":"0",
        "color":"999900",
        "yaxisside":"0",
        "calc_fnc":"2",
        "type":"0",
        "periods_cnt":"5"
        },{
        "itemid":"100100000018468",
        "drawtype":"0",
        "sortorder":"1",
        "color":"009900",
        "yaxisside":"0",
        "calc_fnc":"2",
        "type":"0",
        "periods_cnt":"5"
        },{
        "itemid":"100100000018467",
        "drawtype":"0",
        "sortorder":"2",
        "color":"990000",
        "yaxisside":"0",
        "calc_fnc":"2",
        "type":"0",
        "periods_cnt":"5"
    }],
    "name":"CPU Loads",
    "width":"900",
    "height":"200",
    "yaxismin":"0.0000",
    "yaxismax":"3.0000",
    "templateid":"0",
    "show_work_period":"1",
    "show_triggers":"1",
    "graphtype":"0",
    "show_legend":"0",
    "show_3d":"0",
    "percent_left":"0.0000",
    "percent_right":"0.0000",
    "ymin_type":"0",
    "ymax_type":"0",
    "ymin_itemid":"0",
    "ymax_itemid":"0"
}],
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":2
}
```

Graph created successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "graphids":["100100000012213"]
},
"id":2
}
```

Graph already exists:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CGraph::create ] Graph already exists [ CPU Loads ] on Host [ ZABBIX-Server ]"
},
"id":2
}
```

**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several graphs. Graph items will be removed.

**Parameters** Array of Graph IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Host IDs. |
| **error** | In case of any errors |

**Example** Delete graphs by graph **ID**

```
{
"jsonrpc":"2.0",
"method":"graph.delete",
"params":["107824",  "107825"],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Graphs deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "graphids": ["107824", "107825"]
},
"id":2
}
```

Graphs does not exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CGraph::delete ] Graph does not exist"
```

```
},
"id":2
}
```

**exists()**

Available since version: **1.8.3**
This function allows you to check whether graph with given graph data exists.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **nodeids** | *array* | yes | List of node IDs where to search for given graph | |
| **name** | *string* | No | Graph name | |
| **hostid** | *string* | yes | Host ID | |
| **host** | *string* | yes | Host name | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain boolean variable. |
| **error** | In case of any errors |

**Example** Check if graph with name ”CPU Loads” exists for host ”ZABBIX-Server”

```
{
"jsonrpc":"2.0",
"method":"graph.exists",
"params":{
    "host": "ZABBIX-Server",
    "name": "CPU Loads"
},
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Graph exists:

```
{
"jsonrpc":"2.0",
"result": true,
"id":2
}
```

**get()**

Available since version: **1.8**
This method allows you to retrieve graph details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **groupids** | *array* | HostGroup IDs | |
| **templateids** | *array* | Template IDs | |
| **hostids** | *array* | Host IDs | |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **graphids** | *array* | Graph IDs | |
| **itemids** | *array* | Item IDs | |
| **type** | *integer* | Graph type | |
| **inherited** | *integer* | Inherited from templates | ”0” - not inherited, ”1” - inherited |
| **templated** | *integer* | Templated items | ”0” - belongs to hosts, ”1” - belongs to templates |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by graph fields | |
| **search** | *array* | Return graphs by any given object field pattern | |
| **startSearch** | *integer* | Search graphs field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, graphs by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_groups** | *string* | Select host groups | Values: shorten, refer, extend |
| **select_templates** | *string* | Select host templates | Values: shorten, refer, extend |
| **select_hosts** | *string* | Select hosts | Values: shorten, refer, extend |
| **select_items** | *string* | Select host items | Values: shorten, refer, extend |
| **select_graph_items** | *string* | Select graph items | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count graphs, return the number of graphs found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by graph field | Values: graphid, name |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of graph objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Graph objects. |
| **error** | In case of any errors |

**Example** Get graphs details for graphs containing ”CPU” in their names for host ”ZABBIX-Server” and limit output to two graphs:

```
{
   "jsonrpc":"2.0",
   "method":"graph.get",
   "params":{
      "output":"extend",
      "search":{
         "name":"CPU"
      },
      "filter":{
         "host":[
            "Zabbix-server"
```

```
         ]
      },
      "limit":2
   },
   "auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
   "id":2
}
```


Retrieved graph details:


```
{
   "jsonrpc":"2.0",
   "result":[
      {
         "graphid":"100100000000589",
         "name":"CPU Loads 2",
         "width":"900",
         "height":"400",
         "yaxismin":"0.0000",
         "yaxismax":"100.0000",
         "templateid":"0",
         "show_work_period":"1",
         "show_triggers":"0",
         "graphtype":"0",
         "show_legend":"0",
         "show_3d":"0",
         "percent_left":"0.0000",
         "percent_right":"0.0000",
         "ymin_type":"0",
         "ymax_type":"0",
         "ymin_itemid":"0",
         "ymax_itemid":"0"
      },
      {
         "graphid":"100100000006093",
         "name":"CPU Loads",
         "width":"900",
         "height":"400",
         "yaxismin":"0.0000",
         "yaxismax":"100.0000",
         "templateid":"0",
         "show_work_period":"1",
         "show_triggers":"0",
         "graphtype":"0",
         "show_legend":"0",
         "show_3d":"0",
         "percent_left":"0.0000",
         "percent_right":"0.0000",
         "ymin_type":"0",
         "ymax_type":"1",
         "ymin_itemid":"0",
         "ymax_itemid":"0"
      }
   ],
   "id":2
}
```


**update()**

Available since version: **1.8**
The method is used to control all graph attributes including graph graphs.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **graphid** | *string* | | Graph ID. | |
| **graph attribute** | *any* | Yes | New value for a graph attribute. | |
| **gitems** | *any* | Yes | New graph item list. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Graph IDs. |
| **error** | In case of any errors |

**Example** Set graph y axis max value to '100':

```
{
 "jsonrpc":"2.0",
 "method":"graph.update",
 "params":{
    "graphid": "100100000010092",
    "ymax_type": 1,
    "yaxismax": 100
 },
 "auth":"700ca65537074ec963db7efabda78259",
 "id":2
}
```

Retrieved updated graph IDs:

```
{
 "jsonrpc":"2.0",
 "result": {
     "graphids":["100100000010092"]
 },
 "id":2
}
```

## Graphitem

**Methods** Class containing methods for operations with Graphitems.

| Methods | Description |
|---|---|
| get() | Get graphitem details |

**Object details** The table contains complete list of Graphitem attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| gitemid | *integer* | Graphitem ID | |
| graphid | *integer* | Graph ID | |
| itemid | *integer* | Item ID | |
| drawtype | *integer* | Draw type | Line(0), filled region(1), bold line(2), dot(3), dashed(4), gradient(5) |
| sortorder | *integer* | Position in legend | |
| color | *string* | Draw color | Hex RGB |
| yaxisside | *integer* | Graph Y axis side | left (0), right (1) |

| Parameter | Type | Description | Details |
|---|---|---|---|
| calc_fnc | *integer* | Graph item data selection function | min(1), max(4), avg(2), all(7) |
| type | *integer* | Graph item type | simple (0), aggregated (1), graph sum (2) - used in Pie and Exploded graph types |
| periods_cnt | *integer* | Aggregated periods count | |

**Common tasks** The table contains list of common graphitem-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Get a graphitem | Use method **graphitem.get** |

**get()**

Available since version: **1.8**
This function allows you to retrieve graph details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **graphids** | *array* | Graph IDs | |
| **itemids** | *array* | Item IDs | |
| **type** | *integer* | Graph item type | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **expandData** | *string* | Add additional fields to result | Values: shorten, refer, extend |
| **select_graphs** | *string* | Select graphs | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count graphs items, return the number of graph items found | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by graph item field | Values: gitemid |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of graph item objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of graph objects. |
| **error** | In case of any errors |

**Example** Get graph items details by graph ID:

```
{
"jsonrpc":"2.0",
"method":"graphitem.get",
"params":{
```

```
    "graphids": ["100100000012214"],
    "output": "extend"
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved graph item details:

```
{
"jsonrpc":"2.0",
"result":
[{
    "graphs":[{
        "graphid":"100100000012214"
    }],
    "gitemid":"100100000025237",
    "graphid":"100100000012214",
    "itemid":"100100000018506",
    "drawtype":"0",
    "sortorder":"0",
    "color":"009900",
    "yaxisside":"1",
    "calc_fnc":"2",
    "type":"0",
    "periods_cnt":"5"
},{
    "graphs":[{
        "graphid":"100100000012214"
    }],
    "gitemid":"100100000025240",
    "graphid":"100100000012214",
    "itemid":"100100000018529",
    "drawtype":"0",
    "sortorder":"6",
    "color":"660066",
    "yaxisside":"1",
    "calc_fnc":"2",
    "type":"0",
    "periods_cnt":"5"
}],
"id":2
}
```

## History

**Methods** Class containing methods for operations with History.

| Methods | Description |
|---|---|
| get() | Get history details |
| delete() | Delete items history |

**Object details** The table contains complete list of History attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| **id** | *integer* | History ID | Not all history tables got this field |
| **itemid** | *integer* | Item ID | |
| **clock** | *integer* | Unix timestamp. | |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **value** | *integer* | Item value. | |

**Common tasks** The table contains list of common history-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Get some item history | Use method **history.get** |
| Delete items history | Use method **history.delete** with array of item **IDs** |

**delete()**

**get()**

Available since version: **1.8.3**
This method allows you to retrieve history details based on filtering options.

All parameters are optional except "history". If parameter is set in query, this option is considered as being ON, except if parameter is equal to NULL.

Getting values for multiple items of different types (**history** parameter) is not supported at this time.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **history** | *array* | Item value type | |
| **nodeids** | *array* | Node IDs | |
| **hostids** | *array* | Host IDs | |
| **triggerids** | *array* | Template IDs | |
| **itemids** | *array* | Item IDs | |
| **time_from** | *integer* | Select data from | Unix timestamp |
| **time_till** | *integer* | Select data till | Unix timestamp |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by history fields | |
| **search** | *array* | Return history by any given history object field pattern | |
| **startSearch** | *integer* | Search history field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, history by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count history values, return the number of values found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **groupOutput** | *integer* | Group result by passed IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by history field | Values: itemid, clock |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of history objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of History objects. |
| **error** | In case of any errors |

**Example** Get history details for numeric(float) item with **ID** "100100000018467" starting from "1284387605" (13.09.2010 17:23:39) till "1284387846" (13.09.2010 17:24:06)

```
{
"jsonrpc":"2.0",
"method":"history.get",
"params":{
    "history": 0,
    "itemids": ["100100000018467"],
    "time_from": "1284387605"
    "time_till": "1284387846"
    "output":"extend"
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved history details:

```
{
"jsonrpc":"2.0",
"result":[{
    "itemid":"100100000018467",
    "clock":"1284387607",
    "value":"0.9300"
},{
    "itemid":"100100000018467",
    "clock":"1284387627",
    "value":"0.9200"

},{
    "itemid":"100100000018467",
    "clock":"1284387807",
    "value":"0.9400"
},{
    "itemid":"100100000018467",
    "clock":"1284387827",
    "value":"0.9300"
}],
"id":2
}
```

## Host

**Methods** Class containing methods for operations with Hosts.

| Methods | Description |
|---|---|
| get() | Get host details |
| exists() | Check if host exists |
| create() | Create hosts |
| update() | Update host details |
| delete() | Delete hosts |
| massAdd() | Mass add template linkage, macros, host groups |

| Methods | Description |
|---|---|
| massUpdate() | Mass update host details, link templates, add host groups |
| massRemove() | Mass remove template linkage, macros, host groups |

**Object details** The table contains complete list of Host attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| hostid | *int* | Host ID | |
| host | *string* | Host name. | |
| port | *int* | Port number. | |
| status | *int* | Host Status. | |
| useip | *int* | Use IP. | |
| dns | *string* | DNS. | |
| ip | *string* | IP. | |
| proxy_hostid | *int* | Proxy Host ID. | |
| useipmi | *int* | Use IPMI. | |
| ipmi_ip | *string* | IPMAI IP. | |
| ipmi_port | *int* | IPMI port. | |
| ipmi_authtype | *int* | IPMI authentication type. | |
| ipmi_privilege | *int* | IPMI privilege. | |
| ipmi_username | *string* | IPMI username. | |
| ipmi_password | *string* | IPMI password. | |

**Common tasks** The table contains list of common host-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add a host | Use method **host.create** |
| Add a bunch of new hosts | Use method **host.create** with array of Host objects |
| Enable a host | Use method **host.update**, set **”status”:0** |
| Disable a host | Use method **host.update**, set **”status”:1** |
| Retrieve host details by Host IDs | Use method **host.get** with parameter **hostids** |
| Retrieve host details by Host name | Use method **host.get** with parameter **filter**, specify **”host”:”<your host>”** |

**create()**

This method allows you to create a host as defined by the **host data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **host data** | *array or object* | No | Array of Host objects or a single object | **hostid** shouldn’t be specified |
| **groups** | *array* | No | HostGroup Objects add Host to. | |
| **templates** | *array* | No | Templates Objects link Host to. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Host IDs. **hostid** are assigned to each Host object |
| **error** | In case of any errors |

```
{
"jsonrpc":"2.0",
"method":"host.create",
"params":{
      "host":"Linux001",
      "ip":"192.168.3.1",
      "port":10050,
      "useip":1,
      "groups":[
         {
            "groupid":50
         }
      ],
      "templates":[
         {
            "templateid":20045
         }
      ]
   },
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":3
}
```

**Example** Host added successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "hostids": ["107819"]
},
"id":3
}
```

Host already exists:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CHost::create ] Host [ Linux001 ] already exists"
},
"id":3
}
```

**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several hosts. All host-related information will be removed including items, graphs, macros, application, historical data, etc.

**Parameters** Array of Host IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Host IDs. |
| **error** | In case of any errors |

**Example** Delete hosts by Host **ID**

```
{
"jsonrpc":"2.0",
"method":"host.delete",
"params":[
      {
         "hostid":"107824"
      },
      {
         "hostid":"107825"
      }
],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Hosts deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "hostids": ["107824", "107825"]
},
"id":2
}
```

Host does not exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CHost::delete ] Host does not exist"
},
"id":2
}
```

**exists()**

Available since version: **1.8.3**
This function allows you to check whether host with given host name or host ID exists.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **nodeids** | *array* | yes | List of node IDs where to search for given host ID or host name | |
| **hostid** | *string* | yes | Host ID | |
| **host** | *string* | yes | Host name | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain boolean variable. |
| **error** | In case of any errors |

**Example**

```
{
"jsonrpc":"2.0",
"method":"host.exists",
"params":{
    "nodeids": ["1"],
    "host": "ZABBIX-Server"
},
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Host exists:

```
{
"jsonrpc":"2.0",
"result": true,
"id":2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve host details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **groupids** | *array* | HostGroup IDs | |
| **hostids** | *array* | Host IDs | |
| **templateids** | *array* | Template IDs | |
| **itemids** | *array* | Item IDs | |
| **triggerids** | *array* | Trigger IDs | |
| **graphids** | *array* | Graph IDs | |
| **proxyids** | *array* | Proxy IDs | |
| **maintenanceids** | *array* | Maintenance IDs | |
| **dhostids** | *array* | Discovered host IDs | |
| **dserviceids** | *array* | Discovered services IDs | |
| **monitored_hosts** | *integer* | return only monitored Hosts | |
| **templated_hosts** | *integer* | include templates in result | |
| **proxy_hosts** | *integer* | return only Proxies | |
| **with_items** | *integer* | only with items | |
| **with_monitored_items** | *integer* | only with monitored items | |
| **with_historical_items** | *integer* | only with historical items | |
| **with_triggers** | *integer* | only with triggers | |
| **with_monitored_triggers** | *integer* | only with monitored triggers | |
| **with_httptests** | *integer* | only with http tests | |
| **with_monitored_httptests** | *integer* | only with monitored http tests | |
| **with_graphs** | *integer* | only with graphs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by host fields | |
| **search** | *array* | Return hosts by any given host object field pattern | |
| **startSearch** | *integer* | Search hosts field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, hosts by given field pattern | |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_groups** | *string* | Select host groups | Values: shorten, refer, extend |
| **selectParentTemplates** | *string* | Select host templates | Values: shorten, refer, extend |
| **select_items** | *string* | Select host items | Values: shorten, refer, extend |
| **select_triggers** | *string* | Select host triggers | Values: shorten, refer, extend |
| **select_graphs** | *string* | Select host graphs | Values: shorten, refer, extend |
| **select_dhosts** | *string* | Select host related discovery hosts | Values: shorten, refer, extend |
| **select_dservices** | *string* | Select host related discovery services | Values: shorten, refer, extend |
| **select_applications** | *string* | Select host applications | Values: shorten, refer, extend |
| **select_macros** | *string* | Select host macros | Values: shorten, refer, extend |
| **select_profile** | *string* | Select host profile | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count hosts, return the number of hosts found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by host field | Values: hostid,host,status,dns,ip |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of host objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Host objects. |
| **error** | In case of any errors |

**Example** Get hosts details by host name ”Zabbix-server”,”Zabbix-server TEST”:

```
{
"jsonrpc":"2.0",
"method":"host.get",
"params":{
    "output":"extend",
    "filter":{
        "host":["Zabbix-server","Zabbix-server TEST"]
        }
    },
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved host details:

```
{
"jsonrpc":"2.0",
"result":[{
```

```
    "maintenances":[{
        "maintenanceid":"0"
    }],
    "hostid":"100100000010017",
    "proxy_hostid":"0",
    "host":"ZABBIX-Server",
    "dns":"ip4-dm",
    "useip":"1",
    "ip":"192.168.3.4",
    "port":"31055",
    "status":"0",
    "disable_until":"0",
    "error":"",
    "available":"1",
    "errors_from":"0",
    "lastaccess":"0",
    "inbytes":"0",
    "outbytes":"0",
    "useipmi":"0",
    "ipmi_port":"623",
    "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_ip":"",
    "ipmi_errors_from":"0",
    "snmp_errors_from":"0",
    "ipmi_error":"",
    "snmp_error":""
},{
    "maintenances":[{
        "maintenanceid":"0"
    }],
    "hostid":"100100000010229",
    "proxy_hostid":"0",
    "host":"ZABBIX-Server TEST",
    "dns":"ip4-dm",
    "useip":"1",
    "ip":"192.168.3.4",
    "port":"31055",
    "status":"0",
    "disable_until":"0",
    "error":"",
    "available":"1",
    "errors_from":"0",
    "lastaccess":"0",
    "inbytes":"0",
    "outbytes":"0",
    "useipmi":"0",
    "ipmi_port":"623",
    "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_ip":"",
    "ipmi_errors_from":"0",
    "snmp_errors_from":"0",
    "ipmi_error":"",
    "snmp_error":""
}],
"id":2
}
```


**massAdd()**

Available since version: **1.8**
==== Parameters ==== multidimensional array with Hosts, Groups, Template, Macros data
Only host details can be added this way, not hosts themselves.

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **hosts** | *array* | | Host objects to update | |
| **groups** | *array* | Yes | Host group objects where hosts should be added. | |
| **templates** | *array* | Yes | Template objects which should be linked to hosts. | |
| **macros** | *array* | Yes | Macros objects which should be added to hosts. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Host IDs. |
| **error** | In case of any errors |

**Example** Add two macros to host with **ID** ”100100000010092”

```
{
"jsonrpc":"2.0",
"method":"host.massAdd",
"params":{
"hosts":
 {
  "hostid": "100100000010092"
 },
"macros": [
  {
    "macro": "{$TEST1}",
    "value": "MACROTEST1"
  },
  {
    "macro": "{$TEST2}",
    "value": "MACROTEST2"
  }
]
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

Hosts updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "hostids":["100100000010092"]
},
"id":2
}
```

**massRemove()**

Available since version: **1.8**
==== Parameters ==== multidimensional array with Hosts data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **hostids** | *array* | | Hostids to update | |
| **groupids** | *array* | Yes | Host groupids where hosts should be removed. | |
| **templateids** | *array* | Yes | Templateid which should be unlinked from hosts. | |
| **macros** | *array* | Yes | Macros which should be removed from hosts. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Host IDs. |
| **error** | In case of any errors |

**Example** Remove from host with **ID** ”100100000010092” two macros {$TEST1},{$TEST2}

```
{
"jsonrpc":"2.0",
"method":"host.massRemove",
"params":{
    "hostids": ["100100000010092"],
    "macros": ["{$TEST1}","{$TEST2}"]
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

Hosts updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
  "hostids":["100100000010092"]
},
"id":2
}
```

**massUpdate()**

Available since version: **1.8**
==== Parameters ==== multidimensional array with Hosts data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **hosts** | *array* | | Host objects to update | |
| **host** | *string* | | Host name. | |
| **groupids** | *array* | | Host group IDs to add host to. | |
| **port** | *int* | Yes | Port. | |
| **status** | *int* | Yes | Host Status. | |
| **useip** | *int* | Yes | Use IP. | |
| **dns** | *string* | Yes | DNS. | |
| **ip** | *string* | Yes | IP. | |
| **proxy_hostid** | *int* | Yes | Proxy Host ID. | |
| **useipmi** | *int* | Yes | Use IPMI. | |
| **ipmi_ip** | *string* | Yes | IPMI IP. | |
| **ipmi_port** | *int* | Yes | IPMI port. | |
| **ipmi_authtype** | *int* | Yes | IPMI authentication type. | |
| **ipmi_privilege** | *int* | Yes | IPMI privilege. | |
| **ipmi_username** | *string* | Yes | IPMI username. | |
| **ipmi_password** | *string* | Yes | IPMI password. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated host IDs. |
| **error** | In case of any errors |

**Example** Enable two hosts and switch to monitoring by IP addresses:

```
{
"jsonrpc":"2.0",
"method":"host.massUpdate",
"params":{
    "hosts":[
        {
            "hostid": "69665"
        },
        {
            "hostid": "69666"
        }
      ],
      "status":0,
      "useip":1
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

Hosts updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "hostids":["69665","69666"]
},
"id":2
}
```

**update()**

Available since version: **1.8**
The method is used to control all host attributes including host template linkage, macros and host group membership. The method is a wrapper for **host.massUpdate** function.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **hostid** | *string* | | Host ID. | |
| **host** | *string* | Yes | Host name. | |
| **status** | *int* | Yes | Host Status. | |
| **dns** | *string* | Yes | Host DNS. | |
| **ip** | *string* | Yes | Host IP. | |
| **useip** | *int* | Yes | Use IP. | |
| **port** | *int* | Yes | Host Port. | |
| **proxy_hostid** | *int* | Yes | Proxy id. | |
| **useipmi** | *int* | Yes | Use IPMI. | |
| **ipmi_ip** | *string* | Yes | IPMI IP. | |
| **ipmi_port** | *int* | Yes | IPMI port. | |
| **ipmi_authtype** | *int* | Yes | IPMI authentication type. | |
| **ipmi_privilege** | *int* | Yes | IPMI privilege. | |
| **ipmi_username** | *string* | Yes | IPMI username. | |
| **ipmi_password** | *string* | Yes | IPMI password. | |
| **groups** | *array* | Yes | Host groups. | |
| **templates** | *array* | Yes | Update host templates linkage. Missing templates will be linked, existed stay, others unlinked | |
| **templates_clear** | *array* | Yes | Templates to unlink and clear. | |
| **macros** | *array* | Yes | Update host macros. Missing macros will be added, existed updated, others deleted. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Host IDs. |
| **error** | In case of any errors |

**Example 1** Enable host, .i.e set its status to '0':

```
{
 "jsonrpc": "2.0",
 "method": "host.update",
 "params": {
    "hostid": "10092",
    "status": 0
 },
 "auth": "700ca65537074ec963db7efabda78259",
 "id": 2
}
```

Retrieved updated host IDs:

```
{
 "jsonrpc": "2.0",
 "result": {
     "hostids": ["10092"]
 },
 "id": 2
}
```

**Example 2** Unlink and clear templates on host:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
```

```
    "params": {
        "hostid": "10126",
        "templates_clear": [{
            "templateid": "10124"
         },{
            "templateid": "10125"
         }]
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 2
}
```

Retrieved updated host IDs:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": ["10126"]
    },
    "id": 2
}
```

**Example 3** Update macros on host:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10126",
        "macros":[{
            "macro":"{$PASS}",
            "value":"password"
        },{
            "macro":"{$DISC}",
            "value":"sda"
        }]
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 2
}
```

Retrieved updated host IDs:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": ["10126"]
    },
    "id": 2
}
```

## Hostgroup

**Methods** Class containing methods for operations with Hosts.

| Methods | Description |
|---|---|
| get() | Get host group details |
| exists() | Check if host group exists |
| create() | Create host groups |
| update() | Update host group details |
| delete() | Delete host groups |

| Methods | Description |
|---|---|
| massAdd() | Mass add templates, hosts to host groups |
| massUpdate() | Mass update host group details, update list of templates, hosts |
| massRemove() | Mass remove templates, hosts |

**Object details** The table contains complete list of Host attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| groupid | *int* | Host ID | |
| name | *string* | Host name. | |
| internal | *integer* | HostGroup status, if equal to 1 - host group can't be deleted. | |

**Common tasks** The table contains list of common host-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add a host group | Use method **hostgroup.create** |
| Add a bunch of new host groups | Use method **hostgroup.create** with array of Host group objects |
| Retrieve host group details by Group IDs | Use method **hostgroup.get** with parameter **groupids** |
| Retrieve host group details by Host group name | Use method **hostgroup.get** with parameter **filter**, specify **"name":"<your hostgroup>"** |

**create()**

This function allows you to create a hostgroup as defined by the **hostgroup data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **hostgroup data** | *array or object* | | Array of Host group objects or a single object | **groupid** shouldn't be specified |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Host group IDs. **groupid** are assigned to each Host group object |
| **error** | In case of any errors |

**Example**

```
{
"jsonrpc":"2.0",
"method":"hostgroup.create",
"params":[
   {"name":"Linux Group"}
],
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":3
}
```

Host group created successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "groupids": ["107819"]
},
"id":3
}
```

Host group already exists:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CHostGroup::create ] HostGroup [ Linux Group ] already exists"
},
"id":3
}
```

**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several host groups.

Host group can't be deleted:

1. contained hosts are linked to single(deleted) host group, hosts must be linked to at least one host group;
2. host group is used to link discovered hosts to it (Administration→General→Other);

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **groups** | *array* | | Array of Host Group objects | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Host Group IDs. |
| **error** | In case of any errors |

**Example**

```
{
"jsonrpc":"2.0",
"method":"hostgroup.delete",
"params":[
      {
         "groupid":107824
      },
      {
         "groupid":107825
      }
],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Host Groups deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "groupids": ["107824", "107825"]
},
"id":2
}
```

Host Group does not exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CHostgroup::delete ] Host group does not exist"
},
"id":2
}
```

**exists()**

Available since version: **1.8.3**
This function allows you to check whether host group with given host group name or host group ID exists.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **nodeids** | *array* | yes | List of node IDs where to search for given host group ID or name | |
| **groupid** | *string* | yes | Host group ID | |
| **name** | *string* | yes | Host group name | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain boolean variable. |
| **error** | In case of any errors |

**Example**

```
{
"jsonrpc":"2.0",
"method":"hostgroup.exists",
"params":{
    "nodeids": ["1"],
    "name": "Linux Group"
},
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Host group exists:

```
{
"jsonrpc":"2.0",
"result": true,
"id":2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve host group details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **groupids** | *array* | HostGroup IDs | |
| **hostids** | *array* | Host IDs | |
| **templateids** | *array* | Template IDs | |
| **triggerids** | *array* | Trigger IDs | |
| **graphids** | *array* | Graph IDs | |
| **proxyids** | *array* | Return only host groups with hosts, that are monitored by the given proxies. | |
| **maintenanceids** | *array* | Maintenance IDs | |
| **monitored_hosts** | *integer* | return only host groups containing monitored hosts | |
| **templated_hosts** | *integer* | return only host groups containing templates | |
| **real_hosts** | *integer* | return only host groups containing hosts (monitored/not monitored) in result | |
| **not_proxy_hosts** | *integer* | return only host groups not containing Proxies | |
| **with_items** | *integer* | only with items | |
| **with_monitored_items** | *integer* | only with monitored items | |
| **with_historical_items** | *integer* | only with historical items | |
| **with_triggers** | *integer* | only with triggers | |
| **with_monitored_triggers** | *integer* | only with monitored triggers | |
| **with_httptests** | *integer* | only with http tests | |
| **with_monitored_httptests** | *integer* | only with monitored http tests | |
| **with_graphs** | *integer* | only with graphs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by host group fields | |
| **search** | *array* | Return host groups by any given object field pattern | |
| **startSearch** | *integer* | Search host groups field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, host groups by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_templates** | *string* | Select contained templates | Values: shorten, refer, extend |
| **select_hosts** | *string* | Select contained hosts | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count host groups, return the number of host groups found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by host group field | Values: groupid, name |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **limit** | *int* | Max number of host group objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Host group objects. |
| **error** | In case of any errors |

**Example** Get host groups details by host group name ”Zabbix servers”,”Linux servers”:

```
{
"jsonrpc":"2.0",
"method":"hostgroup.get",
"params":{
    "output": "extend",
    "filter":{
        "name":["Zabbix servers","Linux servers"]
        }
    },
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved host group details:

```
{
"jsonrpc":"2.0",
"result":[
    {
        "groupid":"100100000000002",
        "name":"Linux servers",
        "internal":"0"
    },
    {
        "groupid":"100100000000004",
        "name":"ZABBIX Servers",
        "internal":"0"
    }
],
"id":2
}
```

**massAdd()**

Available since version: **1.8**

**Parameters** Multidimensional array with Host groups data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **groups** | *array* | | Host group to update. | |
| **hosts** | *array* | Yes | Host objects that should be added to host groups. | |
| **templates** | *array* | Yes | Template objects that should be added to host groups. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Host group IDs. |
| **error** | In case of any errors |

**Example** Add two hosts with **ID** ”100100000010092”, ”100100000010086” to two host groups with **ID** ”100100000000042”, ”100100000000013”

```
{
"jsonrpc":"2.0",
"method":"hostgroup.massAdd",
"params":{
    "groups": [
      {"groupid": "100100000000042"},
      {"groupid": "100100000000013"}
    ],
    "hosts": [
        {"hostid": "100100000010092"},
        {"hostid": "100100000010086"}
    ]
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

Host groups updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "groupids":["100100000000042","100100000000013"]
},
"id":2
}
```

**massRemove()**

Available since version: **1.8**

**Parameters** multidimensional array with Hosts data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **groupids** | *array* | | Host groupids to update. | |
| **hostids** | *array* | Yes | Hostids to remove from host groups. | |
| **templateids** | *array* | Yes | Templateids to remove from host groups. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Host group IDs. |
| **error** | In case of any errors |

**Example** Remove two hosts with **ID** ”100100000010092”, ”100100000010086” from two host groups with **ID** ”100100000000042”, ”100100000000013”

```
{
"jsonrpc":"2.0",
"method":"hostgroup.massRemove",
"params":{
    "groupids": ["100100000000042","100100000000013"],
    "hostids": ["100100000010092","100100000010086"]
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

Host groups updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "groupids":["100100000000042","100100000000013"]
},
"id":2
}
```

**massUpdate()**

Available since version: **1.8**
==== Parameters ==== multidimensional array with Hosts data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **groups** | *array* | | HostGroup objects to update. | |
| **hosts** | *array* | Yes | Host objects that should be added to host groups, others will be removed. | |
| **templates** | *array* | Yes | Template objects that should be added to host groups, others will be removed. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Host group IDs. |
| **error** | In case of any errors |

**Example** Add host with **ID** ”100100000010092” to host group with **ID** ”100100000000042” others hosts in host group will be removed:

```
{
"jsonrpc":"2.0",
"method":"hostgroup.massUpdate",
"params":{
    "groups": [
      {"groupid": "100100000000042"},
    ],
    "hosts": [
        {"hostid": "100100000010092"},
    ]
},
```

```
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

Host groups updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "groupids":["100100000000042"]
},
"id":2
}
```

**update()**

Available since version: **1.8**
The method is used to control host group attributes.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **groupid** | *string* | | Host name. | |
| **name** | *any* | Yes | New value for a hostgroup name. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Host group IDs. |
| **error** | In case of any errors |

**Example** Rename host group:

```
{
 "jsonrpc":"2.0",
 "method":"hostgroup.update",
 "params":[
     {"groupid": "100100000000042", "name": "Rename 1"},
     {"groupid": "100100000000013", "name": "Rename 2"}
 ],
 "auth":"700ca65537074ec963db7efabda78259",
 "id":2
}
```

Retrieved updated host IDs:

```
{
  "jsonrpc":"2.0",
  "result": {
  "groupids":["100100000000042","100100000000013"]
  },
  "id":2
}
```

## Image

**Methods** Class containing methods for operations with Images.

| Methods | Description |
|---|---|
| get() | Get image details |
| exists() | Check if image exists |
| create() | Create images |
| update() | Update image details |
| delete() | Delete images |

**Object details** The table contains complete list of Image attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| imageid | *int* | Image ID | |
| imagetype | *int* | Type | |
| name | *string* | Image description | |
| image | *string* | Image binary data | Sent as base64 encoded string |

**Common tasks** The table contains list of common image-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add an image | Use method **image.create** |
| Add a bunch of new images | Use method **image.create** with array of Image objects |
| Retrieve image details by Image IDs | Use method **image.get** with parameter **imageids** |
| Retrieve images details by Image name | Use method **image.get** with parameter **filter**, specify **"name": ["<your image>"]** |

**create()**

This function allows you to create a image as defined by the **image data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **image data** | *array or object* | | Array of Image objects or a single object | **imageid** shouldn't be specified **image** must be sent as base64 encoded binary data |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Image IDs. **imageid** are assigned to each Image object |
| **error** | In case of any errors |

**Example** Create a new image:

```
{
"jsonrpc": "2.0",
"method": "image.create",
"params": [{
    "imagetype": "1",
```

```
    "name": "ZABBIX Hub",
    "image": "iVBORw0KGgoAAAANSUhEUgAAADAAAAAwCAYAAABXAvmHAAAABmJLR0QA\/wD\/AP+gvaeTAAAAB3RJTUUH0QEfCSscTQ
}],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 2
}
```


Image created successfully:


```
{
"jsonrpc": "2.0",
"result": {
    "imageids": ["100100000000047"]
},
"id": 2
}
```


Image already exists:


```
{
"jsonrpc": "2.0",
"error": {
      "code": -32602,
      "message": "Invalid params.",
      "data": "[ CImage::create ] Cannot create Image"
},
"id":2
}
```


**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several images. Image can't be removed if is used in maps as icon or background. Available only to super admins.

**Parameters** Array of Image IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Image IDs. |
| **error** | In case of any errors |

**Example** Delete images by image **ID**


```
{
"jsonrpc":"2.0",
"method":"image.delete",
"params":["107824",  "107825"],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```


Images deleted successfully:


```
{
"jsonrpc":"2.0",
"result":{
   "imageids": ["107824", "107825"]
},
"id":2
```

```
}
```

Images does not exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CImage::delete ] Image does not exist"
},
"id":2
}
```

**exists()**

Available since version: **1.8.3**
This function allows you to check whether image with given image data exists.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **nodeids** | *array* | yes | List of node IDs where to search for given image | |
| **imageid** | *string* | yes | Image ID | |
| **name** | *string* | yes | Name | |
| **imagetype** | *string* | No | Type | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain boolean variable. |
| **error** | In case of any errors |

**Example** Check if image icon with name Hub exists:

```
{
"jsonrpc":"2.0",
"method":"image.exists",
"params":{
    "name": "Hub",
    "imagetype": 1
},
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Image exists:

```
{
"jsonrpc":"2.0",
"result": true,
"id":2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve image details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **imageids** | *array* | Image IDs | |
| **sysmapids** | *array* | Map IDs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by image fields | |
| **search** | *array* | Return images by any given image object field pattern | |
| **startSearch** | *integer* | Search images field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, images by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_image** | *string* | Select image source | |
| **countOutput** | *integer* | Count images, return the number of images found | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by image field | Values: imageid, name |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of image objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Image objects. |
| **error** | In case of any errors |

**Example** Get images details by image description ”Hub” and ”UPS”, return only **image** details, without image source:

```
{
"jsonrpc":"2.0",
"method":"image.get",
"params":{
    "filter": {"name": ["Hub", "UPS"]},
    "output": "extend"
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved images details:

```
{
"jsonrpc":"2.0",
"result":[
    {
        "imageid":"100100000000001",
        "imagetype":"1",
        "name":"Hub"
    },
    {
        "imageid":"100100000000017",
        "imagetype":"1",
        "name":"UPS"
    }
```

```
],
"id":2
}
```

#### update()

Available since version: **1.8**
The method is used to control all image attributes including image applications linkage.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **imageid** | *string* | | Image ID. | |
| **image attribute** | *any* | Yes | New value for a image attribute. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Image IDs. |
| **error** | In case of any errors |

**Example** Rename image and change type to background:

```
{
"jsonrpc":"2.0",
"method":"image.update",
"params":[{
    "imageid":"100100000000047",
    "imagetype":"2",
    "name":"Hub background"
}],
"auth":"700ca65537074ec963db7efabda78259",
"id":2
}
```

Retrieved updated image IDs:

```
  {
   "jsonrpc":"2.0",
   "result": {
       "imageids":["100100000010092"]
   },
   "id":2
  }
```

## Item

**Methods** Class containing methods for operations with Items.

| Methods | Description |
|---|---|
| get() | Get item details |
| exists() | Check if item exists |
| create() | Create items |
| update() | Update item details |

| Methods | Description |
|---|---|
| delete() | Delete items |

**Object details** The table contains complete list of Item attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| itemid | *int* | Item ID | |
| type | *int* | Type | |
| snmp_community | *string* | SNMP Community name | |
| snmp_oid | *string* | SNMP OID | |
| snmp_port | *int* | SNMP port | |
| hostid | *int* | Host ID | |
| description | *string* | Item description | |
| 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) | |
| lastvalue | *string* | Last value | |
| lastclock | *int* | Last check | |
| prevvalue | *string* | Previous value | |
| 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 | |
| prevorgvalue | *string* | | |
| 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* | | |
| error | *string* | Item check error | |
| lastlogsize | *int* | Last log size | |
| logtimefmt | *string* | Log time format | |
| templateid | *int* | Parent item ID | |
| valuemapid | *int* | Value map ID | |
| 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* | | |
| mtime | *int* | Micro time | |

**Field values**

**Type**

| Value | Type |
|---|---|
| 0 | Zabbix agent |
| 1 | SNMPv1 |
| 2 | Trapper |
| 3 | Simple check |
| 4 | SNMPv2 |
| 5 | Internal |

| Value | Type |
|---|---|
| 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 |

**Status**

| Value | Type |
|---|---|
| 0 | active |
| 1 | disabled |
| 3 | not supported |

**Value type**

| Value | Type |
|---|---|
| 0 | Numeric (float) |
| 1 | Character |
| 2 | Log |
| 3 | Numeric (unsigned) |
| 4 | Text |

**Data type**

| Value | Type |
|---|---|
| 0 | Decimal |
| 1 | Octal |
| 2 | Hexadecimal |

**Delta**

| Value | Status |
|---|---|
| 0 | As is |
| 1 | Delta (speed per second) |
| 2 | Delta (simple change) |

**Common tasks** The table contains list of common item-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add an item | Use method **item.create** |
| Add a bunch of new items | Use method **item.create** with array of Item objects |
| Enable an item | Use method **item.update**, set **”status”:0** |
| Disable an item | Use method **item.update**, set **”status”:1** |
| Retrieve item details by Item IDs | Use method **item.get** with parameter **itemids** |
| Retrieve items details by Host name | Use method **item.get** with parameter **filter**, specify **”host”: [”<your host1>”]** |

### create()

This function allows you to create a item as defined by the **item data** array.

#### Parameters

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **item data** | *array or object* | | Array of item objects or a single object | **itemid** shouldn't be specified |

#### Returns

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created item IDs. **itemid** are assigned to each item object |
| **error** | In case of any errors |

**Example** Create new item for host with host **ID** "100100000010048"

```
{
"jsonrpc":"2.0",
"method":"item.create",
"params":{
      "description": "Free disk space on $1",
      "key_": "vfs.fs.size[/home/aly/,free]",
      "hostid": "100100000010048",
      "applications": ["100100000000001", "100100000000002"]
},
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":2
}
```

Item created successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "itemids": ["100100000214797"]
},
"id":2
}
```

Item already exists:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CItem::create ] Cannot create Item"
},
"id":2
}
```

### delete()

Available since version: **1.8**
This function allows you to delete information about one or several items. All item-related information will be removed including triggers, empty graphs, child items, historical data.

**Parameters** Array of Item IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Item IDs. |
| **error** | In case of any errors |

**Example** Delete items by item **ID**

```
{
"jsonrpc":"2.0",
"method":"item.delete",
"params":["107824",  "107825"],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Items deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "itemids": ["107824", "107825"]
},
"id":2
}
```

Items does not exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CItem::delete ] Item does not exist"
},
"id":2
}
```

**exists()**

Available since version: **1.8.3**
This function allows you to check whether item with given item data exists.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **nodeids** | *array* | yes | List of node IDs where to search for given item | |
| **key_** | *string* | No | Item key | |
| **hostid** | *string* | yes | Host ID | |
| **host** | *string* | yes | Host name | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain boolean variable. |
| **error** | In case of any errors |

**Example** Check if item with key ”vfs.file.cksum[c:\config.sys]” exists for host ”Windows-Server”

```
{
"jsonrpc":"2.0",
"method":"item.exists",
"params":{
    "host": "Windows-Server",
    "key_": "vfs.file.cksum[c:\config.sys]"
},
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Item exists:

```
{
"jsonrpc":"2.0",
"result": true,
"id":2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve item details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **groupids** | *array* | HostGroup IDs | |
| **hostids** | *array* | Host IDs | |
| **templateids** | *array* | Template IDs | |
| **proxyids** | *array* | Return only items that belong to the hosts, that are monitored by the given proxies. | |
| **itemids** | *array* | Item IDs | |
| **graphids** | *array* | Graph IDs | |
| **triggerids** | *array* | Trigger IDs | |
| **applicationids** | *array* | Application IDs | |
| **webitems** | *integer* | Search **also** in web items | |
| **inherited** | *integer* | Inherited from templates | ”0” - not inherited, ”1” - inherited |
| **templated** | *integer* | Templated items | ”0” - belongs to hosts, ”1” - belongs to templates |
| **monitored** | *integer* | Monitored items | Checks item and host status |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **group** | *string* | Optional filter by host group name | |
| **host** | *string* | Optional filter by host name | |
| **application** | *string* | Optional filter by application name | |
| **belongs** | *string* | Optional filter by host fields | |
| **with_triggers** | *integer* | Items with triggers | |
| **filter** | *array* | Optional filter by item fields | |
| **search** | *string* | Return items by given item fields pattern | |
| **startSearch** | *integer* | Search given patterns only in start of the field | |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **excludeSearch** | *integer* | Exclude from result items by given patterns | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_hosts** | *string* | Select hosts | Values: shorten, refer, extend |
| **select_triggers** | *string* | Select item triggers | Values: shorten, refer, extend |
| **select_graphs** | *string* | Select item graphs | Values: shorten, refer, extend |
| **select_applications** | *string* | Select item applications | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count hosts, return the number of items found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by item field | Values: itemid, description, key_, delay, history, trends, type, status |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of item objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Item objects. |
| **error** | In case of any errors |

**Example** Get items details by item description pattern ”Apache” and limit output to 10 items, return only **item** IDS:

```
{
"jsonrpc":"2.0",
"method":"item.get",
"params":{
    "output":"shorten",
    "search": {"description": "apache"},
    "limit": 10
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved items details:

```
{
"jsonrpc":"2.0",
"result":[
    {"itemid":"100100000010048"},
    {"itemid":"100100000010137"},
    {"itemid":"100100000017431"},
    {"itemid":"100100000017533"},
    {"itemid":"100100000017635"},
    {"itemid":"100100000017737"},
    {"itemid":"100100000017839"},
    {"itemid":"100100000017941"},
    {"itemid":"100100000018043"},
```

```
    {"itemid":"100100000018145"}
],
"id":2
}
```

#### update()

Available since version: **1.8**
The method is used to control all item attributes including item applications linkage.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **itemid** | *string* | | Item ID. | |
| **item attribute** | *any* | Yes | New value for a item attribute. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Item IDs. |
| **error** | In case of any errors |

**Example** Enable item, .i.e set its status to '0':

```
{
 "jsonrpc":"2.0",
 "method":"item.update",
 "params":{
    "itemid": "100100000010092",
    "status": 0
 },
 "auth":"700ca65537074ec963db7efabda78259",
 "id":2
}
```

Retrieved updated item IDs:

```
{
 "jsonrpc":"2.0",
 "result": {
     "itemids":["100100000010092"]
 },
 "id":2
}
```

## Maintenance

**Methods** Class containing methods for operations with Maintenances.

| Methods | Description |
|---|---|
| get() | Get maintenance details |
| exists() | Check if maintenance exists |
| create() | Create maintenances |
| update() | Update maintenance details |
| delete() | Delete maintenances |

**Object details** The table contains complete list of Maintenance attributes.

**Maintenance**

| Parameter | Type | Description | Details |
|---|---|---|---|
| maintenanceid | *integer* | Maintenance ID | |
| name | *string* | Name. | |
| maintenance_type | *integer* | Type. | 0: With data collection<br>1: No data collection |
| description | *string* | Description. | |
| active_since | *integer* | Activation date. | Unix timestamp |
| active_till | *integer* | Deactivation date. | Unix timestamp |
| timeperiods | *array of timeperiod objects* | Timeperiods | |

**Timeperiod**

| Parameter | Type | Description | Details |
|---|---|---|---|
| timeperiod_type | *integer* | Type defines what fields are used and how values of those fields are processed.<br>Required fields by timeperiod type:<br>0 - start_date, period;<br>2 - start_time, period, every;<br>3 - start_time, period, every, dayofweek;<br>4 - start_time, period, every, dayofweek, month, day; | 0: Onetime<br>2: Daily<br>3: Weekly<br>4: Monthly |
| every | *integer* | Depends on type:<br>2 - every Nth day (if every=2 timeperiod is triggered every second day);<br>3 - every Nth week<br>4 - is used when field **day** is 0 and then means every Nth week of month (1 - 5) | |
| month | *integer* | Number, got by converting binary number, where each bit represents one month (Dec is first bit, Jan is last bit) to decimal number.<br>For example if you need maintenance on March and April, binary representation is '000000001100' and decimal is 12. | |
| dayofweek | *integer* | Used for type 3 and for type 4 when **day** is 0.<br>Number with week days calculated in same way as **month**. (Sun is first bit, Mon is last bit) | |
| day | *integer* | If equal to 0 then field **every** and **dayofweek** are used, otherwise represents number of day on which timeperiod is triggered. | |
| start_time | *integer* | Period start time in seconds | |
| period | *integer* | Period length in seconds | |
| start_date | *integer* | Period start date as Unix timestamp | |

**Common tasks** The table contains list of common maintenance-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add a maintenance | Use method **maintenance.create** |
| Add a bunch of new maintenances | Use method **maintenance.create** with array of Maintenance objects |
| Rename a maintenance | Use method **maintenance.update**, set **"name":"<new name>"** |
| Retrieve maintenance details by Maintenance IDs | Use method **maintenance.get** with parameter **maintenanceids** |
| Retrieve maintenance details by Maintenance name | Use method **maintenance.get** with parameter **filter**, specify **"name":"<your maintenance>"** |

**create()**

This method allows you to create a maintenance as defined by the **maintenance data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **maintenance data** | *array or object* | No | Array of Maintenance objects or a single object | **maintenanceid** shouldn't be specified |
| **groupids** | *array* | No | Host group ids | |
| **hostids** | *array* | No | Host ids | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Maintenance IDs. **maintenanceid** are assigned to each Maintenance object |
| **error** | In case of any errors |

**Examples** simple create

```
{
"jsonrpc":"2.0",
"method":"maintenance.create",
"params":[{
    "groupids":[],
    "hostids":["100100000010229"],
    "name":"ZABBIX Servers",
    "maintenance_type":"0",
    "description":"",
    "active_since":"1276163035",
    "active_till":"1307698980"
}],
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":3
}
```

Maintenance added successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "maintenanceids": ["100100000000005"]
},
"id":3
}
```

Maintenance already exists:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CMaintenance::create ] Maintenance [ ZABBIX Servers ] already exists"
},
"id":3
}
```

create maintenance with onetime period

```
{
    "groupids": [
        14
    ],
    "name": "T1",
    "maintenance_type": 0,
    "description": "",
    "active_since": "1276163035",
    "active_till": "1307698980",
    "timeperiods": [
        {
            "timeperiod_type": 0,
            "start_date": "1307689239",
            "period": 7200
        }
    ]
}
```

create maintenance with daily period (at 11:00 every five days for 6 hours 2 minutes)

```
{
    "groupids": [
        14
    ],
    "name": "T2",
    "maintenance_type": 0,
    "description": "",
    "active_since": "1276163035",
    "active_till": "1307698980",
    "timeperiods": [
        {
            "timeperiod_type": 2,
            "start_time": 39600,
            "period": 21720,
            "every": 5
        }
    ]
}
```

create maintenance with weekly period (at 11:00 on Monday and Tuesday of every second week for 6 hours 2 minutes)

```
{
    "groupids": [
        14
    ],
    "name": "T3",
    "maintenance_type": 0,
    "description": "",
    "active_since": "1276163035",
    "active_till": "1307698980",
    "timeperiods": [
        {
```

```
            "timeperiod_type": 3,
            "start_time": 39600,
            "period": 21720,
            "every": 2,
            "dayofweek": 3
        }
    ]
}
```

create maintenance with monthly period (at 10:00 on every second week Monday and Wednesday of every January and March for 2 hours)

```
{
    "groupids": [
        14
    ],
    "name": "T4",
    "maintenance_type": 0,
    "description": "",
    "active_since": "1276163035",
    "active_till": "1307698980",
    "timeperiods": [
        {
            "timeperiod_type": 4,
            "start_time": 36000,
            "period": 7200,
            "every": 2,
            "dayofweek": 5,
            "month": 5,
            "day": 0
        }
    ]
}
```

**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several maintenances. All maintenance-related information will be removed including items, graphs, macros, application, historical data, etc.

**Parameters** Array of Maintenance IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Maintenance IDs. |
| **error** | In case of any errors |

**Example** Delete maintenances by Maintenance **ID**

```
{
"jsonrpc":"2.0",
"method":"maintenance.delete",
"params":["107824", "107825"],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Maintenances deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "maintenanceids": ["107824", "107825"]
},
"id":2
}
```

Maintenance does not exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CMaintenance::delete ] Maintenance does not exist"
},
"id":2
}
```

**exists()**

Available since version: **1.8.3**
This function allows you to check whether maintenance with given maintenance name or maintenance ID exists.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **nodeids** | *array* | yes | List of node IDs where to search for given maintenance ID or maintenance name | |
| **maintenanceid** | *string* | yes | Maintenance ID | |
| **maintenance** | *string* | yes | Maintenance name | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain boolean variable. |
| **error** | In case of any errors |

**Example**

```
{
"jsonrpc":"2.0",
"method":"maintenance.exists",
"params":{
    "nodeids": ["1"],
    "maintenance": "ZABBIX Servers"
},
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Maintenance exists:

```
{
"jsonrpc":"2.0",
"result": true,
"id":2
}
```

**get()**

Available since version: **1.8**
This method allows you to retrieve maintenance details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **groupids** | *array* | HostGroup IDs | |
| **hostids** | *array* | Host IDs | |
| **maintenanceids** | *array* | Maintenance IDs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by maintenance fields | |
| **search** | *array* | Return maintenances by any given maintenance object field pattern | |
| **startSearch** | *integer* | Search maintenances field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, maintenances by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_groups** | *string* | Select host groups | Values: shorten, refer, extend |
| **select_hosts** | *string* | Select hosts | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count maintenances, return the number of maintenances found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by maintenance field | Values: maintenanceid, name |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of maintenance objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Maintenance objects. |
| **error** | In case of any errors |

**Example** Get maintenances details by maintenance name pattern ”server” and limit output to two maintenances:

```
{
"jsonrpc":"2.0",
"method":"maintenance.get",
"params":{
    "search": {
        "name":"server"
    }
    "output": "extend",
```

```
    "select_hosts": "refer",
    "select_groups": "refer",
    "limit": 2
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved maintenance details:

```
{
"jsonrpc":"2.0",
"result":[{
    "groups":[],
    "hosts":[{
        "hostid":"100100000010229",
        "maintenanceid":"100100000000005"
    }],
    "maintenanceid":"100100000000005",
    "name":"Zabbix server maintenance",
    "maintenance_type":"0",
    "description":"",
    "active_since":"1276163035",
    "active_till":"1307698980"
}],
"id":2
}
```

,

**update()**

This method allows you to update a maintenance as defined by the **maintenance data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **maintenance data** | *array or object* | No | Array of Maintenance objects or a single object | **maintenanceid** must be specified |
| **groupids** | *array* | No | Host group ids add/remove to/from maintenance | |
| **hostids** | *array* | No | Host ids add/remove to/from maintenance | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Maintenance IDs. |
| **error** | In case of any errors |

**Example** Update maintenance name, remove all hosts and update groups:

```
{
"jsonrpc":"2.0",
"method":"maintenance.update",
"params":[{
    "maintenanceid":"100100000000005",
    "name":"TEST",
```

```
    "groupids":["100100000010229"],
    "hostids":[]
}],
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":3
}
```

Maintenance updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "maintenanceids": ["100100000000005"]
},
"id":3
}
```

Maintenance already exists:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CMaintenance::update ] Maintenance [ ZABBIX Servers ] already exists"
},
"id":3
}
```

## Map

**Methods** Class containing methods for operations with Maps.

| Methods | Description |
|---|---|
| get() | Get map details |
| exists() | Check if map exists |
| create() | Create maps |
| update() | Update map details |
| delete() | Delete maps |

**Object details** The table contains complete list of Map attributes.

**Map**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **sysmapid** | *integer* | Map ID | |
| **name** | *string* | Name | |
| **width** | *integer* | Width | |
| **height** | *integer* | Height | |
| **backgroundid** | *integer* | Background image ID | |
| **label_type** | *integer* | Icon label type | Label, Element Name, IP, Status only |
| **label_location** | *integer* | Icon label location | Top, Bottom, Right, Left |
| **highlight** | *integer* | Icon highlight | |
| **expandproblem** | *integer* | Expanding single problem | |
| **markelements** | *integer* | Extended icon highlighting in case of status changes | |
| **show_unack** | *integer* | Unacknowledged problem viewing | All problems, Separate, Only unacknowledged |

**Map item**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **selementid** | *integer* | Map element ID | |
| **sysmapid** | *integer* | Map ID | |
| **elementid** | *integer* | Resource ID | |
| **elementtype** | *integer* | Resource type | 0 - host, 1 - map, 2 - trigger, 3 - host group, 4 - image |
| **iconid_off** | *integer* | OK status icon ID | |
| **iconid_on** | *integer* | PROBLEM status icon ID | |
| **iconid_unknown** | *integer* | UNKNOWN status icon ID | |
| **iconid_disabled** | *integer* | Disabled status icon ID | |
| **iconid_maintenance** | *integer* | Maintenance status icon ID | |
| **label** | *integer* | Description | |
| **label_location** | *integer* | Description location | |
| **x** | *integer* | X axis position | |
| **y** | *integer* | Y axis position | |
| **url** | *integer* | Page to open on element click | |

**Map Item Links**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **linkid** | *integer* | Map link ID | |
| **sysmapid** | *integer* | Map ID | |
| **selementid1** | *integer* | First linked map element ID | |
| **selementid2** | *integer* | Second linked map element ID | |
| **drawtype** | *integer* | Link draw type | Line, Bold line, Dot, Dashed line |
| **color** | *string* | Link default color | Hex presentation |
| **label** | *string* | Link description | |

**Map Item Link Status Indicator**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **linktriggerid** | *integer* | Map link Indicator ID | |
| **linkid** | *integer* | Map link ID | |
| **triggerid** | *integer* | Trigger ID | |
| **drawtype** | *integer* | Draw type | Line, Bold line, Dot, Dashed line |
| **color** | *string* | Color | Hex presentation |

**Common tasks** The table contains list of common map-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add a map | Use method **map.create** |
| Add a bunch of new maps | Use method **map.create** with array of Map objects |
| Remove map by Map IDs | Use method **map.delete** array of Map **IDs** |
| Retrieve map details by Map IDs | Use method **map.get** with parameter **sysmapids** |
| Retrieve map details by Map name | Use method **map.get** with parameter **filter**, specify **”name”:”<your map>”** |

**create()**

This function allows you to create a map as defined by the **map data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **map data** | *array or object* | | Array of Map objects or a single object with additional paramter **selements**, array of map item objects | **sysmapid** shouldn't be specified |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Map IDs. **sysmapid** is assigned to each Map object |
| **error** | In case of any errors |

**Example** Create new map

```
{
"jsonrpc": "2.0",
"method": "map.create",
"params": [{
    "selements": [{
        "elementid": "0",
        "elementtype": "4",
        "iconid_off": "100100000000036",
        "iconid_on": "0",
        "iconid_unknown": "0",
        "label": "New element",
        "label_location": "0",
        "x": "200",
        "y": "100",
        "url": "",
        "iconid_disabled": "0",
        "iconid_maintenance": "0"
    }],
    "name": "ZABBIX-Map",
    "width": "800",
    "height": "600",
    "backgroundid": "0",
    "label_type": "0",
    "label_location": "0",
    "highlight": 0,
    "expandproblem": 0,
    "markelements": 0,
    "show_unack": 0
}],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 2
}
```

Map created successfully:

```
{
"jsonrpc": "2.0",
"result": {
    "sysmapids": ["100100000012213"]
},
"id": 2
}
```

Map already exists:

```
{
"jsonrpc": "2.0",
"error": {
      "code": -32602,
      "message": "Invalid params.",
      "data": "[ CMap::create ] Map [ ZABBIX-Map ] already exists"
},
"id": 2
}
```


**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several maps. Map items and links will be removed.

**Parameters** Array of Map IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Map IDs. |
| **error** | In case of any errors |

**Example** Delete maps by map **ID**


```
{
"jsonrpc":"2.0",
"method":"map.delete",
"params":["107824",  "107825"],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```


Maps deleted successfully:


```
{
"jsonrpc":"2.0",
"result":{
   "sysmapids": ["107824", "107825"]
},
"id":2
}
```


Maps does not exist:


```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CMap::delete ] Map does not exist"
},
"id":2
}
```


**exists()**

Available since version: **1.8.3**
This function allows you to check whether map with given map data exists.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **nodeids** | *array* | yes | List of node IDs where to search for given map | |
| **sysmapid** | *string* | Yes | Map ID | |
| **name** | *string* | Yes | Name | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain boolean variable. |
| **error** | In case of any errors |

**Example** Check if map with name ”ZABBIX” exists

```
{
"jsonrpc":"2.0",
"method":"map.exists",
"params":{
    "name": "ZABBIX"
},
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Map exists:

```
{
"jsonrpc":"2.0",
"result": true,
"id":2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve Map details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **sysmapids** | *array* | Map IDs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by map fields | |
| **search** | *array* | Return maps by any given object field pattern | |
| **startSearch** | *integer* | Search maps field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, maps by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_selements** | *string* | Select map items | Values: shorten, refer, extend |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **select_links** | *string* | Select map item links | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count maps, return the number of maps found | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by Map field | Values: name |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *integer* | max number of map objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Map objects. |
| **error** | In case of any errors |

**Example** Get maps details by Map name ”zabbix”, with map elements and links:

```
{
"jsonrpc":"2.0",
"method":"Map.get",
"params":{
    "filter": {"name": ["zabbix"]},
    "select_selements": "extend",
    "select_links": "extend",
    "output": "extend"
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved Map details:

```
{
"jsonrpc":"2.0",
"result":[{
    "sysmapid":"100100000000031",
    "name":"Link Map",
    "width":"800",
    "height":"600",
    "backgroundid":"0",
    "label_type":"2",
    "label_location":"0",
    "highlight":1,
    "expandproblem":1,
    "markelements":0,
    "show_unack":0,
    "selements":[{
        "selementid":"100100000000414",
        "sysmapid":"100100000000031",
        "elementid":"100100000037426",
        "elementtype":"2",
        "iconid_off":"100100000000001",
        "iconid_on":"0",
        "iconid_unknown":"0",
        "label":"New Element",
        "label_location":"0",
        "x":"242",
        "y":"267",
        "url":"",
        "iconid_disabled":"0",
```

```
    "iconid_maintenance":"0"
},{
    "selementid":"100100000000415",
    "sysmapid":"100100000000031",
    "elementid":"0",
    "elementtype":"4",
    "iconid_off":"100100000000001",
    "iconid_on":"0",
    "iconid_unknown":"0",
    "label":"New Element",
    "label_location":"0",
    "x":"455",
    "y":"279",
    "url":"",
    "iconid_disabled":"0",
    "iconid_maintenance":"0"
},{
    "selementid":"100100000000416",
    "sysmapid":"100100000000031",
    "elementid":"0",
    "elementtype":"4",
    "iconid_off":"100100000000001",
    "iconid_on":"0",
    "iconid_unknown":"0",
    "label":"New Element",
    "label_location":"0",
    "x":"454",
    "y":"111",
    "url":"",
    "iconid_disabled":"0",
    "iconid_maintenance":"0"
},{
    "selementid":"100100000000417",
    "sysmapid":"100100000000031",
    "elementid":"100100000037426",
    "elementtype":"2",
    "iconid_off":"100100000000001",
    "iconid_on":"0",
    "iconid_unknown":"0",
    "label":"New Element",
    "label_location":"-1",
    "x":"222",
    "y":"68",
    "url":"",
    "iconid_disabled":"0",
    "iconid_maintenance":"0"
},{
    "selementid":"100100000000418",
    "sysmapid":"100100000000031",
    "elementid":"0",
    "elementtype":"4",
    "iconid_off":"100100000000001",
    "iconid_on":"0",
    "iconid_unknown":"0",
    "label":"New Element",
    "label_location":"0",
    "x":"103",
    "y":"150",
    "url":"",
    "iconid_disabled":"0",
    "iconid_maintenance":"0"
},{
    "selementid":"100100000000422",
```

```
        "sysmapid":"100100000000031",
        "elementid":"100100000000006",
        "elementtype":"1",
        "iconid_off":"100100000000005",
        "iconid_on":"100100000000013",
        "iconid_unknown":"0",
        "label":"{HOSTNAME}",
        "label_location":"2",
        "x":"570",
        "y":"197",
        "url":"",
        "iconid_disabled":"0",
        "iconid_maintenance":"0"
    }],
    "links":[{
        "linkid":"100100000000232",
        "sysmapid":"100100000000031",
        "selementid1":"100100000000414",
        "selementid2":"100100000000415",
        "drawtype":"0",
        "color":"0000CC",
        "label":"",
        "linktriggers":[{
            "linktriggerid":"100100000000326",
            "linkid":"100100000000232",
            "triggerid":"100100000037426",
            "drawtype":"0",
            "color":"0"
        }]
    },{
        "linkid":"100100000000233",
        "sysmapid":"100100000000031",
        "selementid1":"100100000000414",
        "selementid2":"100100000000415",
        "drawtype":"0",
        "color":"0000CC",
        "label":"",
        "linktriggers":[]
    },{
        "linkid":"100100000000234",
        "sysmapid":"100100000000031",
        "selementid1":"100100000000416",
        "selementid2":"100100000000417",
        "drawtype":"0",
        "color":"0000CC",
        "label":"\u0414\u042b\u0424\u041e\u0414 \u041b\u042b\u041e\u0424\u0414\u041b \u042b\u041e\u0424",
        "linktriggers":[]
    },{
        "linkid":"100100000000235",
        "sysmapid":"100100000000031",
        "selementid1":"100100000000416",
        "selementid2":"100100000000417",
        "drawtype":"0",
        "color":"0000CC",
        "label":"",
        "linktriggers":[{
            "linktriggerid":"100100000000327",
            "linkid":"100100000000235",
            "triggerid":"100100000037426",
            "drawtype":"0",
            "color":"0"
        }]
    },{
```

```
    "linkid":"100100000000236",
    "sysmapid":"100100000000031",
    "selementid1":"100100000000414",
    "selementid2":"100100000000415",
    "drawtype":"0",
    "color":"0000CC",
    "label":"",
    "linktriggers":[]
},{
    "linkid":"100100000000237",
    "sysmapid":"100100000000031",
    "selementid1":"100100000000414",
    "selementid2":"100100000000415",
    "drawtype":"0",
    "color":"0000CC",
    "label":"",
    "linktriggers":[]
},{
    "linkid":"100100000000238",
    "sysmapid":"100100000000031",
    "selementid1":"100100000000414",
    "selementid2":"100100000000415",
    "drawtype":"0",
    "color":"0000CC",
    "label":"",
    "linktriggers":[]
},{
    "linkid":"100100000000239",
    "sysmapid":"100100000000031",
    "selementid1":"100100000000414",
    "selementid2":"100100000000415",
    "drawtype":"0",
    "color":"0000CC",
    "label":"",
    "linktriggers":[]
},{
    "linkid":"100100000000311",
    "sysmapid":"100100000000031",
    "selementid1":"100100000000422",
    "selementid2":"100100000000415",
    "drawtype":"3",
    "color":"00AA00",
    "label":"",
    "linktriggers":[{
        "linktriggerid":"100100000000323",
        "linkid":"100100000000311",
        "triggerid":"100100000012794",
        "drawtype":"4",
        "color":"DD0000"
    },{
        "linktriggerid":"100100000000324",
        "linkid":"100100000000311",
        "triggerid":"100100000012795",
        "drawtype":"4",
        "color":"DD0000"
    },{
        "linktriggerid":"100100000000325",
        "linkid":"100100000000311",
        "triggerid":"100100000012796",
        "drawtype":"4",
        "color":"DD0000"
    }]
}]
```

```
}] ,
"id":2
}
```

#### update()

Available since version: **1.8**
The method is used to control all map attributes.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **sysmapid** | *string* | | Map ID. | |
| **map attribute** | *any* | Yes | New value for a map attribute. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Map IDs. |
| **error** | In case of any errors |

**Example** Set map name to ”New Name”:

```
{
 "jsonrpc":"2.0",
 "method":"map.update",
 "params":{
    "sysmapid": "100100000010092",
    "name": "New Name"
 },
 "auth":"700ca65537074ec963db7efabda78259",
 "id":2
}
```

Retrieved updated map IDs:

```
{
 "jsonrpc":"2.0",
 "result": {
     "sysmapids":["100100000010092"]
 },
 "id":2
}
```

## Mediatype

**Methods** Class containing methods for operations with Media types.

| Methods | Description |
|---|---|
| get() | Get media type details |
| create() | Create media types |
| update() | Update media type details |
| delete() | Delete media types |

**Object details**

**Media_type** The table contains complete list of Media types attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| **mediatypeid** | *integer* | Media type ID | |
| **description** | *string* | Name | |
| **type** | *integer* | Media type | 0 - Email, 1 - External script, 2 - SMS, 3 - Jabber, 100 - EzTexting |
| **smtp_server** | *string* | SMTP server name | |
| **smtp_helo** | *string* | HELO value for SMTP server | |
| **smtp_email** | *string* | Email address of Zabbix server | |
| **exec_path** | *string* | Name of external script | |
| **gsm_modem** | *string* | Serial device name of GSM modem | |
| **username** | *string* | User name | Jabber user name used by Zabbix server |
| **passwd** | *string* | User password | Jabber password used by Zabbix server |

**Common tasks** The table contains list of common user-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add a media type | Use method **mediatype.create** |
| Add a bunch of new media types | Use method **mediatype.create** with array of User group objects |
| Update media type | Use method **mediatype.update** with media type IDs |
| Retrieve media type details by Group IDs | Use method **mediatype.get** with parameter **mediatypeids** |
| Retrieve media type details by User group name | Use method **mediatype.get** with parameter **filter**, specify **"description":"<your mediatype>"** |

**create()**

This function allows you to create a mediatype as defined by the **mediatype data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **mediatype data** | *array or object* | | Array of Mediatype objects or a single object | **mediatypeid** shouldn't be specified |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Mediatype IDs. **mediatypeid** are assigned to each Mediatype object |
| **error** | In case of any errors |

**Example** Create two new mediatypes:

```
{
"jsonrpc": "2.0",
"method": "mediatype.create",
"params": [{
    "type": "0",
    "description": "Root Email",
    "smtp_server": "rootmail@domain.com",
```

```
    "smtp_helo": "domain.com",
    "smtp_email": "domain@domain.com",
    "exec_path": "",
    "gsm_modem": "",
    "username": "",
    "passwd": ""
},
{
    "type": "2",
    "description": "SMS",
    "smtp_server": "",
    "smtp_helo": "",
    "smtp_email": "",
    "exec_path": "",
    "gsm_modem": "\/dev\/ttyS0",
    "username": "",
    "passwd": ""
}],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 2
}
```

Mediatype created successfully:

```
{
"jsonrpc": "2.0",
"result": {
   "mediatypeids": ["100100000214797", "100100000214798"]
},
"id": 2
}
```

Mediatype already exists:

```
{
"jsonrpc": "2.0",
"error": {
      "code": -32602,
      "message": "Invalid params.",
      "data": "[ CMediatype::create ] Cannot create Mediatype"
},
"id": 2
}
```

**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several mediatypes. All mediatype-related information will be removed including triggers, empty graphs, child mediatypes, historical data.

**Parameters** Array of Mediatype IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Mediatype IDs. |
| **error** | In case of any errors |

**Example** Delete mediatypes by mediatype **ID**

```
{
"jsonrpc": "2.0",
"method": "mediatype.delete",
"params": ["107824", "107825"],
"auth": "3a57200802b24cda67c4e4010b50c065",
"id": 2
}
```

Mediatypes deleted successfully:

```
{
"jsonrpc": "2.0",
"result": {
   "mediatypeids": ["107824", "107825"]
},
"id": 2
}
```

Mediatypes do not exist:

```
{
"jsonrpc": "2.0",
"error": {
      "code": -32500,
      "message": "Application error.",
      "data": "[ CMediatype::delete ] Mediatype does not exist"
},
"id": 2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve mediatype details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **userids** | *array* | User IDs | |
| **mediaids** | *array* | User media IDs | |
| **mediatypeids** | *array* | Mediatype IDs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by mediatype fields | |
| **search** | *string* | Return mediatypes by given mediatype fields pattern | |
| **startSearch** | *integer* | Search given patterns only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result mediatypes by given patterns | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_users** | *string* | Select users | Values: shorten, refer, extend |
| **select_medias** | *string* | Select user media | Values: shorten, refer, extend |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **countOutput** | *integer* | Count mediatypes, returned the number of mediatypes found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by mediatype field | Values: mediatypeid |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number mediatype of objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Mediatype objects. |
| **error** | In case of any errors |

**Example** Get mediatypes details by mediatype description pattern ”sms” and limit output to 1 mediatype, return full **mediatype** object:

```
{
"jsonrpc": "2.0",
"method": "mediatype.get",
"params": {
    "search": {"description": "sms"},
    "output": "extend",
},
"auth": "6f38cddc44cfbb6c1bd186f9a220b5a0",
"id": 2
}
```

Retrieved mediatypes details:

```
{
"jsonrpc": "2.0",
"result": [{
    "mediatypeid": "100100000000003",
    "type": "2",
    "description": "SMS",
    "smtp_server": "",
    "smtp_helo": "",
    "smtp_email": "",
    "exec_path": "",
    "gsm_modem": "\/dev\/ttyS0",
    "username": "",
    "passwd": ""
}],
"id": 2
}
```

**update()**

Available since version: **1.8**
The method is used to control all mediatype attributes including mediatype applications linkage.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **mediatypeid** | *string* | | Mediatype ID. | |
| **mediatype attribute** | *any* | Yes | New value for a mediatype attribute. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Mediatype IDs. |
| **error** | In case of any errors |

**Example** Change SMTP server for mediatype:

```
{
 "jsonrpc": "2.0",
 "method": "mediatype.update",
 "params": [{
    "mediatypeid": "100100000010092",
    "smtp_server": "usdomain@usdomain.org"
 }],
 "auth": "700ca65537074ec963db7efabda78259",
 "id": 2
}
```

Retrieved updated mediatype IDs:

```
{
 "jsonrpc": "2.0",
 "result": {
     "mediatypeids": ["100100000010092"]
 },
 "id": 2
}
```

## Proxy

**Methods** Class containing methods for operations with Proxys.

| Methods | Description |
|---|---|
| get() | Get proxy details |

**Object details** The table contains complete list of Proxy attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| proxyid | *integer* | Proxy ID | |
| host | *string* | Proxy name. | |
| status | *integer* | Proxy Status. | |

**Common tasks** The table contains list of common proxy-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add a proxy | Use method **proxy.create** |
| Add a bunch of new proxys | Use method **proxy.create** with array of Proxy objects |
| Enable a proxy | Use method **proxy.update**, set **"status":5** |
| Disable a proxy | Use method **proxy.update**, set **"status":6** |

| Task | HOWTO |
|---|---|
| Retrieve proxy details by Proxy IDs | Use method **proxy.get** with parameter **proxyids** |
| Retrieve proxy details by Proxy name | Use method **proxy.get** with parameter **filter**, specify **"host":"<your proxy>"** |

**get()**

Available since version: **1.8**
This function allows you to retrieve proxy details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **proxyids** | *array* | Proxy IDs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by proxy fields | |
| **search** | *array* | Return proxies by any given proxy object field pattern | |
| **startSearch** | *integer* | Search proxies field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, proxies by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_hosts** | *string* | Select hosts | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count proxies, return the number of proxies found | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by proxy field | Values: hostid,host,status |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of proxy objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Proxy objects. |
| **error** | In case of any errors |

**Example** Get proxys details by proxy name pattern "proxy":

```
{
"jsonrpc":"2.0",
"method":"proxy.get",
"params":{
    "output":"extend",
    "search":{
        "host":["proxy"]
    }
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
```

```
"id":2
}
```

Retrieved proxy details:

```
{
"jsonrpc":"2.0",
"result":[{
    "proxyid":"100100000010017",
    "host":"Linux Proxy",
    "status":"0"
},{
    "proxyid":"100100000010229",
    "host":"ZABBIX Proxy",
    "status": 6
}],
"id":2
}
```

## Screen

**Methods** Class containing methods for operations with Screens.

| Methods | Description |
|---|---|
| get() | Get screen details |
| exists() | Check if screen exists |
| create() | Create screens |
| update() | Update screen details |
| delete() | Delete screens |

**Object details** The table contains complete list of Screen attributes.

**screen**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **screenid** | *integer* | Screen ID | |
| **name** | *integer* | Name | |
| **hsize** | *integer* | Horizontal size | |
| **vsize** | *integer* | Vertical size | |

**screenitem**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **screenid** | *integer* | | |
| **resourcetype** | *integer* | Screen item type | |
| **x** | *integer* | X position | |
| **y** | *integer* | Y position | |
| **resourceid** | *integer* | Depends on screen item type | |
| **width** | *integer* | Width | |
| **height** | *integer* | Height | |
| **colspan** | *integer* | Column span | |
| **rowspan** | *integer* | Row span | |
| **elements** | *integer* | Number of displayed lines | |
| **valign** | *integer* | Vertical align | |
| **halign** | *integer* | Horizontal align | |
| **style** | *integer* | Depends on screen item type | |
| **url** | *integer* | Opens URL on click | |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **dynamic** | *integer* | Dynamic screen items | |

**Common tasks** The table contains list of common screen-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add a screen | Use method **screen.create** |
| Add a bunch of new screens | Use method **screen.create** with array of Screen objects |
| Remove screen by Screen IDs | Use method **screen.delete** array of Screen **IDs** |
| Retrieve screen details by Screen IDs | Use method **screen.get** with parameter **screenids** |
| Retrieve screen details by Screen name | Use method **screen.get** with parameter **filter**, specify **”name”:”<your screen>”** |

**create()**

This function allows you to create a screen as defined by the **screen data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **screen data** | *array or object* | | Array of Screen objects with additional parameter **screenitems** | **screenid** shouldn’t be specified |
| **screenitems data** | *array or object* | | array of screen item objects | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Screen IDs. **screenid** are assigned to each Screen object |
| **error** | In case of any errors |

**Example** Create new screen

```
{
"jsonrpc":"2.0",
"method":"screen.create",
"params":[{
    "name":"ZABBIX Server",
    "hsize":"2",
    "vsize":"4",
    "screenitems":[{
        "resourcetype":"2",
        "resourceid":"100100000000002",
        "width":"0",
        "height":"0",
        "x":"0",
        "y":"0",
        "colspan":"2",
        "rowspan":"0",
        "elements":"0",
        "valign":"0",
        "halign":"0",
        "style":"0",
        "url":"",
```

```
        "dynamic":"0"
    },{
        "resourcetype":"0",
        "resourceid":"100100000000002",
        "width":"400",
        "height":"100",
        "x":"0",
        "y":"1",
        "colspan":"0",
        "rowspan":"0",
        "elements":"0",
        "valign":"0",
        "halign":"0",
        "style":"0",
        "url":"",
        "dynamic":"1"
    },{
        "resourcetype":"0",
        "resourceid":"100100000000003",
        "width":"400",
        "height":"100",
        "x":"0",
        "y":"3",
        "colspan":"0",
        "rowspan":"0",
        "elements":"0",
        "valign":"0",
        "halign":"0",
        "style":"0",
        "url":"",
        "dynamic":"0"
    },{
        "resourcetype":"0",
        "resourceid":"100100000000004",
        "width":"400",
        "height":"100",
        "x":"1",
        "y":"3",
        "colspan":"0",
        "rowspan":"0",
        "elements":"0",
        "valign":"0",
        "halign":"0",
        "style":"0",
        "url":"",
        "dynamic":"0"
    },{
        "resourcetype":"0",
        "resourceid":"100100000000005",
        "width":"400",
        "height":"100",
        "x":"1",
        "y":"2",
        "colspan":"0",
        "rowspan":"0",
        "elements":"0",
        "valign":"0",
        "halign":"0",
        "style":"0",
        "url":"",
        "dynamic":"0"
    },{
```

```
        "resourcetype":"0",
        "resourceid":"100100000000587",
        "width":"500",
        "height":"100",
        "x":"1",
        "y":"1",
        "colspan":"0",
        "rowspan":"0",
        "elements":"0",
        "valign":"0",
        "halign":"0",
        "style":"0",
        "url":"",
        "dynamic":"0"
    },{
        "resourcetype":"7",
        "resourceid":"0",
        "width":"500",
        "height":"100",
        "x":"0",
        "y":"2",
        "colspan":"0",
        "rowspan":"0",
        "elements":"0",
        "valign":"0",
        "halign":"0",
        "style":"0",
        "url":"",
        "dynamic":"0"
    }]
}],
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":2
}
```


Screen created successfully:


```
{
"jsonrpc":"2.0",
"result":{
    "screenids":["100100000012213"]
},
"id":2
}
```


Screen already exists:


```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CScreen::create ] Screen [ ZABBIX Server ] already exists"
},
"id":2
}
```


**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several screens. Screen items will be removed.

**Parameters** Array of Screen IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Screen IDs. |
| **error** | In case of any errors |

**Example** Delete screens by screen **ID**

```
{
"jsonrpc":"2.0",
"method":"screen.delete",
"params":["107824",  "107825"],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Screens deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "screenids": ["107824", "107825"]
},
"id":2
}
```

Screens does not exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CScreen::delete ] Screen does not exist"
},
"id":2
}
```

**exists()**

Available since version: **1.8.3**
This function allows you to check whether screen with given screen data exists.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **nodeids** | *array* | Yes | List of node IDs where to search for given screen | |
| **screenid** | *string* | Yes | Screen name | |
| **name** | *string* | Yes | Screen name | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain boolean variable. |
| **error** | In case of any errors |

**Example** Check if screen with name ”ZABBIX Server” exists

```
{
"jsonrpc":"2.0",
"method":"screen.exists",
"params":{
    "name": "ZABBIX Server"
},
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Screen exists:

```
{
"jsonrpc":"2.0",
"result": true,
"id":2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve Screen details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **screenids** | *array* | Screen IDs | |
| **screenitemids** | *array* | Screen item IDs | |
| **type** | *integer* | Screen type | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by screen fields | |
| **search** | *array* | Return screens by any given object field pattern | |
| **startSearch** | *integer* | Search screens field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, screens by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_screenitems** | *string* | Select screen items | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count screens, return the number of screens found | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by Screen field | Values: screenid, name |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *integer* | max number of screen objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Screen objects. |

| Parameter | Description |
|---|---|
| **error** | In case of any errors |

**Example** Get screens details by screen name pattern ”zabbix”:

```
{
"jsonrpc":"2.0",
"method":"screen.get",
"params":{
    "search": {"name": "zabbix"},
    "output": "extend"
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved screen details:

```
{
"jsonrpc":"2.0",
"result":[{
    "screenid":"100100000000002",
    "name":"Zabbix server",
    "hsize":"2",
    "vsize":"4"
},{
    "screenid":"100100000000007",
    "name":"ZABBIX",
    "hsize":"1",
    "vsize":"1"
}],
"id":2
}
```

**update()**

Available since version: **1.8**
The method is used to control all screen attributes including screen screens.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **screenid** | *string* | | Screen ID. | |
| **screen attribute** | *any* | Yes | New value for a screen attribute. | |
| **screenitems** | *any* | Yes | New screen item list. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Screen IDs. |
| **error** | In case of any errors |

**Example** Set screen name to ”New Name”:

```
{
 "jsonrpc":"2.0",
 "method":"screen.update",
 "params":{
```

```
        "screenid": "100100000010092",
        "name": "New Name"
    },
    "auth":"700ca65537074ec963db7efabda78259",
    "id":2
}
```

Retrieved updated screen IDs:

```
{
    "jsonrpc":"2.0",
    "result": {
        "screenids":["100100000010092"]
    },
    "id":2
}
```

## Script

**Methods** Class containing methods for operations with Scripts.

| Methods | Description |
|---|---|
| get() | Get script details |
| execute() | Check if script exists |
| create() | Create scripts |
| update() | Update script details |
| delete() | Delete scripts |

**Object details** The table contains complete list of Script attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| scriptid | *int* | Script ID | |
| name | *string* | Script description | |
| command | *string* | Command to execute | |
| host_access | *integer* | Needed host access for script execution | |
| usrgrpid | *integer* | User group ID | |
| groupid | *integer* | Host group ID | |

**Common tasks** The table contains list of common script-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add an script | Use method **script.create** |
| Add a bunch of new scripts | Use method **script.create** with array of Script objects |
| Retrieve script details by Script IDs | Use method **script.get** with parameter **scriptids** |
| Retrieve scripts details by Host name | Use method **script.get** with parameter **filter**, specify **"name": ["<your script>"]** |

### create()

This function allows you to create a script as defined by the **script data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **script data** | *array or object* | | Array of Script objects or a single object | **scriptid** shouldn't be specified |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Script IDs. **scriptid** are assigned to each Script object |
| **error** | In case of any errors |

**Example** Create new script for all hosts and host groups:

```
{
"jsonrpc":"2.0",
"method":"script.create",
"params":{
    "name":"Ping",
    "command":"\/bin\/ping -c 3 {HOST.CONN}",
    "host_access":"2",
    "usrgrpid":"0",
    "groupid":"0"
},
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":2
}
```

Script created successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "scriptids": ["100100000214797"]
},
"id":2
}
```

Script already exists:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CScript::create ] Cannot create Script"
},
"id":2
}
```

**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several scripts. All script-related information will be removed including triggers, empty graphs, child scripts, historical data.

**Parameters** Array of Script IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Script IDs. |
| **error** | In case of any errors |

**Example** Delete scripts by script **ID**

```
{
"jsonrpc":"2.0",
"method":"script.delete",
"params":["107824",  "107825"],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Scripts deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "scriptids": ["107824", "107825"]
},
"id":2
}
```

Scripts does not exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CScript::delete ] Script does not exist"
},
"id":2
}
```

**execute()**

**get()**

Available since version: **1.8**
This function allows you to retrieve script details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **groupids** | *array* | HostGroup IDs | |
| **hostids** | *array* | Host IDs | |
| **scriptids** | *array* | Script IDs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by script fields | |
| **search** | *string* | Return scripts by given script fields pattern | |
| **startSearch** | *integer* | Search given patterns only in start of the field | |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **excludeSearch** | *integer* | Exclude from result scripts by given patterns | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_groups** | *string* | Select host groups | Values: shorten, refer, extend |
| **select_hosts** | *string* | Select hosts | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count scripts, return number of scripts found | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by script field | Values: scriptid, name |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of script objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Script objects. |
| **error** | In case of any errors |

**Example** Get scripts details by script description pattern ”Apache” and limit output to 10 scripts, return only **script** IDS:

```
{
"jsonrpc":"2.0",
"method":"script.get",
"params":{
    "filter": {"name": "TEST"},
    "output": "extend",
    "limit": 1
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved scripts details:

```
{
"jsonrpc":"2.0",
"result":[{
    "groups":[{
        "groupid":"0"
    }],
    "scriptid":"100100000000006",
    "name":"TEST",
    "command":"\/bin\/ping -c 3 {HOST.CONN}",
    "host_access":"3",
    "usrgrpid":"100100000000002",
    "groupid":"0"
}],
"id":2
}
```

**update()**

Available since version: **1.8**
The method is used to control all script attributes including script applications linkage.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **scriptid** | *string* | | Script ID. | |
| **script attribute** | *any* | Yes | New value for a script attribute. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Script IDs. |
| **error** | In case of any errors |

**Example** Change command for script, .i.e set its command to '<your command>':

```
{
 "jsonrpc":"2.0",
 "method":"script.update",
 "params":{
    "scriptid": "100100000010092",
    "command": "\/bin\/fping -c 10 {HOST.CONN}"
 },
 "auth":"700ca65537074ec963db7efabda78259",
 "id":2
}
```

Retrieved updated script IDs:

```
{
 "jsonrpc":"2.0",
 "result": {
     "scriptids":["100100000010092"]
 },
 "id":2
}
```

## Template

Class containing methods for operations with Templates

**Methods** Class containing methods for operations with Templates.

| Methods | Description |
|---|---|
| get() | Get template details |
| exists() | Check if template exists |
| create() | Create templates |
| update() | Update template details |
| delete() | Delete templates |
| massAdd() | Mass add template linkage, hosts, macros, host groups |
| massUpdate() | Mass update template details, link templates, hosts, add host groups |
| massRemove() | Mass remove template linkage, hosts, macros, host groups |

**Object details** The table contains complete list of Template attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| templateid | *int* | Template ID | |
| host | *string* | Template name. | |
| groupids | *array* | HostGroup IDs add Template to. | |

**Common tasks** The table contains list of common template-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add a template | Use method **template.create** |
| Add a bunch of new templates | Use method **template.create** with array of Template objects |
| Link template to hosts | Use method **template.massAdd** with parameters **templates**, **hosts** |
| Unlink template from hosts | Use method **template.massRemove** with parameters **templates**, **hosts** |

**create()**

This function allows you to create a template as defined by the **template data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **template data** | *array or object* | | Array of Template objects or a single object | **templateid** shouldn't be specified |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Template IDs. **templateid** are assigned to each Template object |
| **error** | In case of any errors |

```
{
"jsonrpc":"2.0",
"method":"template.create",
"params":{
   "host":"Template Linux 2",
    "groups":[
         {
            "groupid":"100100000000001"
         }
    ],
    "templates":[
         {
            "templateid":"100100000010001"
         }
    ]
},
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":3
}
```

**Example** Template added successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "templateids":["100100000014794"]
},
"id":3
}
```

Template already exists:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CTemplate::create ] Template [ Template Linux 2 ] already exists"
},
"id":3
}
```

### delete()

Available since version: **1.8**
This function allows you to delete information about one or several templates. All template-related information will be removed including items, graphs, macros, application, etc.

#### Parameters

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **templateids** | *array* | | Array of Template IDs | |

#### Returns

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Template IDs. |
| **error** | In case of any errors |

#### Example

```
{
"jsonrpc":"2.0",
"method":"template.delete",
"params":[
      {
         "templateid":107824
      },
      {
         "templateid":107825
      }
],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Templates deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
```

```
   "templateids": ["107824", "107825"]
},
"id":2
}
```

Template does not exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CTemplate::delete ] Template does not exist"
},
"id":2
}
```

**exists()**

Available since version: **1.8.3**
This function allows you to check whether template with given template name or template ID exists.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **nodeids** | *array* | yes | List of node IDs where to search for given template ID or template name | |
| **hostid** | *string* | yes | Template ID | |
| **host** | *string* | yes | Template name | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain boolean variable. |
| **error** | In case of any errors |

**Example**

```
{
"jsonrpc":"2.0",
"method":"template.exists",
"params":{
   "nodeids": ["1"],
   "host": "Template Linux"
},
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Template exists:

```
{
"jsonrpc":"2.0",
"result": true,
"id":2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve template details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **groupids** | *array* | TemplateGroup IDs | |
| **templateids** | *array* | Template IDs | |
| **parentTemplateids** | *array* | Parent template IDs | |
| **hostids** | *array* | Host IDs | |
| **itemids** | *array* | Item IDs | |
| **triggerids** | *array* | Trigger IDs | |
| **graphids** | *array* | Graph IDs | |
| **proxyids** | *array* | Proxy IDs | |
| **maintenanceids** | *array* | Maintenance IDs | |
| **with_items** | *integer* | only with items | |
| **with_triggers** | *integer* | only with triggers | |
| **with_graphs** | *integer* | only with graphs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by template fields | |
| **search** | *array* | Return templates by any given template object field pattern | |
| **startSearch** | *integer* | Search templates field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, templates by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_groups** | *string* | Select template groups | Values: shorten, refer, extend |
| **selectParentTemplates** | *string* | Select parent templates of current template | Values: shorten, refer, extend |
| **select_templates** | *string* | Select child templates of current template | Values: shorten, refer, extend |
| **select_hosts** | *string* | Select linked hosts | Values: shorten, refer, extend |
| **select_items** | *string* | Select template items | Values: shorten, refer, extend |
| **select_triggers** | *string* | Select template triggers | Values: shorten, refer, extend |
| **select_graphs** | *string* | Select template graphs | Values: shorten, refer, extend |
| **select_applications** | *string* | Select template applications | Values: shorten, refer, extend |
| **select_macros** | *string* | Select template macros | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count templates, return the number of templates found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by template field | Values: hostid, host |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of template objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Template objects. |
| **error** | In case of any errors |

**Example** Get templates details by template name "Template Linux":

```
{
"jsonrpc":"2.0",
"method":"template.get",
"params":{
    "output":"extend",
    "filter":{
        "host":"Template_Linux"
        }
    },
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved template details:

```
{
"jsonrpc":"2.0",
"result":[
{
    "templateid":"100100000010001",
    "hostid":"100100000010001",
    "proxy_hostid":"0",
    "host":"Template_Linux",
    "dns":"",
    "useip":"0",
    "ip":"",
    "port":"10050",
    "status":"3",
    "disable_until":"0",
    "error":"",
    "available":"0",
    "errors_from":"0",
    "lastaccess":"0",
    "inbytes":"0",
    "outbytes":"0",
    "useipmi":"0",
    "ipmi_port":"623",
    "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_ip":"127.0.0.1",
    "ipmi_errors_from":"0",
    "snmp_errors_from":"0",
    "ipmi_error":"",
    "snmp_error":""
}
],
"id":2
}
```

**massAdd()**

Available since version: **1.8**
Mass add template linkage, host linkage, macros, host groups

**Parameters** multidimensional array with data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **templates** | *array* | | Template objects to update | |
| **templates_link** | *array* | Yes | Template objects which should be linked to templates. | |
| **hosts** | *array* | Yes | Host objects which should be linked to templates. | |
| **groups** | *array* | Yes | Host group objects where templates should be added. | |
| **macros** | *array* | Yes | Macros objects which should be added to templates. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Template IDs. |
| **error** | In case of any errors |

**Example** Link two Hosts to template with **ID** ”100100000010001”, and add this template to group with **ID** ”100100000000041”

```
{
"jsonrpc":"2.0",
"method":"template.massAdd",
"params":{
   "templates": [{"templateid": "100100000010001"}],
   "groups": [{"groupid": "100100000000041"}],
   "hosts": [{"hostid": "100100000010092"},{"hostid": "100100000011197"}]
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

Templates updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "templateids":["100100000010001"]
},
"id":2
}
```

**massRemove()**

Available since version: **1.8**
Mass remove template linkage, host linkage, macros, host groups

**Parameters** multidimensional array with data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **templateids** | *array* | | Templateids to update | |
| **templateids_link** | *array* | Yes | Templateids which should be unlinked from templates. | |
| **hostids** | *array* | Yes | Hostids which should be unlinked from templates. | |
| **groupids** | *array* | Yes | Host groupids where templates should be removed. | |
| **macros** | *array* | Yes | Macros which should be removed from templates. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Template IDs. |
| **error** | In case of any errors |

**Example** Unlink two Hosts from template with **ID** ”100100000010001”, and remove this template from group with **ID** ”100100000000041”

```
{
"jsonrpc":"2.0",
"method":"template.massRemove",
"params":{
   "templateids": ["100100000010001"],
   "groupids": ["100100000000041"],
   "hostids": ["100100000010092","100100000011197"]
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

Templates updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "templateids":["100100000010001"]
},
"id":2
}
```

**massUpdate()**

Available since version: **1.8**
==== Parameters ==== multidimensional array with Templates data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **templates** | *array* | | Template objects to update | |
| **host** | *string* | Yes | Template name. | |
| **groups** | *array* | Yes | Update templates Host Group linkage. Missing objects will be linked, existed stay, others unlinked | |
| **hosts** | *array* | Yes | Update templates Hosts linkage. Missing objects will be linked, existed stay, others unlinked | |
| **macros** | *array* | Yes | Update templates Macros. Missing objects will be added, existed updated, others removed | |
| **templates_link** | *array* | Yes | Update templates Template linkage. Missing objects will be linked, existed stay, others unlinked | |
| **templates_clear** | *array* | Yes | Templates that should be unlinked and cleared. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Template IDs. |
| **error** | In case of any errors |

**Example** Update template with **ID** ”100100000014792” so:

1. Add to group, and remove from others
2. Link host to this template and unlink others
3. Unlink and clear linked template

```
{
"jsonrpc":"2.0",
"method":"template.massUpdate",
"params":{
    "templates": [{"templateid": "100100000014792"}],
    "groups": [{"groupid": "100100000000041"}],
    "hosts": [{"hostid": "100100000010092"}],
    "templates_clear": [{"templateid": 100100000010232}]
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

Templates updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "templateids":["100100000014792"]
},
"id":2
}
```

**update()**

Available since version: **1.8**
The method is used to control all template attributes including template template linkage, macros and template group membership.
The method is a wrapper for **template.massUpdate** function.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **templateid** | *string* | | Template name. | |
| **host** | *any* | Yes | New name for a template. | |
| **groups** | *array* | Yes | Update templates Host Group linkage. Missing objects will be linked, existed stay, others unlinked | |
| **hosts** | *array* | Yes | Update templates Hosts linkage. Missing objects will be linked, existed stay, others unlinked | |
| **macros** | *array* | Yes | Update templates Macros. Missing objects will be added, existed updated, others removed | |
| **templates** | *array* | Yes | Update templates Template linkage. Missing objects will be linked, existed stay, others unlinked | |
| **templates_clear** | *array* | Yes | Templates that should be unlinked and cleared. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Template IDs. |
| **error** | In case of any errors |

**Example** Change template name, link it to two hosts and unlink from others

```
{
"jsonrpc":"2.0",
"method":"template.update",
"params":[{
    "templateid": "100100000014792",
    "host": "T1",
    "hosts": [{"hostid": "100100000010226" },{"hostid": "100100000010092"}]
}],
"auth":"700ca65537074ec963db7efabda78259",
"id":2
}
```

Retrieved updated template IDs:

```
{
"jsonrpc":"2.0",
"result": {
    "templateids":["100100000014792"]
},
"id":2
}
```

## Trigger

**Methods** Class containing methods for operations with Triggers.

| Methods | Description |
|---|---|
| get() | Get trigger details |
| exists() | Check if trigger exists |
| create() | Create triggers |
| update() | Update trigger details |
| delete() | Delete triggers |
| addDependencies() | Delete triggers |
| deleteDependencies() | Delete triggers |

**Object details** The table contains complete list of Trigger attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| triggerid | *integer* | Trigger **ID** | |
| description | *string* | Trigger name | |
| expression | *string* | Expression | |
| url | *string* | Referenced URL | |
| status | *integer* | Status | |
| value | *integer* | State | |
| priority | *integer* | Severity | |
| lastchange | *integer* | Time of last state change | |
| dep_level | *integer* | Dependency level | |
| comments | *integer* | Description | |
| error | *integer* | Error | |
| templateid | *integer* | Parent trigger **ID** | |
| type | *integer* | Event generation | |

**Field values**

**Status**

| Value | Type |
|---|---|
| 0 | Trigger is active |
| 1 | Trigger is disabled |

**Value**

| Value | Type |
|---|---|
| 0 | OK |
| 1 | PROBLEM |
| 2 | UNKNOWN |

**Priority**

| Value | Type |
|---|---|
| 0 | Not classified |
| 1 | Information |
| 2 | Warning |
| 3 | Average |
| 4 | High |

| Value | Type |
|---|---|
| 5 | Disaster |

**Type**

| Value | Type |
|---|---|
| 0 | Normal event generation |
| 1 | Generate multiple PROBLEM events |

**Common tasks** The table contains list of common trigger-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add an trigger | Use method **trigger.create** |
| Add a bunch of new triggers | Use method **trigger.create** with array of Trigger objects |
| Enable an trigger | Use method **trigger.update**, set **"status":0** |
| Disable an trigger | Use method **trigger.update**, set **"status":1** |
| Retrieve trigger details by Trigger IDs | Use method **trigger.get** with parameter **triggerids** |
| Retrieve triggers details by Host name | Use method **trigger.get** with parameter **filter**, specify **"host": ["<your host1>"]** |

**addDependencies()**

This function allows you to create trigger dependencies.

**Parameters** Array of hashes

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **triggerid** | *integer* | Child trigger **ID** | | |
| **dependsOnTriggerid** | *integer* | Parent trigger **ID** | | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Trigger IDs. |
| **error** | In case of any errors |

**Example** Add dependencies to trigger with trigger **ID** "100100000064544", so it would depend on triggers with trigger **ID** "100100000064537" and "100100000064538"

```
{
"jsonrpc":"2.0",
"method":"trigger.addDependencies",
"params":[
    {
        "triggerid": "100100000064544",
        "dependsOnTriggerid": "100100000064537"
    },
    {
        "triggerid": "100100000064544",
        "dependsOnTriggerid": "100100000064538"
    }
],
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":2
}
```

Trigger updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "triggerids": ["100100000064544"]
},
"id":2
}
```

**create()**

This function allows you to create a trigger as defined by the **trigger data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **trigger data** | *array or object* | | Array of Trigger objects or a single object | **triggerid** shouldn't be specified |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Trigger IDs. **triggerid** are assigned to each Trigger object |
| **error** | In case of any errors |

**Example** Create new trigger for host "ZABBIX-Server" and enable it

```
{
"jsonrpc": "2.0",
"method": "trigger.create",
"params": [{
    "description": "TEST_MACRO",
    "expression": "{ZABBIX-Server:vfs.fs.inode[/,{$MACRO}].max(\"{$MACRO2}\")}={$MACRO3}",
    "status": 0
}],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 2
}
```

Trigger created successfully:

```
{
"jsonrpc": "2.0",
"result": {
   "triggerids": ["100100000214797"]
},
"id": 2
}
```

Trigger already exists:

```
{
"jsonrpc": "2.0",
"error": {
      "code": -32602,
      "message": "Invalid params.",
      "data": "[ CTrigger::create ] Cannot create Trigger"
},
```

```
"id": 2
}
```

**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several triggers. All trigger-related information will be removed including events, map elements, IT services, action conditions dependencies.

**Parameters** Array of Item IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Host IDs. |
| **error** | In case of any errors |

**Example** Delete triggers by trigger **ID**

```
{
"jsonrpc":"2.0",
"method":"trigger.delete",
"params":["107824",  "107825"],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Triggers deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "triggerids": ["107824", "107825"]
},
"id":2
}
```

Triggers does not exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CTrigger::delete ] Trigger does not exist"
},
"id":2
}
```

**deleteDependencies()**

This function allows you to remove trigger dependencies.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **trigger data** | *array or object* | | Array of Trigger objects or a single object | **triggerid** must be specified |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Trigger IDs. |
| **error** | In case of any errors |

**Example** Remove all dependencies from triggers with trigger **ID** ”100100000064544” and ”100100000064545”

```
{
"jsonrpc":"2.0",
"method":"trigger.deleteDependencies",
"params":[
    {"triggerid": "100100000064544"},
    {"triggerid": "100100000064545"}
],
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":2
}
```

Triggers updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "triggerids":["100100000064544","100100000064545"]
},
"id":2
}
```

**exists()**

Available since version: **1.8.3**
This function allows you to check whether trigger with given trigger data exists.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **nodeids** | *array* | yes | List of node IDs where to search for given item | |
| **description** | *string* | No | Trigger description | |
| **expression** | *string* | No | Trigger expression | |
| **hostid** | *string* | yes | Host ID | |
| **host** | *string* | yes | Host name | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain boolean variable. |
| **error** | In case of any errors |

**Example** Check if trigger with description "APC: System UPS Load" and expression "{TEST_hp3000:vfs.file.cksum[c:\config.sys].last(0)}=0" exists

```
{
"jsonrpc":"2.0",
"method":"trigger.exists",
"params":{
    "host":"ZABBIX-Server",
    "description":"Apache is not running on {HOSTNAME}",
    "expression":"{ZABBIX-Server:proc_cnt[httpd].last(0)}<1"
},
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Trigger exists:

```
{
"jsonrpc":"2.0",
"result": true,
"id":2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve trigger details based on filtering options. All parameters are optional. If parameter is set in query, this option is considered as being ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **groupids** | *array* | HostGroup IDs | |
| **templateids** | *array* | Template IDs | |
| **hostids** | *array* | Host IDs | |
| **triggerids** | *array* | Trigger IDs | |
| **itemids** | *array* | Item IDs | |
| **applicationids** | *array* | Application IDs | |
| **functions** | *array* | Trigger functions | |
| **inherited** | *integer* | Inherited from template s | "0" - not inherited, "1" - inherited |
| **templated** | *integer* | Templated triggers | "0" - belongs to hosts, "1" - belongs to templates |
| **monitored** | *integer* | Monitored triggers | Checks trigger, item and host status |
| **active** | *integer* | Monitored triggers | Checks trigger and host status |
| **maintenance** | *integer* | Triggers in maintenance | |
| **withUnacknowledgedEvents** | *integer* | Triggers with unacknowledged events | |
| **withAcknowledgedEvents** | *integer* | Triggers with acknowledged events | |
| **withLastEventUnacknowledged** | *integer* | Triggers with last unacknowledged events | |
| **skipDependent** | *integer* | Do not select dependent triggers in PROBLEM state | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **lastChangeSince** | *string* | Optional filter by last changed state time | |
| **lastChangeTill** | *string* | Optional filter by last changed state time | |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **group** | *string* | Optional filter by host group name | |
| **host** | *string* | Optional filter by host name | |
| **only_true** | *string* | Triggers in state PROBLEM and recently switched (30 min) | |
| **min_severity** | *string* | Optional filter by severity | |
| **filter** | *array* | Optional filter by trigger fields | |
| **search** | *array* | Return triggers by any given object field pattern | |
| **startSearch** | *integer* | Search triggers field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, triggers by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **expandData** | *string* | Adds additional fields to triggers default | host, hostid |
| **expandDescription** | *string* | Expands trigger description | Expands macros |
| **select_groups** | *string* | Select host groups | Values: shorten, refer, extend |
| **select_hosts** | *string* | Select hosts | Values: shorten, refer, extend |
| **select_items** | *string* | Select trigger items | Values: shorten, refer, extend |
| **select_functions** | *string* | Select trigger functions | Values: shorten, refer, extend |
| **select_dependencies** | *string* | Select trigger dependencies | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count triggers, return the number of triggers found | |
| **groupCount** | *integer* | Return the number of results grouped by given IDs | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by trigger field | Values: trigerid,description,status,priority,lastchar |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of trigger objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of trigger objects. |
| **error** | In case of any errors |

**Example** Get triggers details by trigger descriptions ”APC: System UPS Global State”, ”APC: System UPS Load” in host ”ZABBIX-Server”:

```
{
"jsonrpc": "2.0",
"method": "trigger.get",
"params": {
    "filter": {
        "host": ["ZABBIX-Server"],
        "description": ["APC: System UPS Global State", "APC: System UPS Load"]
    },
    "output": "extend"
},
"auth": "6f38cddc44cfbb6c1bd186f9a220b5a0",
```

```
"id": 2
}
```


Retrieved trigger details:


```
{
"jsonrpc": "2.0",
"result": [{
    "triggerid": "100100000013502",
    "expression": "{100100000013078}=0",
    "description": "APC: System UPS Global State",
    "url": "",
    "status": "0",
    "value": "2",
    "priority": "1",
    "lastchange": "1277987805",
    "dep_level": "0",
    "comments": "System UPS Global State",
    "error": "Zabbix was restarted.",
    "templateid": "0",
    "type": "0"
},
{
    "triggerid": "100100000013503",
    "expression": "{100100000013077}=0",
    "description": "APC: System UPS Load",
    "url": "",
    "status": "0",
    "value": "2",
    "priority": "2",
    "lastchange": "1273213952",
    "dep_level": "0",
    "comments": "System UPS Load",
    "error": "Host is unavailable.",
    "templateid": "0",
    "type": "0"
}],
"id": 2
}
```


**update()**

Available since version: **1.8**
The method is used to control all trigger attributes including trigger applications linkage.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **triggerid** | *string* | | Trigger ID. | |
| **trigger attribute** | *any* | Yes | New value for a trigger attribute. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Trigger IDs. |
| **error** | In case of any errors |

**Example** Enable trigger, .i.e set its status to '0':

```
{
 "jsonrpc":"2.0",
 "method":"trigger.update",
 "params":[{
    "triggerid": "100100000010092",
    "status": 0
 }],
 "auth":"700ca65537074ec963db7efabda78259",
 "id":2
}
```

Retrieved updated trigger IDs:

```
{
 "jsonrpc":"2.0",
 "result": {
     "triggerids":["100100000010092"]
 },
 "id":2
}
```

## User

**Methods** Class containing methods for operations with users. Only super admins have access to all users.

| Methods | Description |
|---|---|
| get() | Get user details |
| create() | Create users |
| update() | Update user details |
| updateProfile() | Update user profile |
| delete() | Delete users |
| addMedia() | Add user media |
| updateMedia() | Update user media |
| deleteMedia() | Remove user media |
| authenticate() | Authenticate |
| login() | Login |
| logout() | Logout |

### Object details

**User** The table contains complete list of user attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| **userid** | *integer* | User ID | |
| **alias** | *string* | Login | |
| **name** | *string* | Name | |
| **surname** | *string* | Surname | |
| **passwd** | *string* | Password | md5 |
| **url** | *string* | Url to open after user login | |
| **autologin** | *integer* | Auto login | |
| **autologout** | *integer* | Auto logout | In seconds, 0 - disabled |
| **lang** | *string* | Locale | |
| **refresh** | *integer* | Page refresh period | |
| **type** | *integer* | User type | |
| **theme** | *string* | Theme | |
| **attempt_failed** | *integer* | Number of failed login attempts | |
| **attempt_ip** | *string* | Last used IP to login | |

| Parameter | Type | Description | Details |
|---|---|---|---|
| **attempt_clock** | *integer* | Last login attempt date | |
| **rows_per_page** | *integer* | Rows per page to show | |

**User Media** The table contains complete list of user media attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| **mediaid** | *integer* | User media ID | |
| **userid** | *integer* | User ID | |
| **mediatypeid** | *integer* | User media type ID | |
| **sendto** | *string* | Where to send | |
| **active** | *integer* | Enabled or disabled this media | |
| **severity** | *integer* | Trigger severity | bit arithmetics |
| **period** | *string* | User media period | |

**Common tasks** The table contains list of common user-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add a user | Use method **user.create** |
| Add a bunch of new users | Use method **user.create** with array of User objects |
| Rename user | Use method **user.update**, set **”name”:”<new name>”** |
| Retrieve user details by User IDs | Use method **user.get** with parameter **userids** |
| Retrieve user details by User alias | Use method **user.get** with parameter **filter**, specify **”alias”:”<user alias>”** |

**addMedia()**

Available since version: **1.8**

**Parameters** Multidimensional array with user data and user media data.

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **users** | *array* | | User objects to update | |
| **medias** | *array* | Yes | Media objects, which should be added to users. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain an array of updated user IDs. |
| **error** | In case of any errors. |

**Example** Add two enabled media to user with **ID** ”100100000010092”

```
{
"jsonrpc":"2.0",
"method":"user.addMedia",
"params":{
"users": [
  { "userid": "100100000010092" }
],
"medias": [
```

```
  {"mediatypeid": "100100000000001", "sendto": "zabbix@test.com", "active": "0", "severity": "56", "period
  {"mediatypeid": "100100000000002", "sendto": "zabbix@test.com", "active": "0", "severity": "63", "period
]
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

User media added successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "userids":["100100000010092"]
},
"id":2
}
```

**authenticate()**

Available since version: **1.8**

**Attention:**
Note: This method is obsolete. Use method **user.login** instead.

**create()**

This function allows you to create a user as defined by the **user data** array. Available only to super admins.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **user data** | *array or object* | No | Array of user objects or a single object | **userid** shouldn't be specified |
| **usrgrps** | *array* | No | User groups to add user to. | |
| **user_medias** | *array* | No | Create user media for user. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain an array of created user IDs. **userid** is assigned to each user object. |
| **error** | In case of any errors. |

**Example** Create new user and add it to 3 user groups. Password "zabbix" will automatically be encoded by MD5 hash function.

```
{
"jsonrpc":"2.0",
"method":"user.create",
"params":[{
    "usrgrps":[{
        "usrgrpid":"100100000000009",
        "name":"Internal login"
    },{
        "usrgrpid":"100100000000020",
        "name":"API access"
```

```
    },{
        "usrgrpid":"100100000000022",
        "name":"Debug group"
    }],
    "alias":"Test User",
    "name":"Test User Name",
    "surname":"Test User Surname",
    "passwd":"zabbix",
    "url":"",
    "autologin":"0",
    "autologout":"600",
    "lang":"en_gb",
    "refresh":"90",
    "type":"1",
    "theme":"css_ob.css",
    "attempt_failed":"0",
    "attempt_ip":"",
    "attempt_clock":"0",
    "rows_per_page":"50"
}],
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":3
}
```

User added successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "userids": ["107819"]
},
"id":3
}
```

User already exists:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CUser::create ] User [ Admin ] already exists"
},
"id":3
}
```

**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several users. All user-related information will be removed. Method available only to super admins.

**Parameters** Array of User IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted User IDs. |
| **error** | In case of any errors |

**Example** Delete users by User **ID**

```
{
"jsonrpc":"2.0",
"method":"user.delete",
"params":[
      {
         "userid":"107824"
      },
      {
         "userid":"107825"
      }
],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

Users deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "userids": ["107824", "107825"]
},
"id":2
}
```

User does not exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CUser::delete ] User does not exist"
},
"id":2
}
```

**deleteMedia()**

Available since version: **1.8**

**Parameters** Array with media data, media **ID** must be specified.

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **medias** | *array* | Yes | Media objects which should be deleted (removed / added). | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain an array of deleted user media IDs. |
| **error** | In case of any errors |

**Example** Remove two user media from user with **ID** ”100100000010092”

```
{
"jsonrpc":"2.0",
```

```
"method":"user.deleteMedia",
"params":[
  {"mediaid": "100100000000011"},
  {"mediaid": "100100000000012"}
],
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

User media deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "mediaids":["100100000000011", "100100000000012"]
},
"id":2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve user details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL, Only super admins have access to all users. Admin users may see only users sharing the same user groups. Simple users may gain info only about them selfs.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **usrgrpids** | *array* | User Group IDs | |
| **userids** | *array* | User IDs | |
| **mediaids** | *array* | Media IDs | |
| **mediatypeids** | *array* | Media type IDs | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by user fields | |
| **search** | *array* | Return users by any given object field pattern | |
| **startSearch** | *integer* | Search users field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, users by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_usrgrps** | *string* | Select user groups | Values: refer, extend |
| **select_mediatypes** | *string* | Select user media types | Values: refer, extend |
| **get_access** | *string* | Get additional info about user access to GUI | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count users, return the number of users found | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by user field | Values: userid, alias |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of user objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of User objects. |
| **error** | In case of any errors |

**Example** Get users details by user alias "Admin":

```
{
"jsonrpc":"2.0",
"method":"user.get",
"params":{
    "filter":{ "alias":["Admin"] },
    "output":"extend"
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved user details:

```
{
"jsonrpc":"2.0",
"result":[{
    "userid":"100100000000001",
    "alias":"Admin",
    "name":"admin",
    "surname":"admin",
    "url":"",
    "autologin":"1",
    "autologout":"0",
    "lang":"en_gb",
    "refresh":"2000",
    "type":"3",
    "theme":"css_od.css",
    "attempt_failed":"0",
    "attempt_ip":"127.0.0.1",
    "attempt_clock":"1281014721",
    "rows_per_page":"100"
}],
"id":2
}
```

**login()**

Available since version: **1.8**
This function allows you to login to ZABBIX based on accepted parameters. This method mainly used for login from scripts with retrieving only authentication token. All parameters are mandatory.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **user** | *string* | User login name | |
| **password** | *string* | User login password | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain authentication string. |
| **error** | In case of any errors |

**Example** Login in to ZABBIX by user alias "Admin" and password "zabbix":

```
{
"jsonrpc":"2.0",
"method":"user.login",
"params":{
    "user": "Admin",
    "password":"zabbix"
},
"id":1
}
```

Retrieved user authentication token:

```
{
"jsonrpc":"2.0",
"result":"a9a1f569d10d6339f23c4d122a7f5c46",
"id":1
}
```

**logout()**

**update()**

Available since version: **1.8**
The method is used to control all user attributes including user group linkage. The method is available only to super admins.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **userid** | *string* | | User ID. | |
| **user attribute** | *any* | Yes | New value for a user attribute. | |
| **usrgrps** | *any* | Yes | New list of user groups. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated User IDs. |
| **error** | In case of any errors |

**Example** Rename user, .i.e set its name to 'New user name':

```
{
 "jsonrpc":"2.0",
 "method":"user.update",
 "params":{
    "userid": "100100000010092",
    "name": "New user name"
 },
 "auth":"700ca65537074ec963db7efabda78259",
 "id":2
}
```

Retrieved updated user IDs:

```
{
 "jsonrpc":"2.0",
 "result": {
```

```
        "userids":["100100000010092"]
    },
    "id":2
  }
```

**updateMedia()**

Available since version: **1.8**

**Parameters** Multidimensional array with user data and user media data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **users** | *array* | | User objects to update | |
| **medias** | *array* | Yes | Media objects which should be updated (removed / added). | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated User IDs. |
| **error** | In case of any errors |

**Example** Add two user media, remove all current for user with **ID** ”100100000010092”

```
{
"jsonrpc":"2.0",
"method":"user.updateMedia",
"params":{
"users": [
  { "userid": "100100000010092" }
],
"medias": [
  {"mediatypeid": "100100000000001", "sendto": "zabbix@test.com", "active": "2", "severity": "56", "period
  {"mediatypeid": "100100000000002", "sendto": "zabbix@test.com", "active": "2", "severity": "63", "period
]
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

User media updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "userids":["100100000010092"]
},
"id":2
}
```

**updateProfile()**

Available since version: **1.8**
The method is used to control most user attributes. By this method user may change only it’s own settings.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **user attribute** | *any* | Yes | New value for a user attribute. | password, url, autologin, autologut, locale, theme, refresh period, rows per page |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated User Object. |
| **error** | In case of any errors |

**Example** Change rows per page shown by frontend to 50:

```
{
 "jsonrpc":"2.0",
 "method":"user.updateProfile",
 "params":{
    "rows_per_page": "50"
 },
 "auth":"700ca65537074ec963db7efabda78259",
 "id":2
}
```

Retrieved updated user IDs:

```
{
"jsonrpc":"2.0",
"result": [{
    "userid":"100100000000001",
    "alias":"Admin",
    "name":"admin",
    "surname":"admin",
    "passwd":"7815696ecbf1c96e6894b779456d330e",
    "url":"",
    "autologin":"1",
    "autologout":"0",
    "lang":"en_gb",
    "refresh":"2000",
    "type":"3",
    "theme":"css_od.css",
    "attempt_failed":"0",
    "attempt_ip":"127.0.0.1",
    "attempt_clock":"1281014721",
    "rows_per_page":"100"
}],
"id":2
}
```

## Usergroup

**Methods** Class containing methods for operations with User groups.

| Methods | Description |
|---|---|
| get() | Get user group details |
| exists() | Check if user group exists |
| create() | Create user groups |
| update() | Update user group details |
| delete() | Delete user groups |
| massAdd() | Mass add rights, users to user groups |
| massUpdate() | Mass update user group details, update list of rights, users |
| massRemove() | Mass remove rights, users |

**Object details**

**Usrgrp** The table contains complete list of User Group attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| **usrgrpid** | *integer* | User group id | |
| **name** | *string* | Name | |
| **gui_access** | *integer* | GUI access | system default(0), internal(1), disabled(2) |
| **users_status** | *integer* | User status | enabled(0), disabled(1) |
| **api_access** | *integer* | API access | disabled(0), enabled(1) |
| **debug_mode** | *integer* | Debug mode | disabled(0), enabled(1) |

**Rights** The table contains complete list of Rights attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| groupid | *integer* | User group ID | |
| id | *integer* | Host Group ID. | |
| permission | *string* | Permission. | deny(0), read(2), read-write(3) |

**Common tasks** The table contains list of common user-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add a user group | Use method **usergroup.create** |
| Add a bunch of new user groups | Use method **usergroup.create** with array of User group objects |
| Add users to user group | Use method **usergroup.massAdd** with array of user IDs |
| Add a host group with read-write or read permissions to user group | Use method **usergroup.massAdd** with array of rights objects |
| Retrieve user group details by Group IDs | Use method **usergroup.get** with parameter **usrgrpids** |
| Retrieve user group details by User group name | Use method **usergroup.get** with parameter **filter**, specify **”name”:”<your usergroup>”** |

**create()**

This function allows you to create a user group as defined by the **user group data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **usergroup data** | *array or object* | | Array of User group objects or a single object | **usrgrpid** shouldn’t be specified |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created User group IDs. **usrgrpid** are assigned to each User group object |
| **error** | In case of any errors |

**Example**

```
{
"jsonrpc":"2.0",
"method":"usergroup.create",
"params":[
   {"name":"Debug Group"}
],
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":3
}
```

User group created successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "usrgrpids": ["107819"]
},
"id":3
}
```

User group already exists:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CUserGroup::create ] User group [ Debug Group ] already exists"
},
"id":3
}
```

**delete()**

Available since version: **1.8**
This function allows you to delete information about one or several user groups.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **usrgrpids** | *array* | | Array of User Group IDs | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted User Group IDs. |
| **error** | In case of any errors |

**Example**

```
{
"jsonrpc":"2.0",
"method":"usergroup.delete",
"params":["107824", "107825"],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

User Groups deleted successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "usrgrpids": ["107824", "107825"]
},
"id":2
}
```

**exists()**

Available since version: **1.8.3**
This function allows you to check whether user group with given user group name or user group ID exists.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **nodeids** | *array* | yes | List of node IDs where to search for given user group ID or name | |
| **usrgrpid** | *string* | yes | User group ID | |
| **name** | *string* | yes | User group name | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain boolean variable. |
| **error** | In case of any errors |

**Example**

```
{
"jsonrpc":"2.0",
"method":"usergroup.exists",
"params":{
    "nodeids": ["1"],
    "name": "Admin Group"
},
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```

User group exists:

```
{
"jsonrpc":"2.0",
"result": true,
"id":2
}
```

**get()**

Available since version: **1.8**
This function allows you to retrieve user group details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL.

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **usrgrpids** | *array* | UserGroup IDs | |
| **userids** | *array* | User IDs | |
| **status** | *boolean* | | |
| **with_gui_access** | *boolean* | | |
| **with_api_access** | *boolean* | | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by user group fields | |
| **search** | *array* | Return user groups by any given user group object field pattern | |
| **startSearch** | *integer* | Search user groups field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, user groups by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **select_users** | *string* | Select contained users | Values: refer, extend |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count user groups, return the number of user groups found | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by user group field | Values: usrgrpid, name |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | Max number of user group objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of User group objects. |
| **error** | In case of any errors |

**Example** Get details for user groups with names ”Debug group”,”Zabbix administrators”, and select users **ID** in those groups :

```
{
"jsonrpc":"2.0",
"method":"usergroup.get",
"params":{
    "filter":{"name":["Debug group","Zabbix administrators"]},
    "select_users":"refer",
    "output":"extend"
},
"auth":"6f38cddc44cfbb6c1bd186f9a220b5a0",
"id":2
}
```

Retrieved details for user groups with names ”Debug group”,”Zabbix administrators” :


```
{
"jsonrpc":"2.0",
"result":[
{
    "users":[
        {"userid":"100100000000005"}
    ],
    "usrgrpid":"100100000000007",
    "name":"Zabbix administrators",
    "gui_access":"0",
    "users_status":"0",
    "api_access":"0",
    "debug_mode":"0"
},
{
    "users":[
        {"userid":"100100000000005"},
        {"userid":"100100000000001"},
        {"userid":"100100000000003"},
        {"userid":"100100000000004"},
        {"userid":"100100000000018"}
    ],
    "usrgrpid":"100100000000022",
    "name":"Debug group",
    "gui_access":"0",
    "users_status":"0",
    "api_access":"0",
    "debug_mode":"1"
}],
"id":2
}
```


**massAdd()**

Available since version: **1.8**
This method is used to link users or rights with user groups. Available only to super admins.

**Parameters** Multidimensional array with User groups data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **usrgrpids** | *array* | | User group IDs. | |
| **userids** | *array* | Yes | User IDs. | Those users will be added to all listed user groups in request. |
| **rights** | *array* | Yes | Host group rights | Those rights will be added to all listed user groups in request. |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated user group IDs. |
| **error** | In case of any errors |

**Example I** Add two users with **ID** ”100100000010092”, ”100100000010086” to user group with **ID** ”100100000000013”

```
{
"jsonrpc":"2.0",
"method":"usergroup.massAdd",
"params":{
    "usrgrpids": ["100100000000042"],
    "userids": ["100100000010092",  "100100000010086"],
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

User groups updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "usrgrpids":["100100000000042"]
},
"id":2
}
```

**Example II** Add read rights on host group with **ID** ”100100000010092” and read-write rights on host group with **ID** ”100100000010093” to user group with **ID** ”100100000000013”

```
{
"jsonrpc":"2.0",
"method":"usergroup.massAdd",
"params":{
    "usrgrpids": ["100100000000043"],
    "rights": [
        {"permission": 2, "id": "100100000010092"},
        {"permission": 3, "id": "100100000010093"}
    ],
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

User groups updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "usrgrpids": ["100100000000043"]
},
"id":2
}
```

**massRemove()**

**massUpdate()**

Available since version: **1.8**
This method is used to link users or rights with user groups. Available only to super admins.

**Parameters** Multidimensional array with User groups data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **usrgrpids** | *array* | | User group IDs. | |
| **userids** | *array* | Yes | User IDs. | New users will be added, missed removed for listed user groups in request. |
| **rights** | *array* | Yes | Host group rights | New rights will be added, existed updated, missed removed for listed user groups in request. |
| **user group attribute** | *any* | Yes | New value for a user group attribute. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated user group IDs. |
| **error** | In case of any errors |

**Example** Update rights on host groups with **ID** ”100100000010092”, ”100100000010093” for user group with **ID** ”100100000000013” and rename user group to ”Renamed”

```
{
"jsonrpc":"2.0",
"method":"usergroup.massUpdate",
"params":{
    "usrgrpids": ["100100000000043"],
    "rights": [
        {"permission": 3, "id": "100100000010092"},
        {"permission": 2, "id": "100100000010093"}
    ],
    "name": "Renamed"
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

User groups updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "usrgrpids": ["100100000000043"]
},
"id":2
}
```

**update()**

Available since version: **1.8**
The method is used to control user group attributes.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **user group attribute** | *any* | Yes | New value for a user group attribute. | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated User group IDs. |
| **error** | In case of any errors |

**Example** Rename user group:

```
{
 "jsonrpc":"2.0",
 "method":"usergroup.update",
 "params":[
     {"usrgrpid": "100100000000042", "name": "Rename 1"},
     {"usrgrpid": "100100000000013", "name": "Rename 2"}
 ],
 "auth":"700ca65537074ec963db7efabda78259",
 "id":2
}
```

Retrieved updated user IDs:

```
{
  "jsonrpc":"2.0",
  "result": {
  "usrgrpids":["100100000000042","100100000000013"]
  },
  "id":2
}
```

## Usermacro

**Methods** Class containing methods for operations with Usermacros.

| Methods | Description |
|---|---|
| get() | Get usermacro details |
| createGlobal() | Create global usermacros |
| updateGlobal() | Update global usermacros details |
| deleteGlobal() | Delete global usermacros |
| deleteHostMacro() | Delete host usermacros |
| massAdd() | Add usermacros to hosts or templates |
| massUpdate() | Update usermacros for hosts or templates |
| massRemove() | Remove usermacros from hosts or templates |

**Object details**

**Host Macro** The table contains complete list of Usermacro attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| hostmacroid | *integer* | Host macro ID | |
| hostid | *integer* | Host ID | |
| macro | *string* | Name | Name is unique per single host |
| value | *string* | Value | |

**Global Macro** The table contains complete list of Global Usermacro attributes.

| Parameter | Type | Description | Details |
|---|---|---|---|
| hostmacroid | *integer* | Host macro ID | |
| macro | *string* | Macro | Name is unique for global usermacros |
| value | *string* | Value | |

**Common tasks** The table contains list of common usermacro-related tasks and possible implementation using Zabbix API

| Task | HOWTO |
|---|---|
| Add a usermacro | Use method **usermacro.massAdd**, set hostids and macro objects |
| Add a global usermacro | Use method **usermacro.createGlobal** |
| Retrieve usermacro details by Usermacro IDs | Use method **usermacro.get** with parameter **usermacroids** |
| Retrieve usermacro details by Usermacro name | Use method **usermacro.get** with parameter **filter**, specify **"macro":"<your usermacro>"** |

**createGlobal()**

This method allows you to create a globalmacro as defined by the **globalmacro data** array.

**Parameters**

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **globalmacro data** | *array or object* | No | Array of Globalmacro objects or a single object | **globalmacroid** shouldn't be specified |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of created Globalmacro IDs. **globalmacroid** are assigned to each Globalmacro object |
| **error** | In case of any errors |

```
{
"jsonrpc":"2.0",
"method":"usermacro.createGlobal",
"params":[
    {
        "macro":"{$MACRO1}",
        "value":"192.168.0.1"
    },
    {
        "macro":"{$MACRO2}",
        "value":"192.168.0.2"
    }
],
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":3
}
```

**Example** Globalmacro added successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "globalmacroids": ["107819", "107820"]
},
"id":3
}
```

Globalmacro already exists:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CGlobalmacro::create ] Macro [ {$MACRO1} ] already exists"
},
"id":3
}
```

**deleteGlobal()**

Available since version: **1.8**
This function allows you to delete information about one or several globalmacros. All globalmacro-related information will be removed including items, graphs, macros, application, historical data, etc.

**Parameters** Array of Globalmacro macros

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Globalmacro IDs. |
| **error** | In case of any errors |

**Example** Delete globalmacros by macro

```
{
"jsonrpc": "2.0",
"method": "usermacro.deleteGlobal",
"params": [
    "{$MACRO3}",
    "{$MACRO4}"
],
"auth": "3a57200802b24cda67c4e4010b50c065",
"id": 2
}
```

Globalmacros deleted successfully:

```
{
"jsonrpc": "2.0",
"result": {
   "globalmacroids": ["107824", "107825"]
},
"id": 2
}
```

Globalmacro does not exist:

```
{
"jsonrpc": "2.0",
"error": {
      "code": -32500,
      "message": "Application error.",
      "data": "[ CGlobalmacro::delete ] Globalmacro does not exist"
},
"id": 2
}
```


**deleteHostMacro()**

Available since version: **1.8**
This function allows you to delete information about one or several hostmacros. All hostmacro-related information will be removed including items, graphs, macros, application, historical data, etc.

**Parameters** Array of Hostmacro IDs

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of deleted Hostmacro IDs. |
| **error** | In case of any errors |

**Example** Delete hostmacros by macrohost macro **ID**


```
{
"jsonrpc":"2.0",
"method":"usermacro.deleteHostMacro",
"params":["107824", "107825"],
"auth":"3a57200802b24cda67c4e4010b50c065",
"id":2
}
```


Hostmacros deleted successfully:


```
{
"jsonrpc":"2.0",
"result":{
   "hostmacroids": ["107824", "107825"]
},
"id":2
}
```


Hostmacro does not exist:


```
{
"jsonrpc":"2.0",
"error":{
      "code":-32500,
      "message":"Application error.",
      "data":"[ CHostmacro::delete ] Hostmacro does not exist"
},
"id":2
}
```


**get()**

Available since version: **1.8**
This function allows you to retrieve usermacro details based on filtering options. All parameters are optional. If parameter is set in query this option counted as ON, except if parameter is equal to NULL,

**Parameters**

| Parameter | Type | Description | Details |
|---|---|---|---|
| **nodeids** | *array* | Node IDs | |
| **groupids** | *array* | Host Group IDs | |
| **hostids** | *array* | Select all macros from the given hosts and templates. | |
| **templateids** | *array* | Select all host macros from hosts that are linked to the given templates. | |
| **hostmacroids** | *array* | Host Usermacro IDs | |
| **globalmacroids** | *array* | Host Usermacro IDs | |
| **globalmacro** | *integer* | Search only global macros | |
| **editable** | *integer* | only with read-write permission. Ignored for SuperAdmins | |
| **filter** | *array* | Optional filter by usermacro fields | |
| **search** | *array* | Return user macros by any given usermacro object field pattern | |
| **startSearch** | *integer* | Search usermacros field pattern only in start of the field | |
| **excludeSearch** | *integer* | Exclude from result, usermacros by given field pattern | |
| **searchWildcardsEnabled** | *integer* | Search pattern in whole field using wildcards | 1 - enable, 0 - disable |
| **output** | *string* | Output options | Values: shorten, refer, extend |
| **select_groups** | *string* | Select host groups | Values: shorten, refer, extend |
| **select_hosts** | *string* | Select hosts | Values: shorten, refer, extend |
| **select_templates** | *string* | Select templates | Values: shorten, refer, extend |
| **countOutput** | *integer* | Count usermacros, return the number of usermacros found | |
| **preservekeys** | *integer* | Return hash instead of array | Keys of hash are object IDs |
| **sortfield** | *string* | Sort by usermacro field | Values: macro |
| **sortorder** | *string* | Sort order | Values: ASC, DESC |
| **limit** | *int* | max number of usermacro objects to return | |

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of Usermacro objects. |
| **error** | In case of any errors |

**Example** Get host usermacros details by usermacro name "{$AAA}" in specified host groups **ID** "100100000000011","100100000000099","

```
{
"jsonrpc": "2.0",
"method": "usermacro.get",
"params": {
    "groupids": ["100100000000011","100100000000099","100100000010034"],
    "filter": {"macro": "{$AAA}"},
    "output": "extend"
},
```

```
"auth": "6f38cddc44cfbb6c1bd186f9a220b5a0",
"id": 2
}
```

Retrieved host usermacro details:

```
{
"jsonrpc": "2.0",
"result": [{
    "groups": [
        {"groupid": "100100000000011"}
    ],
    "hosts": [
        {"hostid": "100100000010077"}
    ],
    "hostmacroid": "100100000000005",
    "hostid": "100100000010077",
    "macro": "{$AAA}",
    "value": "aaaa"
}],
"id": 2
}
```

**massAdd()**

Available since version: **1.8**

**Parameters** Multidimensional array with usermacros data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **macros** | *array* | | Usermacros to add. | |
| **hosts** | *array* | Yes | Host objects that should get added usermacros. | |
| **templates** | *array* | Yes | Template objects that should get added usermacros. | |

NOTE: one of the **hosts** or **templates** is required.

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of added host/template usermacro IDs. |
| **error** | In case of any errors |

**Example** Add two usermacros on two hosts with **IDs** ”10092”, ”10086” and on two templates with **IDs** ”10052”, ”10053”:

```
{
"jsonrpc":"2.0",
"method":"usermacro.massAdd",
"params":{
    "macros": [
        {"macro": "{$MACRO1}", "value": "MACRO1"},
        {"macro": "{$MACRO2}", "value": "MACRO2"}
    ],
    "hosts": [
        {"hostid": "10092"},
        {"hostid": "10086"}
    ],
```

```
    "templates": [
        {"templateid": "10052"},
        {"templateid": "10053"}
    ]
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

Host and template usermacros added successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "hostmacroids":["42","43","44","45","46","47","48","49"]
},
"id":2
}
```

NOTE: host and/or template usermacros will be listed together in the ”hostmacroids” array.

Error when some host or template does not exist:

```
{
"jsonrpc":"2.0",
"error":{
    "code":-32500,
    "message":"Application error.",
    "data":"[ CUserMacro::massUpdate ] You do not have permission to perform this operation"
},
"id":2
}
```

**massRemove()**

Available since version: **1.8**

**Parameters** Multidimensional array with usermacros data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **macros** | *array* | | Usermacro to remove. | |
| **hostids** | *array* | Yes | Hostids that should have usermacros removed. | |
| **templateids** | *array* | Yes | Templateids that should have usermacros removed. | |

NOTE: one of the **hosts** or **templates** is required.

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of removed host/template usermacro IDs. |
| **error** | In case of any errors |

**Example** Remove two host usermacros from two hosts with **IDs** ”10092”, ”10086” and from two templates with **IDs** ”10052”, ”10053”::

```
{
"jsonrpc":"2.0",
"method":"usermacro.massRemove",
"params":{
    "macros": ["{$MACRO1}","{$MACRO2}"],
    "hostids": ["10092", "10086"],
    "templateids": ["10052", "10053"]
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

Host usermacros removed successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "hostmacroids":["42","43","44","45","46","47","48","49"]
},
"id":2
}
```

NOTE: host and/or template usermacros will be listed together in the ”hostmacroids” array.

Error when some host or template does not exist:

```
{
"jsonrpc":"2.0",
"error":{
    "code":-32500,
    "message":"Application error.",
    "data":"[ CUserMacro::massUpdate ] You do not have permission to perform this operation"
},
"id":2
}
```

**massUpdate()**

Available since version: **1.8**

**Parameters** Multidimensional array with usermacros data

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **macros** | *array* | | Usermacros to update. | |
| **hosts** | *array* | Yes | Host objects where usermacros should be updated. | |
| **templates** | *array* | Yes | Template objects where usermacros should be updated. | |

NOTE: one of the **hosts** or **templates** is required.

**Returns**

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated host/template usermacro IDs. |
| **error** | In case of any errors |

**Example** Update two usermacros on two hosts with **IDs** ”10092”, ”10086” and on two templates with **IDs** ”10052”, ”10053”:

```
{
"jsonrpc":"2.0",
"method":"usermacro.massUpdate",
"params":{
    "macros": [
        {"macro": "{$MACRO1}", "value": "MACRO1"},
        {"macro": "{$MACRO2}", "value": "MACRO2"}
    ],
    "hosts": [
        {"hostid": "10092"},
        {"hostid": "10086"}
    ],
    "templates": [
        {"templateid": "10052"},
        {"templateid": "10053"}
    ]
},
"auth":"f223adf833b2bf2ff38574a67bba6372",
"id":2
}
```

Host and template usermacros updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
    "hostmacroids":["42","43","44","45","46","47","48","49"]
},
"id":2
}
```

NOTE: host and/or template usermacros will be listed together in the ”hostmacroids” array.

Error when some host or template does not exist:

```
{
"jsonrpc":"2.0",
"error":{
    "code":-32500,
    "message":"Application error.",
    "data":"[ CUserMacro::massUpdate ] You do not have permission to perform this operation"
},
"id":2
}
```

#### updateGlobal()

This method allows you to update a globalmacro as defined by the **globalmacro data** array.

#### Parameters

| Parameter | Type | Optional | Description | Details |
|---|---|---|---|---|
| **globalmacro data** | *array or object* | No | Array of Globalmacro objects or a single object | **macro** name must be specified |

#### Returns

| Parameter | Description |
|---|---|
| **result** | Operation successful. Result will contain array of updated Globalmacro IDs. **globalmacroid** are assigned to each Globalmacro object |
| **error** | In case of any errors |

```
{
"jsonrpc":"2.0",
"method":"usermacro.updateGlobal",
"params":[
    {
        "macro":"{$MACRO1}",
        "value":"NEW VALUE"
    },
    {
        "macro":"{$MACRO2}",
        "value":"NEW VALUE"
    }
],
"auth":"038e1d7b1735c6a5436ee9eae095879e",
"id":3
}
```

**Example** Globalmacro updated successfully:

```
{
"jsonrpc":"2.0",
"result":{
   "globalmacroids": ["107819", "107820"]
},
"id":3
}
```

Globalmacro doesn't exist:

```
{
"jsonrpc":"2.0",
"error":{
      "code":-32602,
      "message":"Invalid params.",
      "data":"[ CGlobalmacro::update ] Macro [ {$MACRO1} ] does not exists"
},
"id":3
}
```

## Example API session

An example Zabbix API session might look like this. See Zabbix API introduction for more details.

Query:

```
{
    "jsonrpc": "2.0",
    "method": "user.authenticate",
    "params": {
        "user": "Admin",
        "password": "zabbix"
    },
    "auth": null,
    "id": 0
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": "13f28ca608a4b12c83a32d749229da71",
    "id": 0
}
```


Query:


```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": "extend"
    },
    "auth": "13f28ca608a4b12c83a32d749229da71",
    "id": 2
}
```


Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "maintenances": [
                {
                    "maintenanceid": "0"
                }
            ],
            "hostid": "10017",
            "proxy_hostid": "0",
            "host": "Zabbix server",
            "dns": "",
            "useip": "1",
            "ip": "127.0.0.1",
            "port": "10050",
            "status": "0",
            "disable_until": "0",
            "error": "",
            "available": "1",
            "errors_from": "0",
            "lastaccess": "0",
            "inbytes": "0",
            "outbytes": "0",
            "useipmi": "0",
            "ipmi_port": "623",
            "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_ip": "",
            "ipmi_errors_from": "0",
            "snmp_errors_from": "0",
            "ipmi_error": "",
            "snmp_error": ""
```

```
        }
    ],
    "id": 2
}
```

## Getting started with Zabbix API

### What is Zabbix API

Normally, you have only one way, to manipulate, configure and create objects in Zabbix - through it's PHP frontend. This is great, but only until you decide to build something custom: create a batch add/update script, or a custom monitoring tool, or anything else, that is not provided by default Zabbix GUI interface.

That's when Zabbix API comes to the rescue. It allows you to create, update and fetch Zabbix objects (like hosts, items, graphs and others) through JSON RPC protocol and do whatever you like (if you have an account authorized for that, of course).

Zabbix API was introduced in version 1.8 and still is under heavy development. In some parts, functionality is still limited, but it promises to become much wider with the release of Zabbix 2.0.

**Example session** For a quick overview, take a look at an example Zabbix API session or read below for detailed explanation.

### Using JSON RPC

If you are unfamiliar with JSON RPC, fear not, there is noting complicated there. All the workflow falls to several steps:

1. Prepare JSON object, that describes what you want to do (create host, fetch graph, update item etc.);
2. Send this object using POST method to http://example.com/zabbix/api_jsonrpc.php, where http://example.com/zabbix/ is the address of your Zabbix frontend;
3. Get a response with desired data in JSON format.

In most cases, you will do this from scripts, using your scripting language tools, but, of course, you can send requests "by hand", using any of the JSON RPC tools you desire.

Actually, that's it! All you need to know now, is how to authenticate for this and what format of JSON is Zabbix expecting to get from you.

### Basic request format

Simplified JSON request to Zabbix API looks like this:

```
{
   "jsonrpc": "2.0",
   "method": "method.name",
   "params": {
      "param_1_name": "param_1_value",
      "param_2_name": "param_2_value"
   },
   "id": 1,
   "auth": "159121b60d19a9b4b55d49e30cf12b81"
}
```

Lets look at it line by line:

- "jsonrpc": "2.0" - this is a standard JSON PRC parameter identifying protocol version. It will remain unchanged for all your requests;
- "method": "method.name" - this parameter defines actual operation to perform. Common examples: "host.create", "item.update" and so on;
- "params" - here you pass the JSON object with parameters required for specific method. If you would like to create an item, for example, "name" and "key_" parameters will be required. Possible parameters for each methods (and methods themselves) are described Zabbix API documentation;
- "id": 1 - this field can be used to tie every JSON request to it's response. The response will have the same "id" as provided by the request. It is useful when you are sending multiple requests at once. These are not required to be unique or sequential;
- "auth": "159121b60d19a9b4b55d49e30cf12b81" - this is an authentication token to identify the user, accessing the API. See Authenticating section below for more information.

**Authenticating**

So, now we know how to use the API. Let's take a peek at host.create method and create a new host. Let's send the request:

```
{
   "jsonrpc": "2.0",
   "method": "host.create",
   "params": {
      "host": "My first host name",
      "ip": "192.168.3.1",
      "port": 10050,
      "useip": 1,
      "groups": [
         {
            "groupid": 50
         }
      ]
   },
   "id": 1
}
```

Zabbix responds:

```
{
    "jsonrpc": "2.0",
    "error": {
        "code": -32602,
        "message": "Invalid params.",
        "data": "Not authorized"
    },
    "id": 1
}
```

What happened? Of course, no random person can send request to Zabbix to fetch the info or to modify something. That's why you need to be authenticated in order to do anything.

Good time to notice few things:

In case of any error, you get "error" parameter in the result:

- "code" parameter will always be -32602 (it's the JSON error code for invalid parameters);
- "message" reflects the same information that "code" gave us and won't differ too much;
- "data" will describe what actually went wrong.

In case of a success, you will get "result" parameter instead of "error" (as you will see later).

So, how to get authenticated? All you need is to send a request, calling "user.login" method and providing "user" and "password" as parameters.

```
{
   "jsonrpc": "2.0",
   "method": "user.login",
   "params": {
      "user": "Admin",
      "password": "zabbix"
   },
   "id": 1
}
```

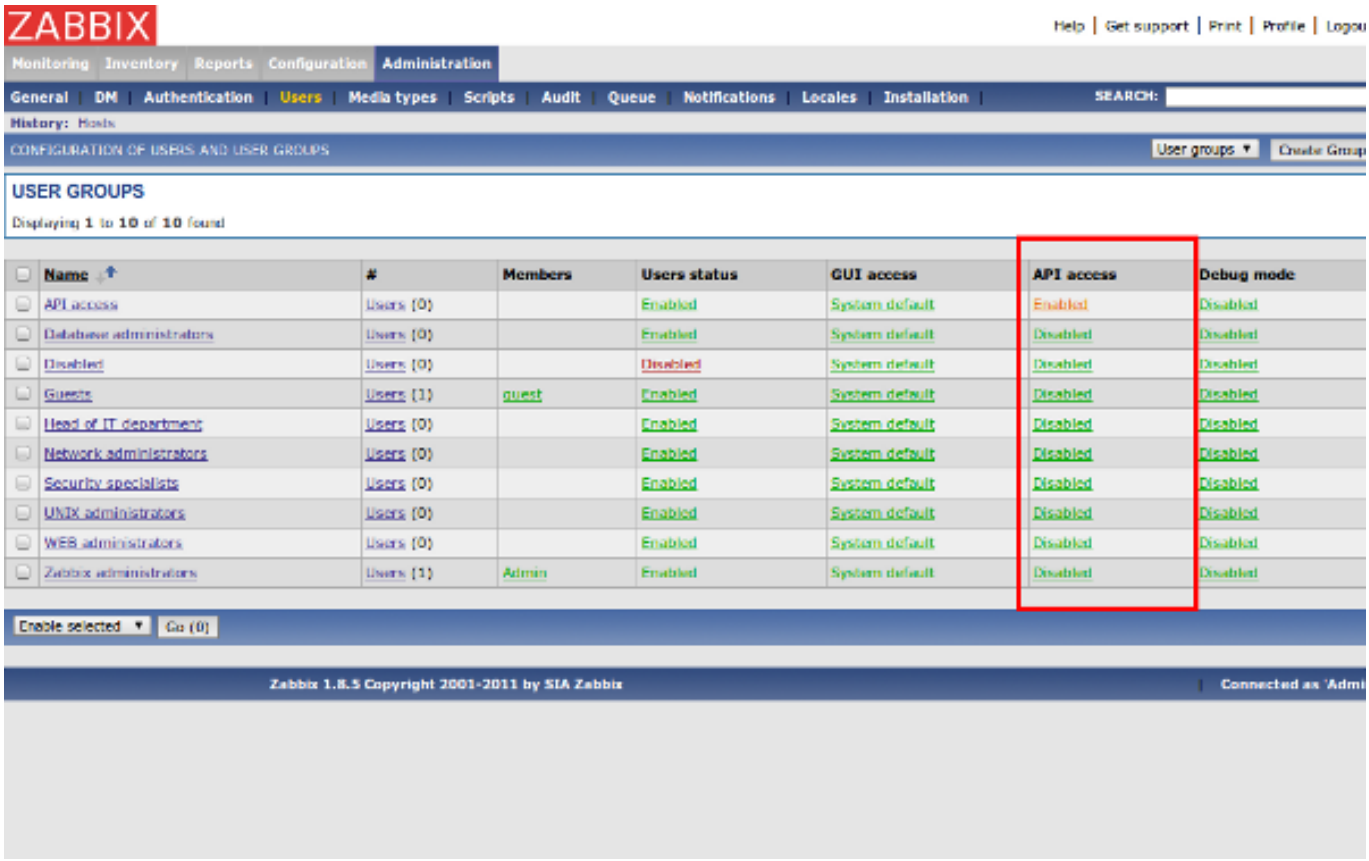
"Admin/zabbix" are default Zabbix credentials, but you have probably changed Admin's password by how. Haven't you?

So, we get the response:

```
{
    "jsonrpc": "2.0",
    "error": {
        "code": -32602,
        "message": "Invalid params.",
        "data": "No API access"
```

```
    },
    "id": 1
}
```

Failure, again. What happened this time? Thing is, in Zabbix 1.8, users that are not in ”API access” group do not have an access to Zabbix API by default. In order to use API with the given user, you need to set ”API access” to ”Enabled” for the user group of that user or place that user into a predefined ”API access” group.




Now, when your user is a member of user group with ”API access” enabled, let’s try the same request again:

```
{
   "jsonrpc": "2.0",
   "method": "user.login",
   "params": {
      "user": "Admin",
      "password": "zabbix"
   },
   "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "7cd4e1f5ebb27236e820db4faebc1769",
    "id": 1
}
```

Hooray! Authentication successful! What now? Now you can use hash, returned in ”result” parameter, as a proof of your rights, by including it with every API call you make, as an ”auth” parameter.

#### Usage examples and common parameters

Now, that you are authenticated, you can go on and actually do something. First of all, let’s try and fetch some info.

**Getting host groups** Here is a simple request to get all available host groups ordered by name:

```
{
   "jsonrpc": "2.0",
   "method": "hostgroup.get",
   "params": {
      "output": "extend",
      "sortfield": "name"
   },
   "id": 1,
   "auth": "7cd4e1f5ebb27236e820db4faebc1769"
}
```

Notice, that "method" contains "hostgroup.get", actual procedure that you are executing, and "params" contain additional options.

"sortfield", as you can guess, allows to sort result you get by chosen field.

"output":"extend" means that you want to get all available info about each group. This, in a way, is similar to "SELECT *" in SQL. Possible options of "output" are:

- "extend" - get all info;
- "shorten" - get only ids of an object;
- "refer" - get id of an object and also ids of related objects;
- list of fields, like ["groupid", "name"] - get only listed fields.

**Attention:**
List of fields is only supported in Alert, DCheck, Host, DService, Screenitem, Template and Trigger *get* methods.

And don't forget about the "auth" hash that you got using "user.login".

The response of given request might look like this:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "groupid": "5",
            "name": "Discovered hosts",
            "internal": "1"
        },
        {
            "groupid": "2",
            "name": "Linux servers",
            "internal": "0"
        },
        {
            "groupid": "1",
            "name": "Templates",
            "internal": "0"
        },
        {
            "groupid": "3",
            "name": "Windows servers",
            "internal": "0"
        },
        {
            "groupid": "4",
            "name": "Zabbix servers",
            "internal": "0"
        }
    ],
    "id": 1
}
```

These are standard groups, created by initial Zabbix configuration. Notice "groupid" field, the "XXXXid" fields are unique system identifiers, that will be used to address the object from another requests. See the next section for explanation.

**Creating host** We fetched the host groups, now let's try creating something. Let's create a host, that will be inside of the user groups "Linux servers" and "Zabbix servers". The request will look like this:

```
{
   "jsonrpc": "2.0",
   "method": "host.create",
   "params": {
      "host": "My new fancy host that I have created using API",
      "ip": "192.168.3.1",
      "port": 10050,
      "useip": 1,
      "groups": [
         {
            "groupid": 2
         },
         {
            "groupid": 4
         }
      ]
   },
   "id": 1,
   "auth": "7cd4e1f5ebb27236e820db4faebc1769"
}
```

Notice, that we are using "groupid" fields that we got earlier, to reference the groups we want our host to be in. We, say, that we want host to be in groups with ids 2 (Linux servers) and 4 (Zabbix servers). This is the way you will be working with all related objects.

If everything goes right, you will get a response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10051"
        ]
    },
    "id": 1
}
```

"hostids" list contains ids of the elements we have just created. In our case, we were creating just one host and got it's id - 10051. You can use it in future requests.

**Updating item** Of course, if you can create something, you should be able to update or delete something as well. And you are. Lest try and update an item. I have created item with description "agent.ping" at "My new fancy host that I have created using API" we created earlier, so we can play around with it. First, let's take a look at it:

Request:

```
{
   "jsonrpc": "2.0",
   "method": "item.get",
   "params": {
      "output": "extend",
      "filter": {
         "description": "agent.ping"
      },
      "hostids": [
         "10051"
      ]
   },
   "id": 1,
   "auth": "7cd4e1f5ebb27236e820db4faebc1769"
}
```

Note, that here we have used "filter" parameter, to specify item description and "hostids", to say that we are interested in item

that is on the host we just created (it had and ID of 10051, remember?)

Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hosts": [
                {
                    "hostid": "10051"
                }
            ],
            "itemid": "22162",
            "type": "0",
            "snmp_community": "",
            "snmp_oid": "",
            "snmp_port": "161",
            "hostid": "10051",
            "description": "agent.ping",
            "key_": "agent.ping",
            "delay": "30",
            "history": "90",
            "trends": "365",
            "lastvalue": null,
            "lastclock": null,
            "prevvalue": null,
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "multiplier": "0",
            "delta": "0",
            "prevorgvalue": null,
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "formula": "0",
            "error": "",
            "lastlogsize": "0",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "delay_flex": "",
            "params": "",
            "ipmi_sensor": "",
            "data_type": "0",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "mtime": "0"
        }
    ],
    "id": 1
}
```


Wow, much info there. Let's try and update item, by changing "snmp_port" to 162 and "item type" to SNMPV1. item.update method is the right tool for this.

Request:

```
{
   "jsonrpc": "2.0",
   "method": "item.update",
   "params": {
      "itemid": "22162",
      "snmp_port": "162",
      "type": 1
   },
   "id": 1,
   "auth": "7cd4e1f5ebb27236e820db4faebc1769"
}
```

Note, that we have specified three parameters: "itemid", so that Zabbix would know which item to update (don't forget this one!) and the two parameters we want to change. By the way, how did I know, that "type": 1 means SNMPV1? Well, it's all in general item section.

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22162"
        ]
    },
    "id": 1
}
```

As usual, Zabbix returned an ID of affected item.

## Zabbix manpages

These are Zabbix manpages for Zabbix processes.

### zabbix_agentd

Section: Maintenance Commands (8)
Updated: 5 July 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 */etc/zabbix/zabbix_agentd.conf*. Use absolute path.

-p, --print
Print known items and exit.

-t, --test <*item key*>
Test single item and exit.

-h, --help
Display this help and exit.

-V, --version
Output version information and exit.

#### FILES

*/etc/zabbix/zabbix_agentd.conf*
Default location of Zabbix agent configuration file.

#### SEE ALSO

**zabbix_get**(8), **zabbix_proxy**(8), **zabbix_sender**(8), **zabbix_server**(8)

#### AUTHOR

Alexei Vladishev <alex@zabbix.com>

---

#### Index



---

This document was created by man2html, using the manual pages.
Time: 13:34:23 GMT, September 26, 2011

### 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



---

This document was created by man2html, using the manual pages.
Time: 13:34:23 GMT, September 26, 2011

### zabbix_proxy

Section: Maintenance Commands (8)
Updated: 4 August 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 */etc/zabbix/zabbix_proxy.conf*.

-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.

-h, --help
Display this help and exit.

-V, --version
Output version information and exit.

## FILES

*/etc/zabbix/zabbix_proxy.conf*
Default location of Zabbix proxy configuration file.

## SEE ALSO

**zabbix_agentd**(8), **zabbix_get**(8), **zabbix_sender**(8), **zabbix_server**(8)

## AUTHOR

Alexei Vladishev <alex@zabbix.com>

---

**Index**



---

This document was created by man2html, using the manual pages.
Time: 13:34:23 GMT, September 26, 2011

# zabbix_sender

Section: Maintenance Commands (8)
Updated: 5 July 2011
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 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*>
Specify agent configuration file for reading server details.

-z, --zabbix-server <*server*>
Hostname or IP address of Zabbix server.

-p, --port <*port*>
Specify port number of server trapper running on the server. Default is 10051.

-s, --host <*host*>
Specify host name as registered in Zabbix front-end. 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 **-** for standard input. Each line of file contains whitespace delimited: **<hostname> <key> <value>**. Specify **-** in **<hostname>** to use hostname from configuration file or **--host** argument.

-T, --with-timestamps
Each line of file contains whitespace delimited: **<hostname> <key> <timestamp> <value>**. This can be used with **--input-file** option. Timestamp should be specified in Unix timestamp format.

-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.

#### 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 /etc/zabbix/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.

#### SEE ALSO

**zabbix_agentd**(8), **zabbix_get**(8), **zabbix_proxy**(8), **zabbix_server**(8)

#### AUTHOR

Alexei Vladishev <alex@zabbix.com>

---

#### Index



---

This document was created by man2html, using the manual pages.
Time: 13:34:23 GMT, September 26, 2011

### zabbix_server

Section: Maintenance Commands (8)
Updated: 4 August 2011
Index Return to Main Contents

---

#### NAME

zabbix_server - Zabbix server daemon.

#### SYNOPSIS

**zabbix_server [-hV] [-c** <*config-file*>] [-n <*nodeid*>] [-R <*option*>]

#### DESCRIPTION

**zabbix_server** is a core daemon of Zabbix software.

**Options** -c, --config <*config-file*>
Use the alternate *config-file* instead of */etc/zabbix/zabbix_server.conf*.

-n, --new-nodeid <*nodeid*>
Convert database data to new *nodeid*.

-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.

-h, --help
Display this help and exit.

-V, --version
Output version information and exit.

**FILES**

*/etc/zabbix/zabbix_server.conf*
Default location of Zabbix server configuration file.

**SEE ALSO**

**zabbix_agentd**(8), **zabbix_get**(8), **zabbix_proxy**(8), **zabbix_sender**(8)

**AUTHOR**

Alexei Vladishev <alex@zabbix.com>

---

**Index**



---

This document was created by man2html, using the manual pages.
Time: 13:34:23 GMT, September 26, 2011

## Zabbix Protocols

This is description of all protocols used by Zabbix 1.8 components.

There are some definitions used in the protocol details:

```
<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
```

# 1 Zabbix Agent

Zabbix uses JSON based communication protocol for communication with Zabbix Agent.

**1.1 Passive checks** Passive check is a simple data request. Zabbix Server (or Proxy) asks for a data, for example, CPU load, and Zabbix Agent sends the result back to the Server.

**Server Request**

```
<item key>\n
```

**Agent Response**

```
<HEADER><DATALEN><DATA>
```

For example:

1. Server opens 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

**1.2 Active checks** Active checks requires more complex processing. The Agent must retrieve list of items for independent processing first. It also periodically sends new values to the Server.

#### 1.2.1 Get list of items Agent Request

```
<HEADER><DATALEN>{
   "request":"active checks",
   "host":"<hostname>"
}
```

**Server Response**

```
{
    "response":"success",
    "data":[
    {
        "key":"log[\/home\/zabbix\/logs\/zabbix_agentd.log]",
        "delay":"30",
        "lastlogsize":"0"
    },
    {
        "key":"agent.version",
        "delay":"600"
    }
    ]
}
```

The Server must respond with success. For each returned item, **key** and **delay** must exist. For items having type "Log", the **lastlogsize** must exist as well.

For example:

1. Agent opens 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

#### 1.2.2 Send collected data Agent Sends

```
<HEADER><DATALEN>{
   "request":"agent data",
   "data":[
```

```
        {
            "host":"<hostname>",
            "key":"log[\/home\/zabbix\/logs\/zabbix_agentd.log]",
            "value":" 13039:20090907:184546.759 zabbix_agentd started. ZABBIX 1.6.6 (revision {7836}).",
            "lastlogsize":80,
            "clock":1252926015
        },
        {
            "host":"<hostname>",
            "key":"agent.version",
            "value":"1.6.6",
            "clock":1252926015
        }
    ],
    "clock":1252926016
}
```

**Server Response**

```
<HEADER><DATALEN>{
    "response":"success",
    "info":"Processed 2 Failed 0 Total 2 Seconds spent 0.002070"
}
```

For example:

1. Agent opens TCP connection
2. Agent sends list of values
3. Server processes the data and sends status back
4. TCP connection is closed